

Car Insurance Claims Classification Report

Hamed Vaheb

Contents

Introduction	1
Load Dataset	2
Preprocess	3
Clean Column Names	3
Categorical Columns	4
Missing Data	5
Encoding Categorical Columns	7
Split to Train/Test	9
Normalize	9
Oversampling	10
Classification Models	12
Generalized Linear Models	12
Random Forest	28
Extra Material: Decoding One-Hot-Encoding	36

Introduction

This report is dedicated to project of a workshop of master of data science at University of Luxembourg, I built the **aiinsurance**. You can install this package by first installing the devtools package and then using the following syntax:

```
devtools::install_github("https://github.com/berserkhmdvnb/aiinsurance")
```

You can explore the documentation of package I provided with

```
help(package="aiinsurance")
```

Also, this report is accessible as vignette of the package, which can be accessed with

```
vignette(package = "aiinsurance")
```

(Note: I just made a meta-reference)

Note that all functions of the package are suffixed with **hmd**.

Load Dataset

To provide a test case on a dataset with discrete target variable, I used the the car insurance data.

The dataset is embedded in the package, therefore by simply running the following commands one can load and display the dataset. . It can be used by the following syntax:

```
data("car_insurance_data")
```

```
data(car_insurance_data)
```

```
dplyr::glimpse(car_insurance_data)
```

```
## Rows: 10,000
## Columns: 19
## $ ID          <int> 569520, 750365, 199901, 478866, 731664, 877557, 93~
## $ AGE         <chr> "65+", "16-25", "16-25", "16-25", "26-39", "40-64"~
## $ GENDER      <chr> "female", "male", "female", "male", "male", "femal~
## $ RACE        <chr> "majority", "majority", "majority", "majority", "m~
## $ DRIVING_EXPERIENCE <chr> "0-9y", "0-9y", "0-9y", "0-9y", "10-19y", "20-29y"~
## $ EDUCATION   <chr> "high school", "none", "high school", "university"~
## $ INCOME      <chr> "upper class", "poverty", "working class", "workin~
## $ CREDIT_SCORE <dbl> 0.6290273, 0.3577571, 0.4931458, 0.2060129, 0.3883~
## $ VEHICLE_OWNERSHIP <int> 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1,~
## $ VEHICLE_YEAR <chr> "after 2015", "before 2015", "before 2015", "befor~
## $ MARRIED     <int> 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,~
## $ CHILDREN    <int> 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,~
## $ POSTAL_CODE <int> 10238, 10238, 10238, 32765, 32765, 10238, 10238, 1~
## $ ANNUAL_MILEAGE <int> 12000, 16000, 11000, 11000, 12000, 13000, 13000, 1~
## $ VEHICLE_TYPE <chr> "sedan", "sedan", "sedan", "sedan", "sedan", "seda~
## $ SPEEDING_VIOLATIONS <int> 0, 0, 0, 0, 2, 3, 7, 0, 0, 0, 6, 4, 4, 0, 0, 0, 10~
## $ DUIS        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 2,~
## $ PAST_ACCIDENTS <int> 0, 0, 0, 0, 1, 3, 3, 0, 0, 0, 7, 0, 2, 0, 1, 0,~
## $ OUTCOME     <int> 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,~
```

Summary of statistical properties of columns are provided in the following:

```
summary(car_insurance_data)
```

```
##           ID           AGE           GENDER           RACE
## Min.      :   101   Length:10000   Length:10000   Length:10000
## 1st Qu.:249638   Class :character   Class :character   Class :character
## Median :501777   Mode  :character   Mode  :character   Mode  :character
## Mean      :500522
## 3rd Qu.:753974
## Max.      :999976
##
## DRIVING_EXPERIENCE EDUCATION           INCOME           CREDIT_SCORE
## Length:10000       Length:10000       Length:10000       Min.      :0.0534
## Class :character   Class :character   Class :character   1st Qu.:0.4172
## Mode  :character   Mode  :character   Mode  :character   Median :0.5250
##                                     Mean      :0.5158
##                                     3rd Qu.:0.6183
```

```
##                                     Max.    :0.9608
##                                     NA's     :982
## VEHICLE_OWNERSHIP VEHICLE_YEAR      MARRIED      CHILDREN
## Min.    :0.000      Length:10000      Min.    :0.0000      Min.    :0.0000
## 1st Qu.:0.000      Class :character      1st Qu.:0.0000      1st Qu.:0.0000
## Median :1.000      Mode  :character      Median :0.0000      Median :1.0000
## Mean    :0.697                                Mean    :0.4982      Mean    :0.6888
## 3rd Qu.:1.000                                3rd Qu.:1.0000      3rd Qu.:1.0000
## Max.    :1.000                                Max.    :1.0000      Max.    :1.0000
##
## POSTAL_CODE      ANNUAL_MILEAGE VEHICLE_TYPE      SPEEDING_VIOLATIONS
## Min.    :10238      Min.    : 2000      Length:10000      Min.    : 0.000
## 1st Qu.:10238      1st Qu.:10000      Class :character      1st Qu.: 0.000
## Median :10238      Median :12000      Mode  :character      Median : 0.000
## Mean    :19865      Mean    :11697                                Mean    : 1.483
## 3rd Qu.:32765      3rd Qu.:14000                                3rd Qu.: 2.000
## Max.    :92101      Max.    :22000                                Max.    :22.000
##                                     NA's     :957
##      DUIS      PAST_ACCIDENTS      OUTCOME
## Min.    :0.0000      Min.    : 0.000      Min.    :0.0000
## 1st Qu.:0.0000      1st Qu.: 0.000      1st Qu.:0.0000
## Median :0.0000      Median : 0.000      Median :0.0000
## Mean    :0.2392      Mean    : 1.056      Mean    :0.3133
## 3rd Qu.:0.0000      3rd Qu.: 2.000      3rd Qu.:1.0000
## Max.    :6.0000      Max.    :15.000      Max.    :1.0000
##
```

Preprocess

Before feeding data to the model, various preprocessing stages are performed in the following subsections:

Clean Column Names

```
df <- janitor::clean_names(car_insurance_data)

names(df)
```

```
## [1] "id"          "age"          "gender"
## [4] "race"        "driving_experience" "education"
## [7] "income"      "credit_score"  "vehicle_ownership"
## [10] "vehicle_year" "married"       "children"
## [13] "postal_code" "annual_mileage" "vehicle_type"
## [16] "speeding_violations" "duis"         "past_accidents"
## [19] "outcome"
```

And the following is to make the codes reproducible, as random values might be involved in different stages:

```
set.seed(12345)
```

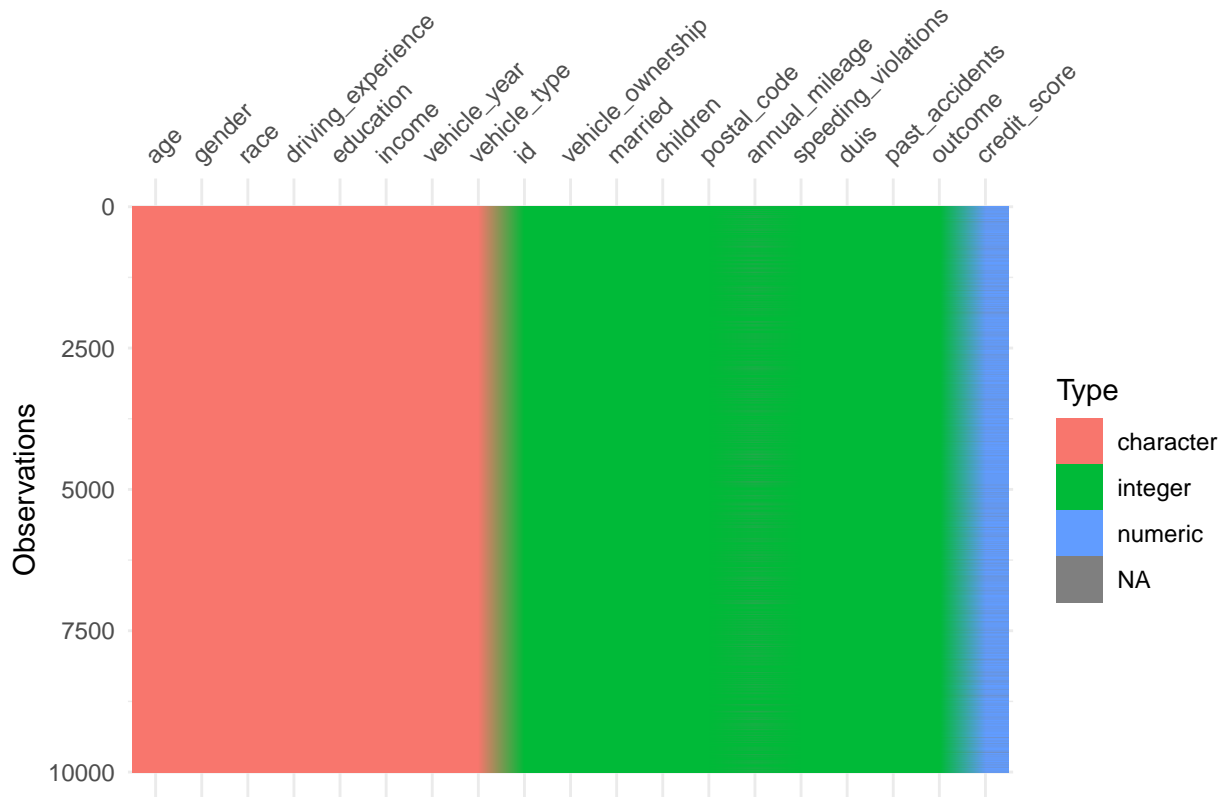
Categorical Columns

We need to factorise the categorical columns, which can be done by merely inputting columns intended to be converted in `cat_cols` in the `prepare_hmd` function. Before doing that, we report and visualise the nature of categorical data and their frequency in the dataset

```
categoricals_print_hmd(df)
```

```
## [1] "List of categorical columns containing characters: "  
## [1] "age"          "gender"       "race"  
## [4] "driving_experience" "education"    "income"  
## [7] "vehicle_year"  "vehicle_type"  
## [1] "List of categorical columns containing numbers: "  
## [1] "vehicle_ownership" "married"      "children"  
## [4] "postal_code"     "outcome"     "credit_score"
```

```
vis_dat(df)
```

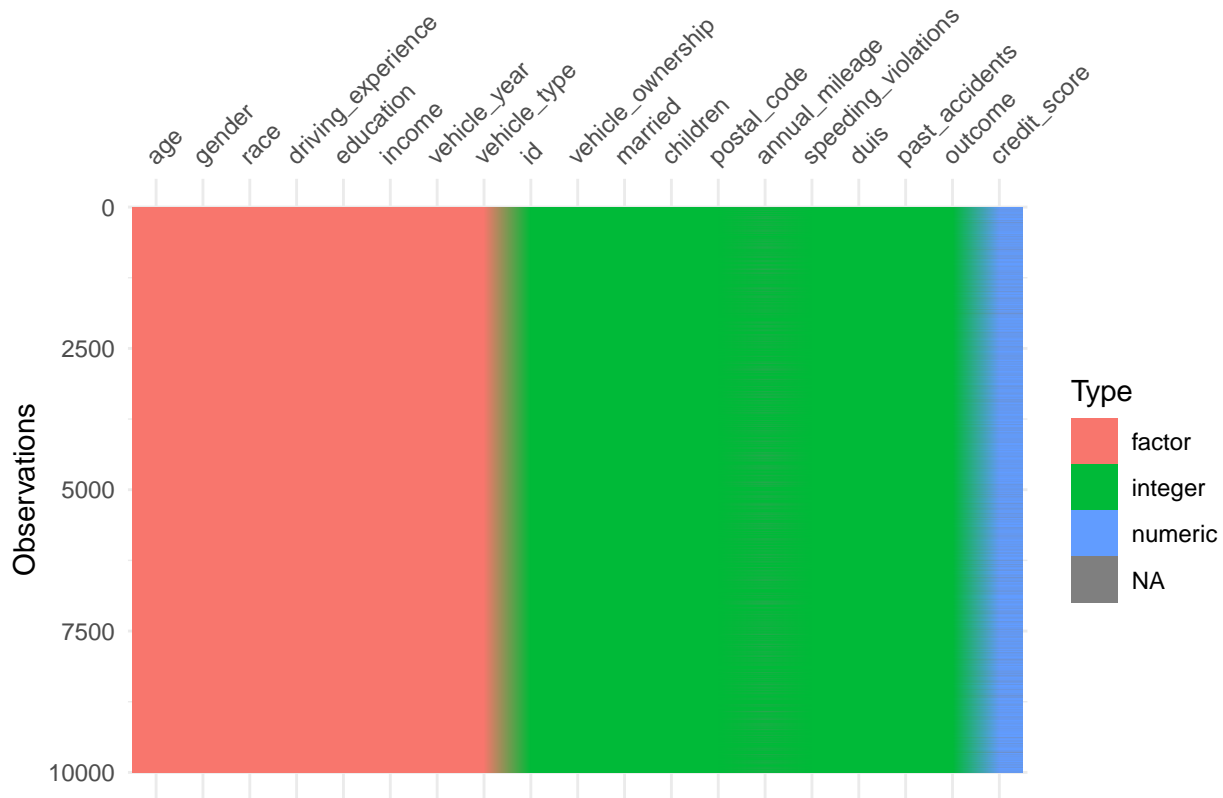


```
df <- categoricals_hmd(df)  
dplyr::glimpse(df)
```

```
## Rows: 10,000  
## Columns: 19  
## $ id          <int> 569520, 750365, 199901, 478866, 731664, 877557, 93~  
## $ age         <fct> 65+, 16-25, 16-25, 16-25, 26-39, 40-64, 65+, 26-39~  
## $ gender      <fct> female, male, female, male, male, female, male, fe~
```

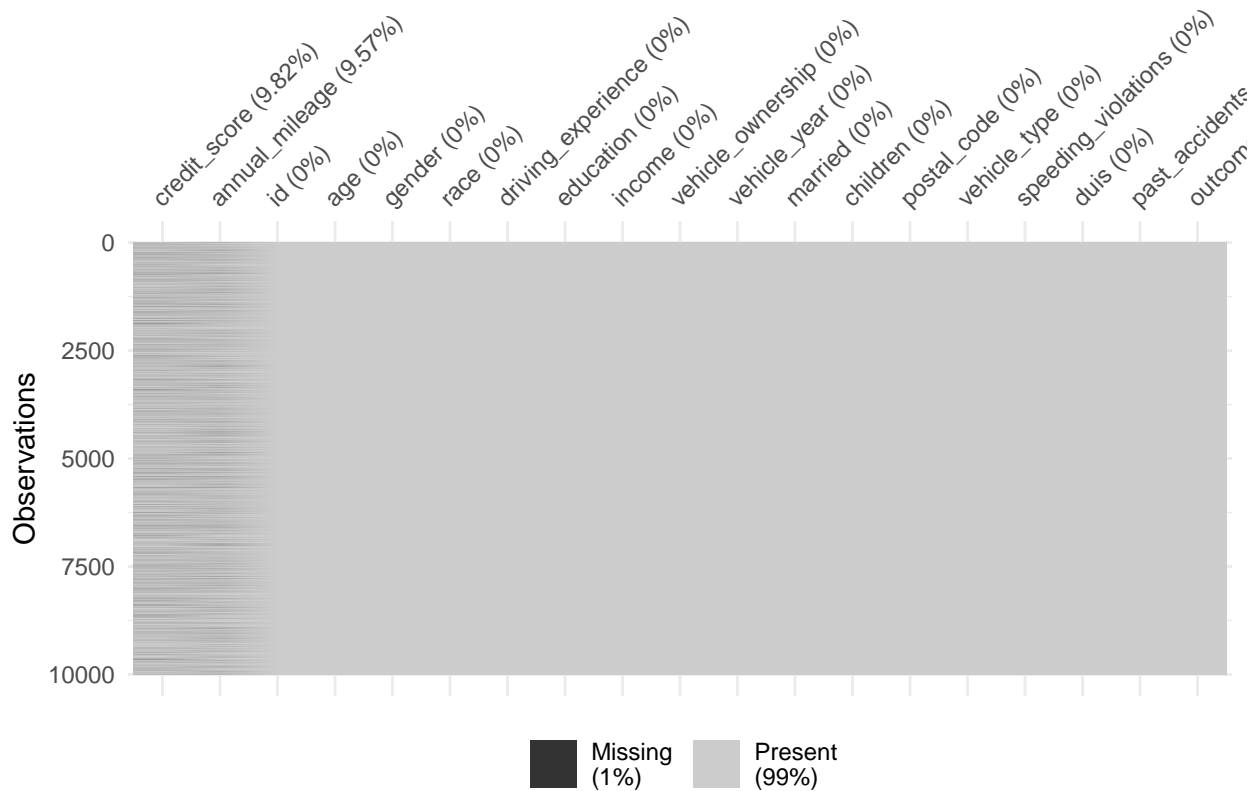
```
## $ race                <fct> majority, majority, majority, majority, majority, ~
## $ driving_experience   <fct> 0-9y, 0-9y, 0-9y, 0-9y, 10-19y, 20-29y, 30y+, 0-9y~
## $ education           <fct> high school, none, high school, university, none, ~
## $ income              <fct> upper class, poverty, working class, working class~
## $ credit_score        <dbl> 0.6290273, 0.3577571, 0.4931458, 0.2060129, 0.3883~
## $ vehicle_ownership   <int> 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1,~
## $ vehicle_year        <fct> after 2015, before 2015, before 2015, before 2015,~
## $ married             <int> 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1,~
## $ children            <int> 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1,~
## $ postal_code         <int> 10238, 10238, 10238, 32765, 32765, 10238, 10238, 1~
## $ annual_mileage       <int> 12000, 16000, 11000, 11000, 12000, 13000, 13000, 1~
## $ vehicle_type        <fct> sedan, sedan, sedan, sedan, sedan, sedan, sedan, s~
## $ speeding_violations <int> 0, 0, 0, 0, 2, 3, 7, 0, 0, 0, 6, 4, 4, 0, 0, 0, 10~
## $ dui                 <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 2, 0,~
## $ past_accidents      <int> 0, 0, 0, 0, 1, 3, 3, 0, 0, 0, 7, 0, 2, 0, 1, 0, 1,~
## $ outcome             <int> 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,~
```

```
vis_dat(df)
```



Missing Data

```
df[is.na(df) | df=="Inf"] = NA
vis_miss(df, sort_miss = TRUE)
```



```
is.na(df |> select("credit_score","annual_mileage")) |> colSums()
```

```
##   credit_score annual_mileage
##           982           957
```

Classification Models

As only two numerical columns include NA values, they will be imputed with the median of their respective columns. For this `impute_median_hmd` function is used from the package:

```
df <- impute_median_hmd(df)
is.na(df) |> colSums()
```

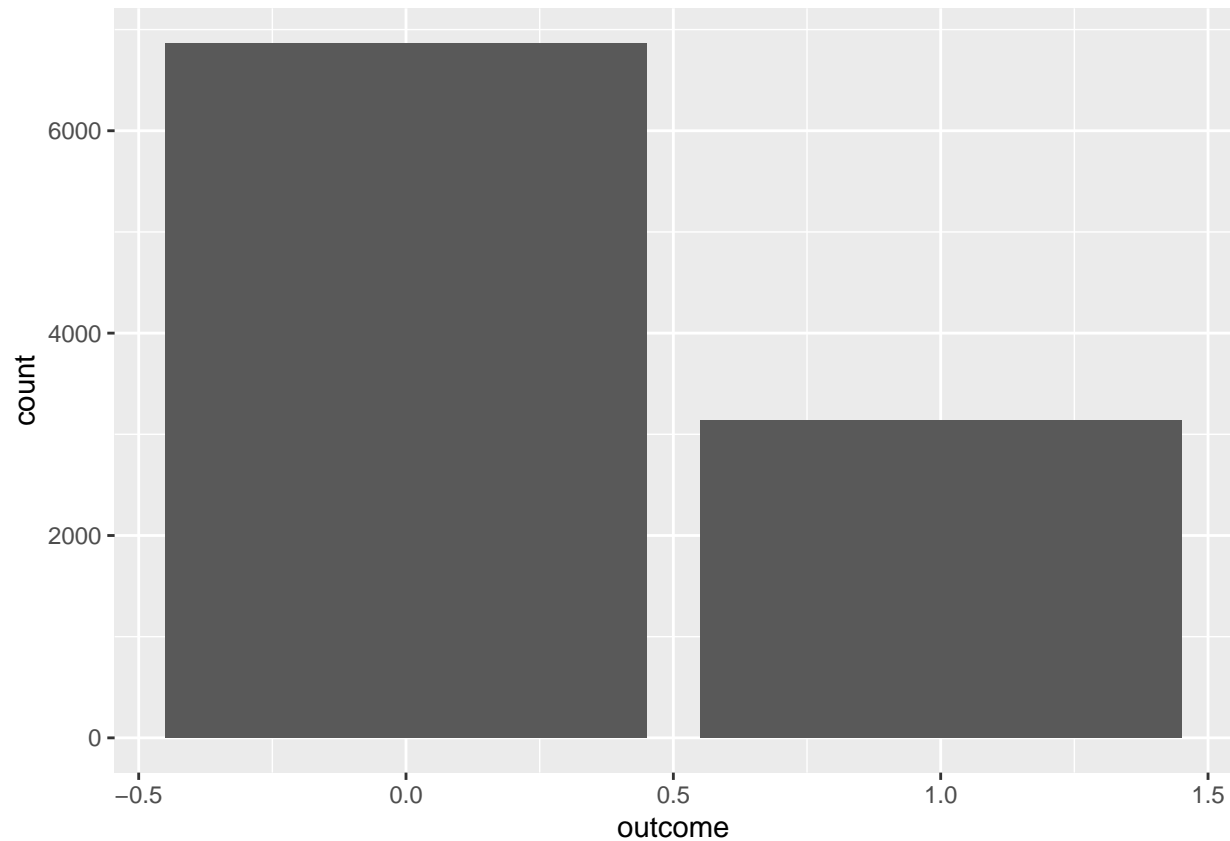
```
##           id           age           gender           race
##           0           0           0           0
## driving_experience education           income credit_score
##           0           0           0           0
## vehicle_ownership vehicle_year           married children
##           0           0           0           0
##           postal_code annual_mileage vehicle_type speeding_violations
##           0           0           0           0
##           dui past_accidents           outcome
##           0           0           0
```

As evident, there is no missing data anymore. ## Check Imbalance of Target

Visualize Balance

As imbalance of target classes affect predictions, the frequency of each class in the `outcome` column is visualized:

```
# Most basic bar chart
ggplot(df, aes(x = outcome)) +
  geom_bar()
```



```
table(df$outcome)
```

```
##
##    0    1
## 6867 3133
```

```
imbalance::imbalanceRatio(df, classAttr = "outcome")
```

```
## [1] 0.45624
```

Encoding Categorical Columns

Since later it is realized that data is imbalanced and therefore it should be resampled, it is required that data would only contain numerical columns for the next part. For this reason, one-hot-encoding is applied in the following:

```
dummy <- dummyVars(" ~ .", data=df)
df_enc <- data.frame(predict(dummy, newdata = df))
dplyr::glimpse(df_enc)
```

```
## Rows: 10,000
## Columns: 34
## $ id <dbl> 569520, 750365, 199901, 478866, 731664, 8775~
## $ age.16.25 <dbl> 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,~
## $ age.26.39 <dbl> 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,~
## $ age.40.64 <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,~
## $ age.65. <dbl> 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,~
## $ gender.female <dbl> 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0,~
## $ gender.male <dbl> 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1,~
## $ race.majority <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ race.minority <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ driving_experience.0.9y <dbl> 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0,~
## $ driving_experience.10.19y <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,~
## $ driving_experience.20.29y <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,~
## $ driving_experience.30y. <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,~
## $ education.high.school <dbl> 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1,~
## $ education.none <dbl> 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ education.university <dbl> 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,~
## $ income.middle.class <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,~
## $ income.poverty <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ income.upper.class <dbl> 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,~
## $ income.working.class <dbl> 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,~
## $ credit_score <dbl> 0.6290273, 0.3577571, 0.4931458, 0.2060129, ~
## $ vehicle_ownership <dbl> 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1,~
## $ vehicle_year.after.2015 <dbl> 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0,~
## $ vehicle_year.before.2015 <dbl> 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,~
## $ married <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,~
## $ children <dbl> 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,~
## $ postal_code <dbl> 10238, 10238, 10238, 32765, 32765, 10238, 10~
## $ annual_mileage <dbl> 12000, 16000, 11000, 11000, 12000, 13000, 13~
## $ vehicle_type.sedan <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ vehicle_type.sports.car <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ speeding_violations <dbl> 0, 0, 0, 0, 2, 3, 7, 0, 0, 0, 6, 4, 4, 0, 0,~
## $ dui <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 2,~
## $ past_accidents <dbl> 0, 0, 0, 0, 1, 3, 3, 0, 0, 0, 7, 0, 2, 0, 1,~
## $ outcome <dbl> 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,~
```

Ensuring the encoded dataframe didn't slip a NA value:

```
is.na(df_enc) |> colSums()
```

```
##           id           age.16.25           age.26.39
##           0                   0                   0
##      age.40.64           age.65.           gender.female
##           0                   0                   0
##      gender.male           race.majority           race.minority
##           0                   0                   0
##      driving_experience.0.9y driving_experience.10.19y driving_experience.20.29y
```



```
##          0          0          0
## driving_experience.30y. education.high.school education.none
##          0          0          0
## education.university income.middle.class income.poverty
##          0          0          0
## income.upper.class income.working.class credit_score
##          0          0          0
## vehicle_ownership vehicle_year.after.2015 vehicle_year.before.2015
##          0          0          0
## married children postal_code
##          0          0          0
## annual_mileage vehicle_type.sedan vehicle_type.sports.car
##          0          0          0
## speeding_violations dui past_accidents
##          0          0          0
## outcome
##          0
```

Split to Train/Test

```
df_dict <- train_test_splitter_hmd(df_enc,
                                   proportion=0.8)
train <- df_dict$train
test <- df_dict$test
```

Normalize

```
for (col in names(train)){
  if (col %in% c("annual_mileage", "postal_code", "credit_score"))
  {
    next
  }
  train[[col]] <- as.integer(train[[col]])
  test[[col]] <- as.integer(test[[col]])
}
```

```
df_dict_norm <- normalizer_hmd(train, test)
```

```
train <- df_dict_norm$train_norm
test <- df_dict_norm$test_norm
```

```
train$id <- as.double(train$id)
test$id <- as.double(test$id)
```

```
print(paste("train data has", nrow(train), "rows and", ncol(train), "columns"))
```

```
## [1] "train data has 8000 rows and 34 columns"
```

```
print(paste("test data has", nrow(test), "rows and", ncol(test), "columns"))
```

```
## [1] "test data has 2000 rows and 34 columns"
```

Oversampling

```
#train_racog <- racog(train,  
#                       numInstances = 3000,  
#                       burnin = 100,  
#                       lag = 20,  
#                       classAttr = "outcome")
```

Merge new sampled data with original train dataset

```
#insurance_train <- rbind(train , train_racog)
```

Save the processed dataframe to memory:

```
#path = "/home/hamed/Documents/R/aiinsurance/inst/"  
#file_name = "insurance_train.csv"  
#readr::write_csv(insurance_train,  
#                 file = #paste(path,file_name,sep=""))
```

Load back the processed data

```
#insurance_train <- #readr::read_csv(paste(path,file_name,sep=""))
```

Do the same for test data

```
#insurance_test <- test  
#path = "/home/hamed/Documents/R/aiinsurance/inst/"  
#file_name = "insurance_test.csv"  
#readr::write_csv(insurance_test,  
#                 file = paste(path,file_name,sep=""))  
  
#insurance_test <- #readr::read_csv(paste(path,file_name,sep=""))
```

we need to also remove the id of the beneficiaries as it will bias the learning towards this and may cause data leakage for test datasets with the same id

```
#insurance_train <- within(insurance_train, rm("id"))  
#insurance_test <- within(insurance_test, rm("id"))
```

```
data("insurance_train")  
data("insurance_test")
```

```
glimpse(insurance_train)
```

```
## Rows: 11,000
## Columns: 33
## $ age.16.25      <int> 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, ~
## $ age.26.39      <int> 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, ~
## $ age.40.64      <int> 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ age.65.        <int> 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ~
## $ gender.female  <int> 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, ~
## $ gender.male    <int> 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, ~
## $ race.majority  <int> 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, ~
## $ race.minority  <int> 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, ~
## $ driving_experience.0.9y <int> 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, ~
## $ driving_experience.10.19y <int> 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, ~
## $ driving_experience.20.29y <int> 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ driving_experience.30y. <int> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ~
## $ education.high.school <int> 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, ~
## $ education.none  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ~
## $ education.university <int> 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, ~
## $ income.middle.class <int> 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, ~
## $ income.poverty  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ income.upper.class <int> 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, ~
## $ income.working.class <int> 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, ~
## $ credit_score    <dbl> 0.7172432, 0.7894224, 0.2496466, 0.3613210, ~
## $ vehicle_ownership <int> 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, ~
## $ vehicle_year.after.2015 <int> 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, ~
## $ vehicle_year.before.2015 <int> 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, ~
## $ married        <int> 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, ~
## $ children        <int> 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, ~
## $ postal_code     <dbl> 0.0000000, 0.0000000, 0.0000000, 0.0000000, ~
## $ annual_mileage  <dbl> 0.60, 0.45, 0.50, 0.70, 0.50, 0.50, 0.45, 0.~
## $ vehicle_type.sedan <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, ~
## $ vehicle_type.sports.car <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ~
## $ speeding_violations <dbl> 0.22727273, 0.09090909, 0.04545455, 0.045454~
## $ dui            <dbl> 0.0000000, 0.0000000, 0.0000000, 0.0000000, ~
## $ past_accidents  <dbl> 0.13333333, 0.00000000, 0.00000000, 0.000000~
## $ outcome        <int> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, ~
```

```
glimpse(insurance_test)
```

```
## Rows: 2,000
## Columns: 33
## $ age.16.25      <int> 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, ~
## $ age.26.39      <int> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ~
## $ age.40.64      <int> 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, ~
## $ age.65.        <int> 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, ~
## $ gender.female  <int> 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, ~
## $ gender.male    <int> 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, ~
## $ race.majority  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ race.minority  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ driving_experience.0.9y <int> 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, ~
## $ driving_experience.10.19y <int> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, ~
```

```
## $ driving_experience.20.29y <int> 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, ~
## $ driving_experience.30y. <int> 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, ~
## $ education.high.school <int> 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, ~
## $ education.none <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, ~
## $ education.university <int> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, ~
## $ income.middle.class <int> 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, ~
## $ income.poverty <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, ~
## $ income.upper.class <int> 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, ~
## $ income.working.class <int> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, ~
## $ credit_score <dbl> 0.6234644, 0.6248241, 0.7454570, 0.6432089, ~
## $ vehicle_ownership <int> 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, ~
## $ vehicle_year.after.2015 <int> 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, ~
## $ vehicle_year.before.2015 <int> 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, ~
## $ married <int> 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, ~
## $ children <int> 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, ~
## $ postal_code <dbl> 0.0000000, 0.0000000, 0.2751793, 0.0000000, ~
## $ annual_mileage <dbl> 0.55, 0.40, 0.50, 0.30, 0.50, 0.50, 0.50, 0.~
## $ vehicle_type.sedan <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, ~
## $ vehicle_type.sports.car <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ~
## $ speeding_violations <dbl> 0.13636364, 0.27272727, 0.18181818, 0.181818~
## $ dui <dbl> 0.0000000, 0.3333333, 0.0000000, 0.1666667, ~
## $ past_accidents <dbl> 0.2000000, 0.4666667, 0.0000000, 0.133333~
## $ outcome <int> 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, ~
```

```
table(insurance_train$outcome)
```

```
##
##    0    1
## 5484 5516
```

I added a `preprocess_hmd` function in package that is a wrapper around all the steps and functions from the preprocessing section. Therefore, simply by writing the following syntax, the final `insurance_train` and `insurance_test` datasets can be obtained from the original `car_insurance_data`:

```
#h <- preprocess_hmd(car_insurance_data)
#insurance_train <- h$insurance_train
#insurance_test <- h$insurance_test
```

Data Visualisation

```
#insurance_train /> ggplot(aes(x = credit_score, y = #speeding_violations)) +
#   geom_point(color = outcome)
```

Classification Models

Generalized Linear Models

```
actual <- insurance_test$outcome
```

GLMNET

```
fit <- glmnet_fit_hmd(insurance_train, target="outcome", family="binomial")
h <- glmnet_predict_hmd(fit,
                        data = insurance_test,
                        target = "outcome",
                        type = "binomial")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
coef = h$coef
pred_glm <- h$predictions
pred_proba_glm <- h$predict_proba
```

```
summary(fit)
```

```
##
## Call:
## stats::glm(formula = glm_format, family = {
##     {
##         family
##     }
## }, data = df)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -3.2941  -0.5332   0.0968   0.5764   3.2898
##
## Coefficients: (4 not defined because of singularities)
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.17229    0.33125   0.520 0.602967
## age.16.25      0.12188    0.11505   1.059 0.289454
## age.26.39     -0.13679    0.09681  -1.413 0.157657
## age.40.64     -0.27321    0.10319  -2.648 0.008105 **
## age.65.       -0.12704    0.12689  -1.001 0.316755
## gender.female  -0.90848    0.05888 -15.430 < 2e-16 ***
## gender.male      NA          NA      NA      NA
## race.majority    0.11025    0.09034   1.220 0.222292
## race.minority    NA          NA      NA      NA
## driving_experience.0.9y  2.25425    0.18599  12.120 < 2e-16 ***
## driving_experience.10.19y 0.55940    0.17247   3.243 0.001181 **
## driving_experience.20.29y -1.04472    0.18714  -5.583 2.37e-08 ***
## driving_experience.30y. -1.65633    0.26683  -6.207 5.39e-10 ***
## education.high.school  0.20773    0.11088   1.873 0.061006 .
## education.none      0.38751    0.10403   3.725 0.000195 ***
## education.university  0.16342    0.10899   1.499 0.133778
## income.middle.class  0.19168    0.09211   2.081 0.037448 *
## income.poverty      0.25226    0.11247   2.243 0.024908 *
## income.upper.class   0.17092    0.10740   1.591 0.111518
## income.working.class  0.40319    0.09166   4.399 1.09e-05 ***
## credit_score      -0.14892    0.26484  -0.562 0.573910
## vehicle_ownership  -1.55159    0.06183 -25.095 < 2e-16 ***
## vehicle_year.after.2015 -1.65975    0.07259 -22.865 < 2e-16 ***
```

```
## vehicle_year.before.2015      NA      NA      NA      NA
## married                      -0.45440  0.06346 -7.161 8.03e-13 ***
## children                     -0.26748  0.06318 -4.233 2.30e-05 ***
## postal_code                   1.73352  0.12759 13.586 < 2e-16 ***
## annual_mileage                1.19594  0.25215  4.743 2.11e-06 ***
## vehicle_type.sedan            -0.17842  0.12571 -1.419 0.155821
## vehicle_type.sports.car       NA      NA      NA      NA
## speeding_violations           1.49235  0.44509  3.353 0.000800 ***
## duis                         -0.16071  0.38786 -0.414 0.678611
## past_accidents                -2.38947  0.45009 -5.309 1.10e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 15249.1 on 10999 degrees of freedom
## Residual deviance: 8434.7 on 10971 degrees of freedom
## AIC: 8492.7
##
## Number of Fisher Scoring iterations: 6
```

```
coef
```

```
## (Intercept)                age.16.25                age.26.39
## 0.1722946                  0.1218785                -0.1367859
## age.40.64                  age.65.                gender.female
## -0.2732099                 -0.1270374                -0.9084817
## gender.male                race.majority              race.minority
## NA                        0.1102514                  NA
## driving_experience.0.9y driving_experience.10.19y driving_experience.20.29y
## 2.2542490                  0.5593966                -1.0447194
## driving_experience.30y.    education.high.school      education.none
## -1.6563349                 0.2077277                0.3875125
## education.university      income.middle.class        income.poverty
## 0.1634237                  0.1916752                0.2522558
## income.upper.class        income.working.class        credit_score
## 0.1709167                  0.4031949                -0.1489177
## vehicle_ownership          vehicle_year.after.2015 vehicle_year.before.2015
## -1.5515944                 -1.6597537                NA
## married                    children                  postal_code
## -0.4544041                 -0.2674758                1.7335224
## annual_mileage            vehicle_type.sedan        vehicle_type.sports.car
## 1.1959431                 -0.1784208                NA
## speeding_violations        duis                    past_accidents
## 1.4923479                 -0.1607137                -2.3894743
```

```
head(pred_glm)
```

```
## 1 2 3 4 5 6
## 0 0 0 0 0 0
```

Extract AIC and BIC from the fit of glm

```
fit$aic
```

```
## [1] 8492.693
```

avplot of results

```
#glmnet_plot_hmd(fit)
```

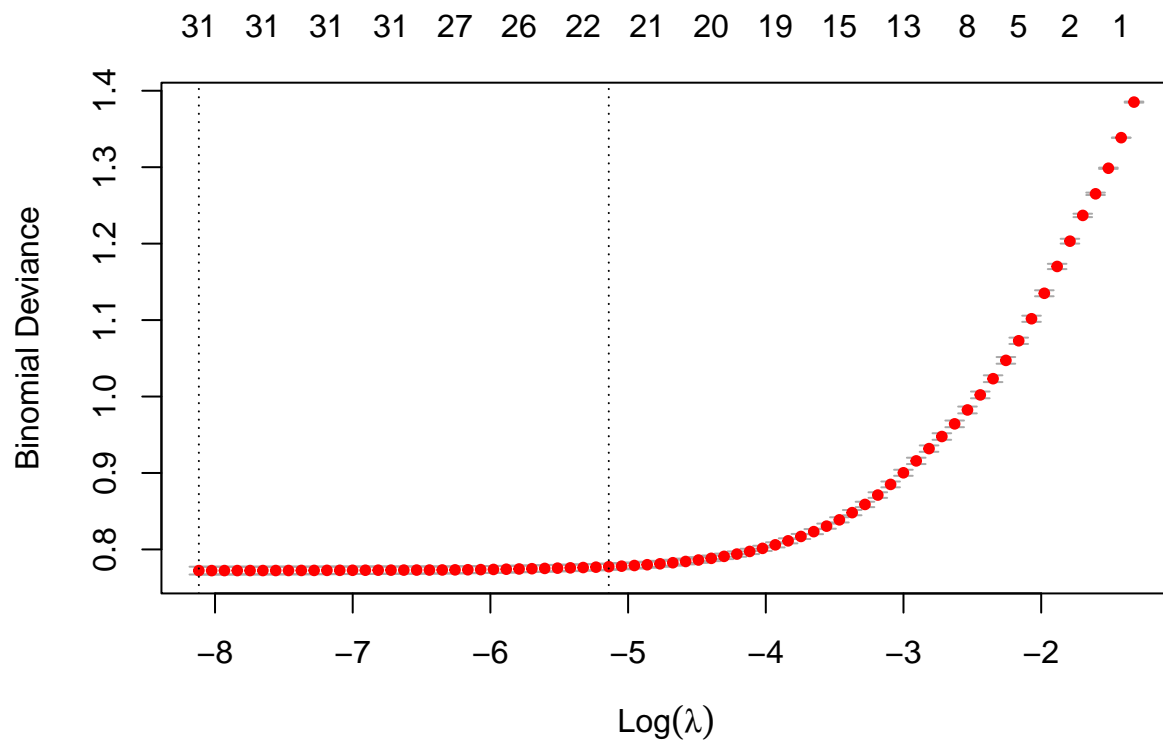
cross validated GLMNET

min

```
fit <- glmnet_cv_fit_hmd(insurance_train, target="outcome", family="binomial")
h <- glmnet_cv_predict_hmd(fit,
                           data = insurance_test,
                           target = "outcome",
                           lchoice = "min",
                           type = "binomial")

coef = h$coef
pred_glm_cv_min <- h$predictions
pred_proba_glm_cv_min <- h$predict_proba
```

```
glmnet_cv_plot_hmd(fit)
```



```
summary(fit)
```

```
##           Length Class  Mode
```

```
## lambda      74      -none- numeric
## cvm         74      -none- numeric
## cvsd        74      -none- numeric
## cvup        74      -none- numeric
## cvlo        74      -none- numeric
## nzero       74      -none- numeric
## call        4       -none- call
## name        1       -none- character
## glmnet.fit  13      lognet list
## lambda.min   1       -none- numeric
## lambda.1se   1       -none- numeric
## index       2       -none- numeric
```

```
coef
```

```
## 33 x 1 sparse Matrix of class "dgCMatrix"
##                               s1
## (Intercept)                2.606699e-01
## age.16.25                   1.470227e-01
## age.26.39                   -1.053756e-01
## age.40.64                   -2.477387e-01
## age.65.                     -9.772754e-02
## gender.female               -9.024817e-01
## gender.male                 5.554381e-05
## race.majority               1.002116e-01
## race.minority               .
## driving_experience.0.9y      2.191045e+00
## driving_experience.10.19y    5.082683e-01
## driving_experience.20.29y   -1.064324e+00
## driving_experience.30y.     -1.649878e+00
## education.high.school       1.627430e-01
## education.none              3.502781e-01
## education.university        1.262105e-01
## income.middle.class         1.579856e-01
## income.poverty              2.240788e-01
## income.upper.class          1.266337e-01
## income.working.class        3.765955e-01
## credit_score                -1.131380e-01
## vehicle_ownership           -1.540379e+00
## vehicle_year.after.2015     -1.644180e+00
## vehicle_year.before.2015    4.626151e-12
## married                    -4.489660e-01
## children                   -2.668132e-01
## postal_code                 1.714963e+00
## annual_mileage              1.160143e+00
## vehicle_type.sedan          -1.672808e-01
## vehicle_type.sports.car     1.407297e-13
## speeding_violations         1.359474e+00
## dui                        -1.350778e-01
## past_accidents              -2.372703e+00
```

```
head(pred_glm_cv_min)
```

```
##      lambda.min
```



```
## [1,]      0
## [2,]      0
## [3,]      0
## [4,]      0
## [5,]      0
## [6,]      0
```

Extract AIC and BIC from the fit of glm

```
glmnet_cv_aic_hmd(fit, lchoice = "min")
```

```
## $AICc
## [1] -6751.432
##
## $BIC
## [1] -6525.138
```

1SE

```
fit <- glmnet_cv_fit_hmd(insurance_train, target="outcome", family="binomial")
h <- glmnet_cv_predict_hmd(fit,
                           data = insurance_test,
                           target = "outcome",
                           lchoice = "1se",
                           type = "binomial")

coef = h$coef
pred_glm_cv_1se <- h$predictions
pred_proba_glm_cv_1se <- h$predict_proba
```

```
summary(fit)
```

```
##          Length Class  Mode
## lambda      74    -none- numeric
## cvm          74    -none- numeric
## cvsd         74    -none- numeric
## cvup         74    -none- numeric
## cvlo         74    -none- numeric
## nzero        74    -none- numeric
## call         4     -none- call
## name         1     -none- character
## glmnet.fit  13     lognet list
## lambda.min   1     -none- numeric
## lambda.1se   1     -none- numeric
## index        2     -none- numeric
```

```
coef
```

```
## 33 x 1 sparse Matrix of class "dgCMatrix"
##                                s1
## (Intercept)                   1.018119e+00
## age.16.25                     2.181744e-01
```

```
## age.26.39 .
## age.40.64 -1.022830e-01
## age.65. -7.228297e-04
## gender.female -7.039896e-01
## gender.male 9.213086e-06
## race.majority .
## race.minority .
## driving_experience.0.9y 1.512906e+00
## driving_experience.10.19y .
## driving_experience.20.29y -1.158297e+00
## driving_experience.30y. -1.306864e+00
## education.high.school .
## education.none 1.261959e-01
## education.university .
## income.middle.class .
## income.poverty 5.757163e-02
## income.upper.class -4.045528e-02
## income.working.class 1.615831e-01
## credit_score -1.235961e-01
## vehicle_ownership -1.338798e+00
## vehicle_year.after.2015 -1.390326e+00
## vehicle_year.before.2015 4.251933e-12
## married -3.962552e-01
## children -2.364787e-01
## postal_code 1.231771e+00
## annual_mileage 5.689875e-01
## vehicle_type.sedan .
## vehicle_type.sports.car .
## speeding_violations .
## dui .
## past_accidents -1.830717e+00
```

```
head(pred_glm_cv_1se)
```

```
##      lambda.1se
## [1,]          0
## [2,]          0
## [3,]          0
## [4,]          0
## [5,]          0
## [6,]          0
```

Extract AIC and BIC from the fit of glm

```
glmnet_cv_aic_hmd(fit, lchoice = "1se")
```

```
## $AICc
## [1] -6634.007
##
## $BIC
## [1] -6480.673
```

Evaluate

GLMNET

```
eval <- eval_hmd(actual, pred_glm)
```

```
print("confusion matrix: ")
```

```
## [1] "confusion matrix: "
```

```
print(eval$confusion_matrix)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction    0    1
```

```
##           0 1186  139
```

```
##           1  197  478
```

```
##
```

```
##           Accuracy : 0.832
```

```
##           95% CI : (0.8149, 0.8481)
```

```
## No Information Rate : 0.6915
```

```
## P-Value [Acc > NIR] : < 2.2e-16
```

```
##
```

```
##           Kappa : 0.6162
```

```
##
```

```
## Mcnemar's Test P-Value : 0.001873
```

```
##
```

```
##           Sensitivity : 0.8576
```

```
##           Specificity : 0.7747
```

```
## Pos Pred Value : 0.8951
```

```
## Neg Pred Value : 0.7081
```

```
## Prevalence : 0.6915
```

```
## Detection Rate : 0.5930
```

```
## Detection Prevalence : 0.6625
```

```
## Balanced Accuracy : 0.8161
```

```
##
```

```
## 'Positive' Class : 0
```

```
##
```

```
print("accuracy: ")
```

```
## [1] "accuracy: "
```

```
print(eval$accuracy)
```

```
## [1] 0.832
```

```
print("precision: ")
```

```
## [1] "precision: "
```

```

print(eval$precision)

## [1] 0.7747164

print("recall: ")

## [1] "recall: "

print(eval$recall)

## [1] 0.7081481

print("f1_score")

## [1] "f1_score"

print(eval$f1_score)

## [1] 1

print("fbeta_score")

## [1] "fbeta_score"

print(eval$fbeta_score)

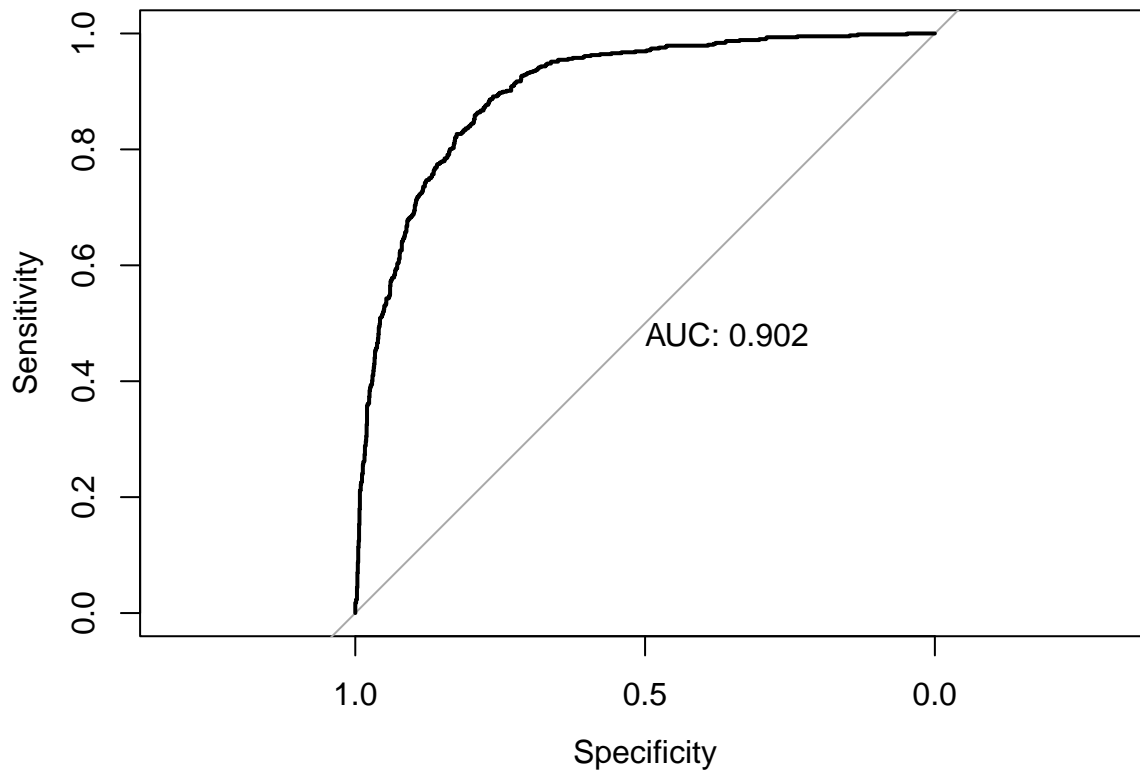
## [1] 0.7399381

pROC::roc(actual ~ pred_proba_glm, plot = TRUE, print.auc = TRUE)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

```



```
##
## Call:
## roc.formula(formula = actual ~ pred_proba_glm, plot = TRUE, print.auc = TRUE)
##
## Data: pred_proba_glm in 1383 controls (actual 0) < 617 cases (actual 1).
## Area under the curve: 0.9017
```

Plot Confusion Matrix

```
eval$confusion_matrix_plot
```

```
## Warning: Use of 'plt$Prediction' is discouraged.
## i Use 'Prediction' instead.
```

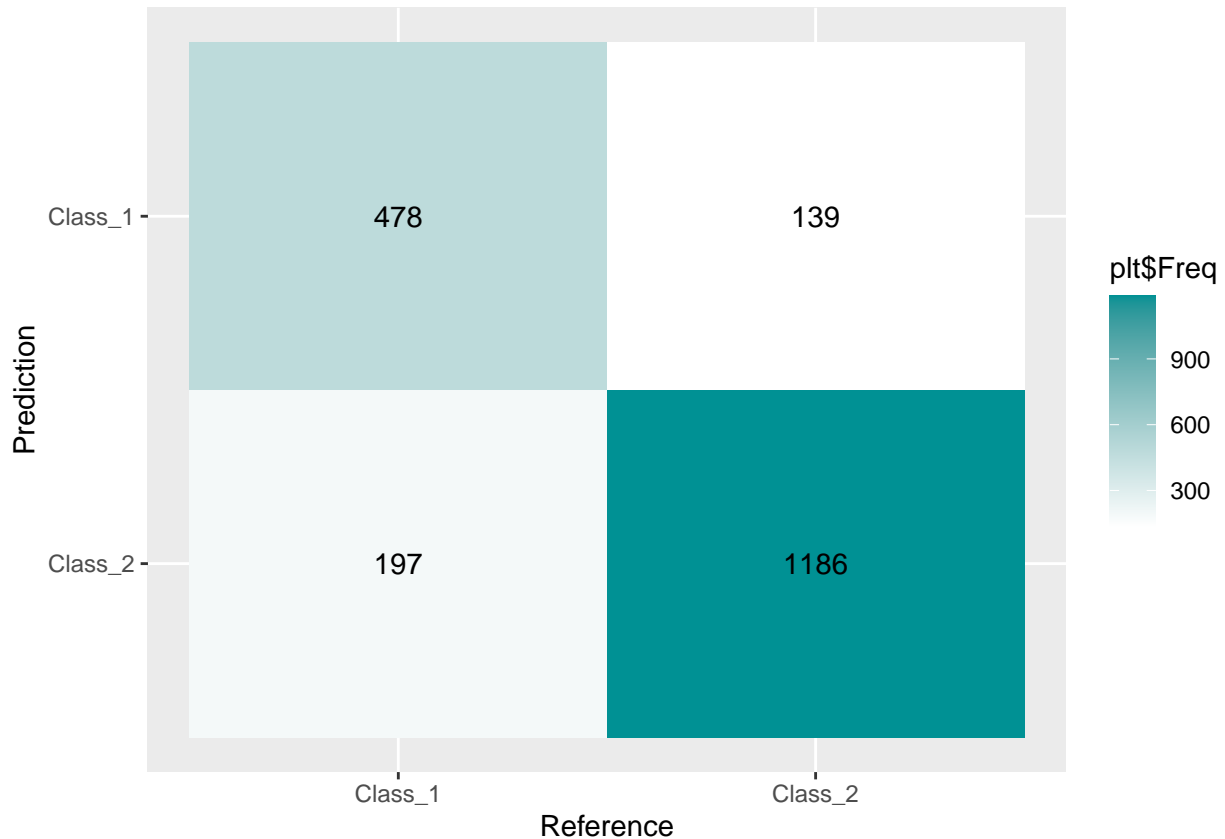
```
## Warning: Use of 'plt$Reference' is discouraged.
## i Use 'Reference' instead.
```

```
## Warning: Use of 'plt$Freq' is discouraged.
## i Use 'Freq' instead.
## Use of 'plt$Freq' is discouraged.
## i Use 'Freq' instead.
```

```
## Warning: Use of 'plt$Prediction' is discouraged.
## i Use 'Prediction' instead.
```

```
## Warning: Use of 'plt$Reference' is discouraged.
## i Use 'Reference' instead.
```

```
## Warning: Use of 'plt$Freq' is discouraged.
## i Use 'Freq' instead.
```



GLMNET CV

min

Confusion Matrix

```
eval <- eval_hmd(actual, pred_glm_cv_min)
```

```
print("confusion matrix: ")
```

```
## [1] "confusion matrix: "
```

```
print(eval$confusion_matrix)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction    0    1
```

```
##           0 1184  140
```

```
##           1  199  477
```

```
##
```

```
##           Accuracy : 0.8305
```

```
##           95% CI : (0.8133, 0.8467)
```

```
##      No Information Rate : 0.6915
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.613
##
##  Mcnemar's Test P-Value : 0.001632
##
##      Sensitivity : 0.8561
##      Specificity : 0.7731
##      Pos Pred Value : 0.8943
##      Neg Pred Value : 0.7056
##      Prevalence : 0.6915
##      Detection Rate : 0.5920
##      Detection Prevalence : 0.6620
##      Balanced Accuracy : 0.8146
##
##      'Positive' Class : 0
##
```

```
print("accuracy: ")
```

```
## [1] "accuracy: "
```

```
print(eval$accuracy)
```

```
## [1] 0.8305
```

```
print("precision: ")
```

```
## [1] "precision: "
```

```
print(eval$precision)
```

```
## [1] 0.7730956
```

```
print("recall: ")
```

```
## [1] "recall: "
```

```
print(eval$recall)
```

```
## [1] 0.7056213
```

```
print("f1_score")
```

```
## [1] "f1_score"
```

```
print(eval$f1_score)
```

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

```
print("fbeta_score")
```

```
## [1] "fbeta_score"
```

```
print(eval$fbeta_score)
```

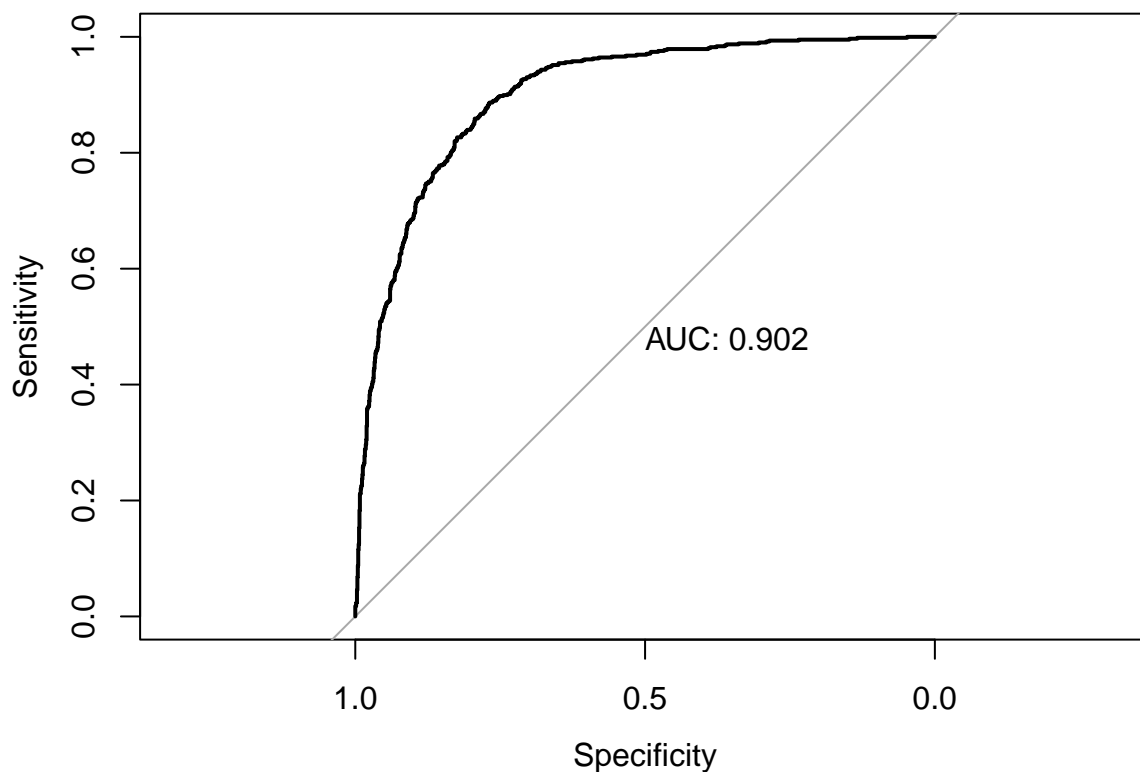
```
## [1] 0.737819
```

```
pROC::roc(actual ~ pred_proba_glm_cv_min, plot = TRUE, print.auc = TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(response, predictors[, 1], ...): Deprecated use a matrix  
## as predictor. Unexpected results may be produced, please pass a numeric vector.
```

```
## Setting direction: controls < cases
```



```
##
```

```
## Call:
```

```
## roc.formula(formula = actual ~ pred_proba_glm_cv_min, plot = TRUE,      print.auc = TRUE)
```

```
##
```

```
## Data: pred_proba_glm_cv_min in 1383 controls (actual 0) < 617 cases (actual 1).
```

```
## Area under the curve: 0.9017
```


Plot Confusion Matrix

```
eval$confusion_matrix_plot
```

```
## Warning: Use of 'plt$Prediction' is discouraged.  
## i Use 'Prediction' instead.
```

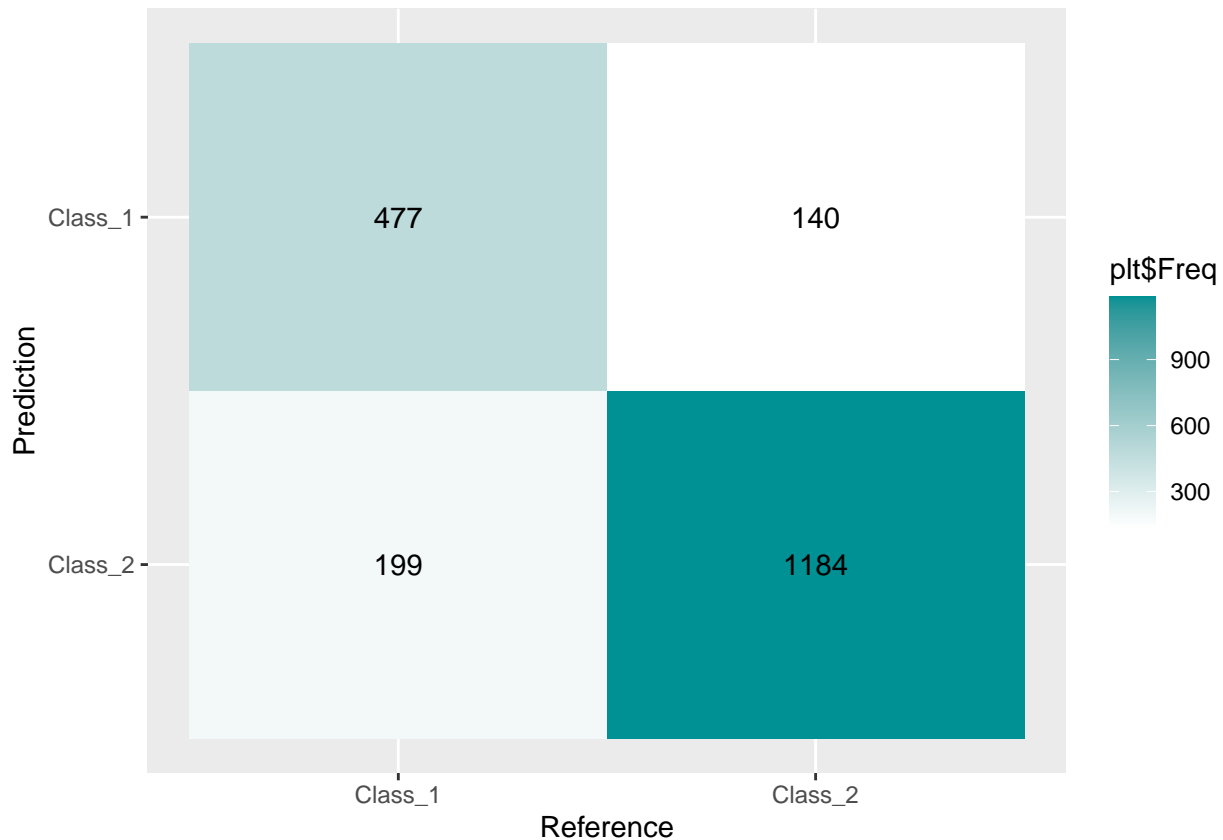
```
## Warning: Use of 'plt$Reference' is discouraged.  
## i Use 'Reference' instead.
```

```
## Warning: Use of 'plt$Freq' is discouraged.  
## i Use 'Freq' instead.  
## Use of 'plt$Freq' is discouraged.  
## i Use 'Freq' instead.
```

```
## Warning: Use of 'plt$Prediction' is discouraged.  
## i Use 'Prediction' instead.
```

```
## Warning: Use of 'plt$Reference' is discouraged.  
## i Use 'Reference' instead.
```

```
## Warning: Use of 'plt$Freq' is discouraged.  
## i Use 'Freq' instead.
```



lse

```
eval <- eval_hmd(actual, pred_glm_cv_1se)
```

```
print("confusion matrix: ")
```

```
## [1] "confusion matrix: "
```

```
print(eval$confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 1193  145
##           1  190  472
##
##           Accuracy : 0.8325
##           95% CI : (0.8154, 0.8486)
##           No Information Rate : 0.6915
##           P-Value [Acc > NIR] : < 2e-16
##
##           Kappa : 0.6152
##
##  Mcnemar's Test P-Value : 0.01622
##
##           Sensitivity : 0.8626
##           Specificity : 0.7650
##           Pos Pred Value : 0.8916
##           Neg Pred Value : 0.7130
##           Prevalence : 0.6915
##           Detection Rate : 0.5965
##           Detection Prevalence : 0.6690
##           Balanced Accuracy : 0.8138
##
##           'Positive' Class : 0
##
```

```
print("accuracy: ")
```

```
## [1] "accuracy: "
```

```
print(eval$accuracy)
```

```
## [1] 0.8325
```

```
print("precision: ")
```

```
## [1] "precision: "
```

```

print(eval$precision)

## [1] 0.7649919

print("recall: ")

## [1] "recall: "

print(eval$recall)

## [1] 0.7129909

print("f1_score")

## [1] "f1_score"

print(eval$f1_score)

## [1] 1

print("fbeta_score")

## [1] "fbeta_score"

print(eval$fbeta_score)

## [1] 0.7380766

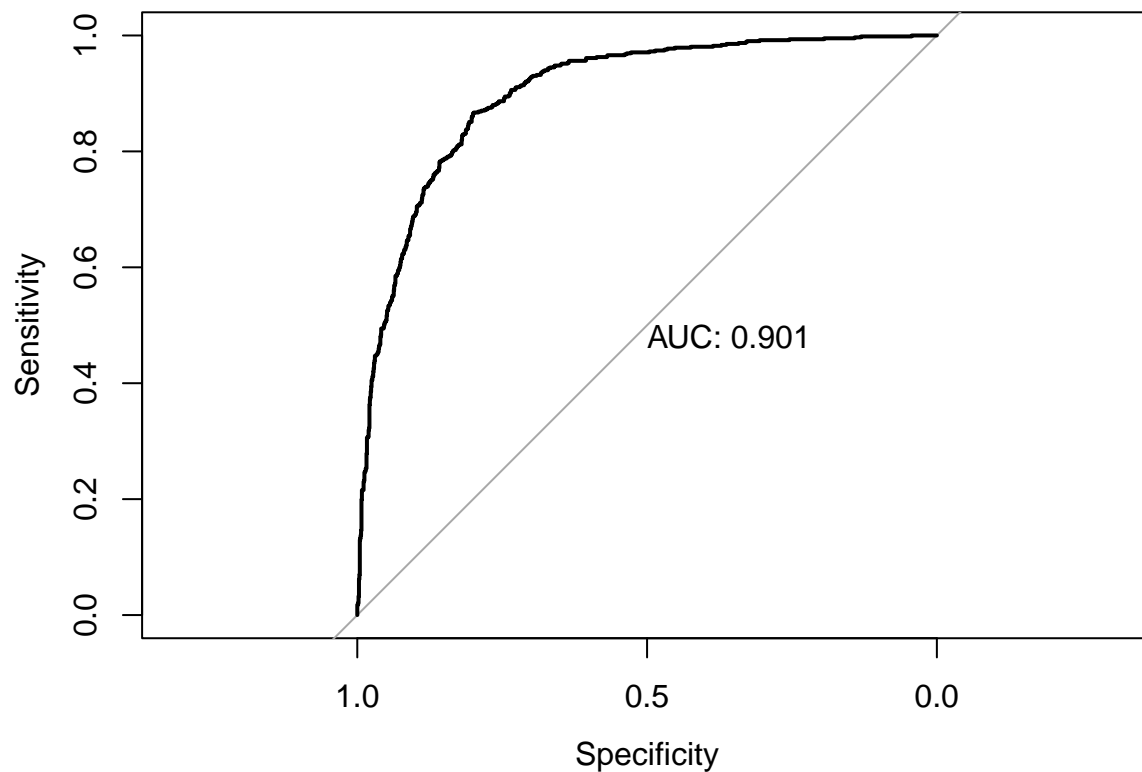
pROC::roc(actual ~ pred_proba_glm_cv_1se, plot = TRUE, print.auc = TRUE)

## Setting levels: control = 0, case = 1

## Warning in roc.default(response, predictors[, 1], ...): Deprecated use a matrix
## as predictor. Unexpected results may be produced, please pass a numeric vector.

## Setting direction: controls < cases

```



```
##
## Call:
## roc.formula(formula = actual ~ pred_proba_glm_cv_1se, plot = TRUE,      print.auc = TRUE)
##
## Data: pred_proba_glm_cv_1se in 1383 controls (actual 0) < 617 cases (actual 1).
## Area under the curve: 0.9012
```

Random Forest

```
actual <- aiinsurance::insurance_test$outcome
X_train <- aiinsurance::insurance_train[, colnames(aiinsurance::insurance_train)[colnames(aiinsurance::insurance_train) != "outcome"]]
X_test <- aiinsurance::insurance_test[, colnames(aiinsurance::insurance_test)[colnames(aiinsurance::insurance_test) != "outcome"]]
y_train <- as.factor(aiinsurance::insurance_train[["outcome"]])
y_test <- as.factor(aiinsurance::insurance_test[["outcome"]])
```

```
rf <- rf_fit_hmd(insurance_train,
                 ntree = 300,
                 mtry = 10,
                 proximity = TRUE,
                 importance = TRUE)
```

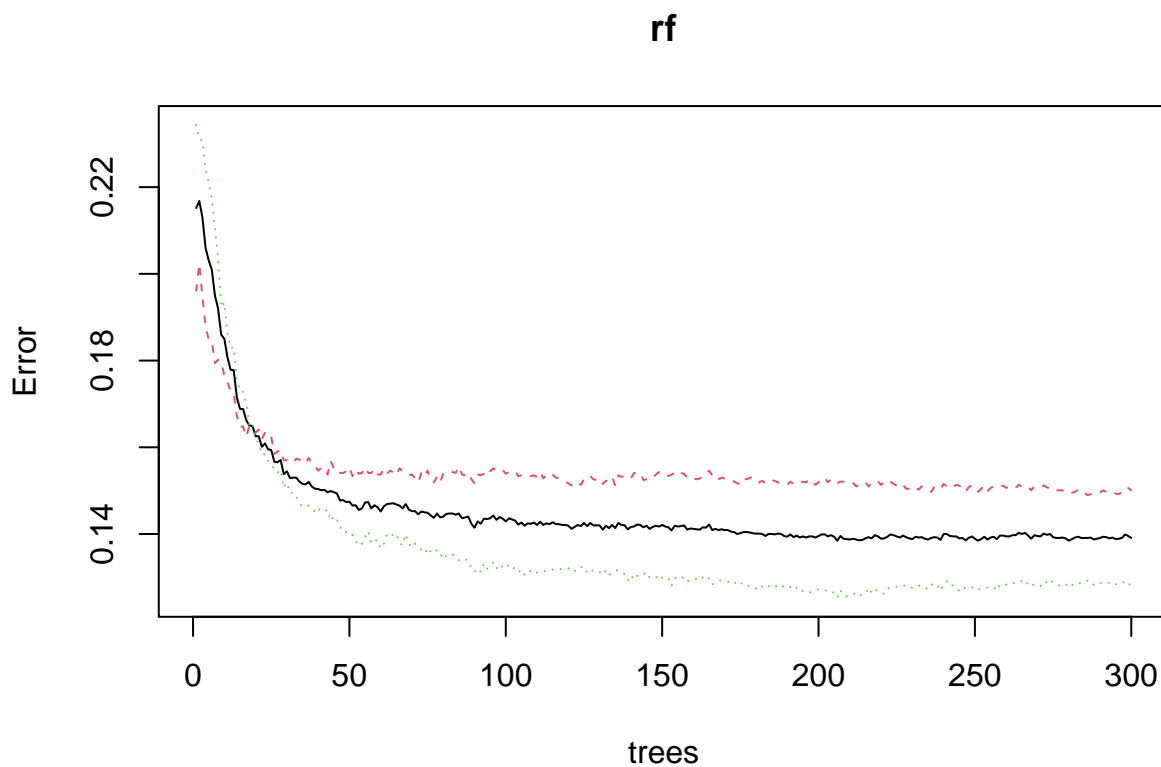
```
print(rf)
```

```
##
## Call:
```

```
## randomForest(x = X_train, y = y_train, ntree = {      {      ntree      } }, mtry = {      {
##              Type of random forest: classification
##              Number of trees: 300
## No. of variables tried at each split: 10
##
##              OOB estimate of  error rate: 13.91%
## Confusion matrix:
##      0      1 class.error
## 0 4661  823  0.1500729
## 1  707 4809  0.1281726
```

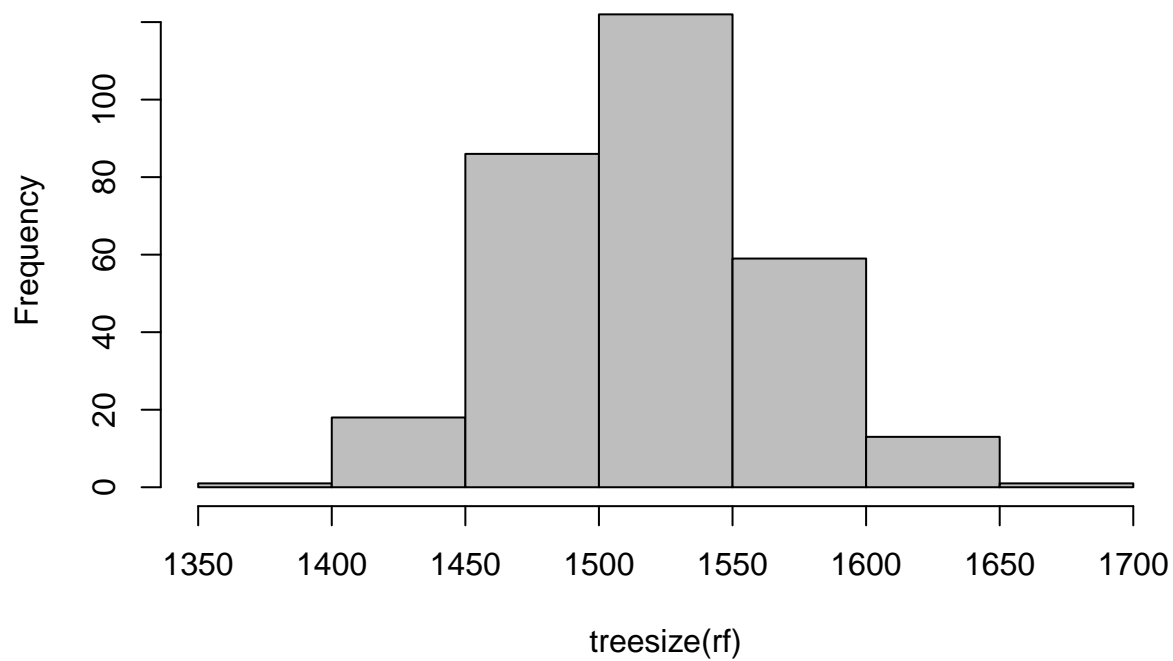
```
predict_rf <- rf_predict_hmd(data=insurance_test, fit=rf)
```

```
plot(rf)
```



```
# Number of nodes for the trees
hist(treesize(rf), # give us the number of trees in term of number of nodes
      main = "Number of Nodes for the Trees",
      col = "grey")
```

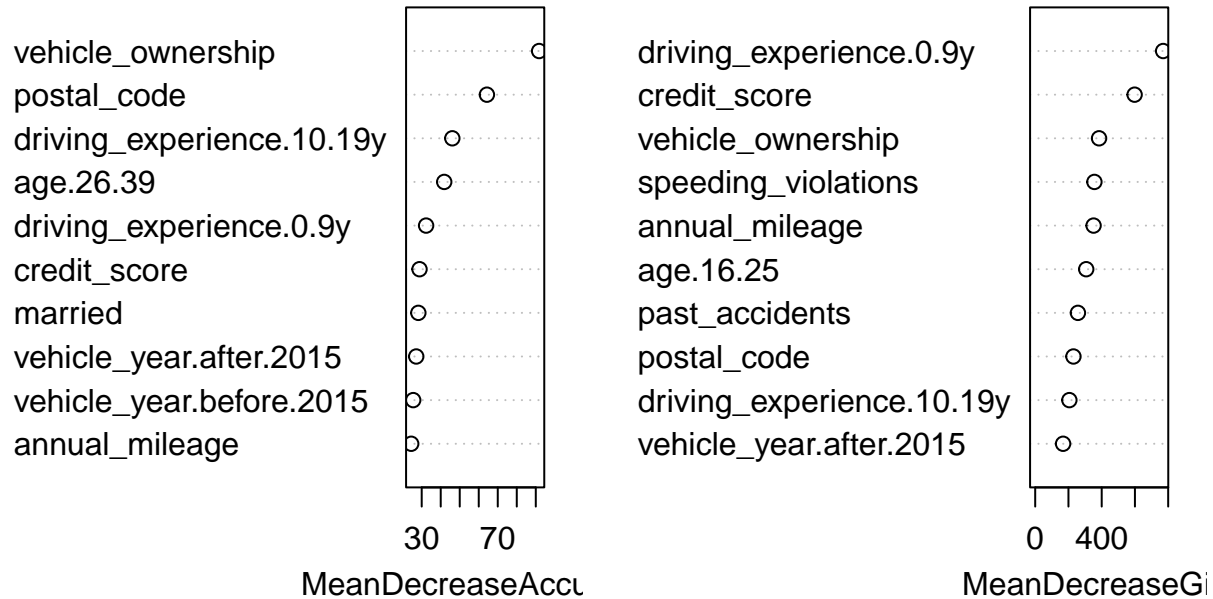
Number of Nodes for the Trees



Variable Importance

```
varImpPlot(rf,  
            sort = T,  
            n.var = 10,  
            main = "Top 10 Variable Importance")
```

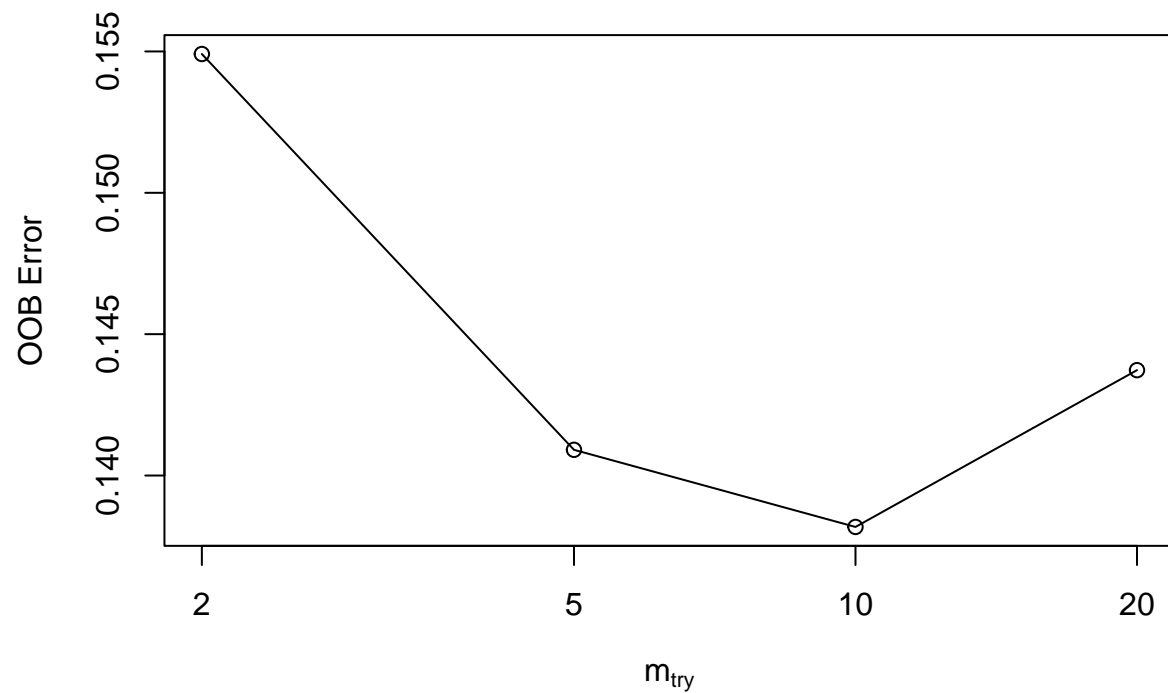
Top 10 Variable Importance



Hyperparameter Tuning

```
t <- tuneRF(X_train,
            y_train,
            stepFactor = 0.5,
            plot = TRUE,
            ntreeTry = 300,
            trace = TRUE,
            improve = 0.01)
```

```
## mtry = 5   OOB error = 14.09%
## Searching left ...
## mtry = 10   OOB error = 13.82%
## 0.01935484 0.01
## mtry = 20   OOB error = 14.37%
## -0.04013158 0.01
## Searching right ...
## mtry = 2     OOB error = 15.49%
## -0.1210526 0.01
```



```
print(t)
```

```
##      mtry  OOBError
## 2.00B    2 0.1549091
## 5.00B    5 0.1409091
## 10.00B   10 0.1381818
## 20.00B   20 0.1437273
```

Evaluate

```
eval <- eval_hmd(actual, predict_rf$predictions_num)
```

```
print("confusion matrix: ")
```

```
## [1] "confusion matrix: "
```

```
print(eval$confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##      Reference
## Prediction    0    1
##      0 1172  123
##      1  211  494
##
##      Accuracy : 0.833
##      95% CI : (0.8159, 0.8491)
```



```
##      No Information Rate : 0.6915
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.6235
##
##  McNemar's Test P-Value : 1.932e-06
##
##      Sensitivity : 0.8474
##      Specificity : 0.8006
##      Pos Pred Value : 0.9050
##      Neg Pred Value : 0.7007
##      Prevalence : 0.6915
##      Detection Rate : 0.5860
##      Detection Prevalence : 0.6475
##      Balanced Accuracy : 0.8240
##
##      'Positive' Class : 0
##
```

```
print("accuracy: ")
```

```
## [1] "accuracy: "
```

```
print(eval$accuracy)
```

```
## [1] 0.833
```

```
print("precision: ")
```

```
## [1] "precision: "
```

```
print(eval$precision)
```

```
## [1] 0.8006483
```

```
print("recall: ")
```

```
## [1] "recall: "
```

```
print(eval$recall)
```

```
## [1] 0.7007092
```

```
print("f1_score")
```

```
## [1] "f1_score"
```

```
print(eval$f1_score)
```

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

```
print("fbeta_score")
```

```
## [1] "fbeta_score"
```

```
print(eval$fbeta_score)
```

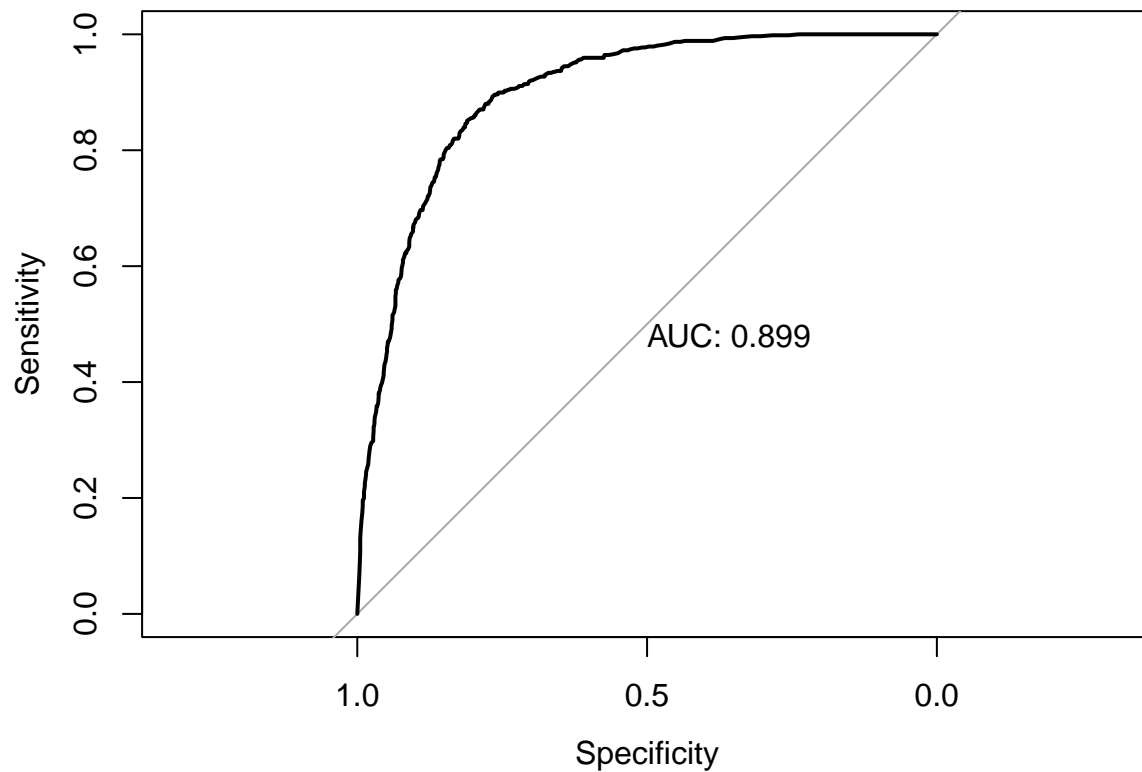
```
## [1] 0.7473525
```

```
pred_proba_rf <- predict_rf$predict_proba
```

```
pROC::roc(actual ~ pred_proba_rf, plot = TRUE, print.auc = TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
##
```

```
## Call:
```

```
## roc.formula(formula = actual ~ pred_proba_rf, plot = TRUE, print.auc = TRUE)
```

```
##
```

```
## Data: pred_proba_rf in 1383 controls (actual 0) < 617 cases (actual 1).
```

```
## Area under the curve: 0.8992
```

Plot Confusion Matrix

```
eval$confusion_matrix_plot
```

```
## Warning: Use of 'plt$Prediction' is discouraged.  
## i Use 'Prediction' instead.
```

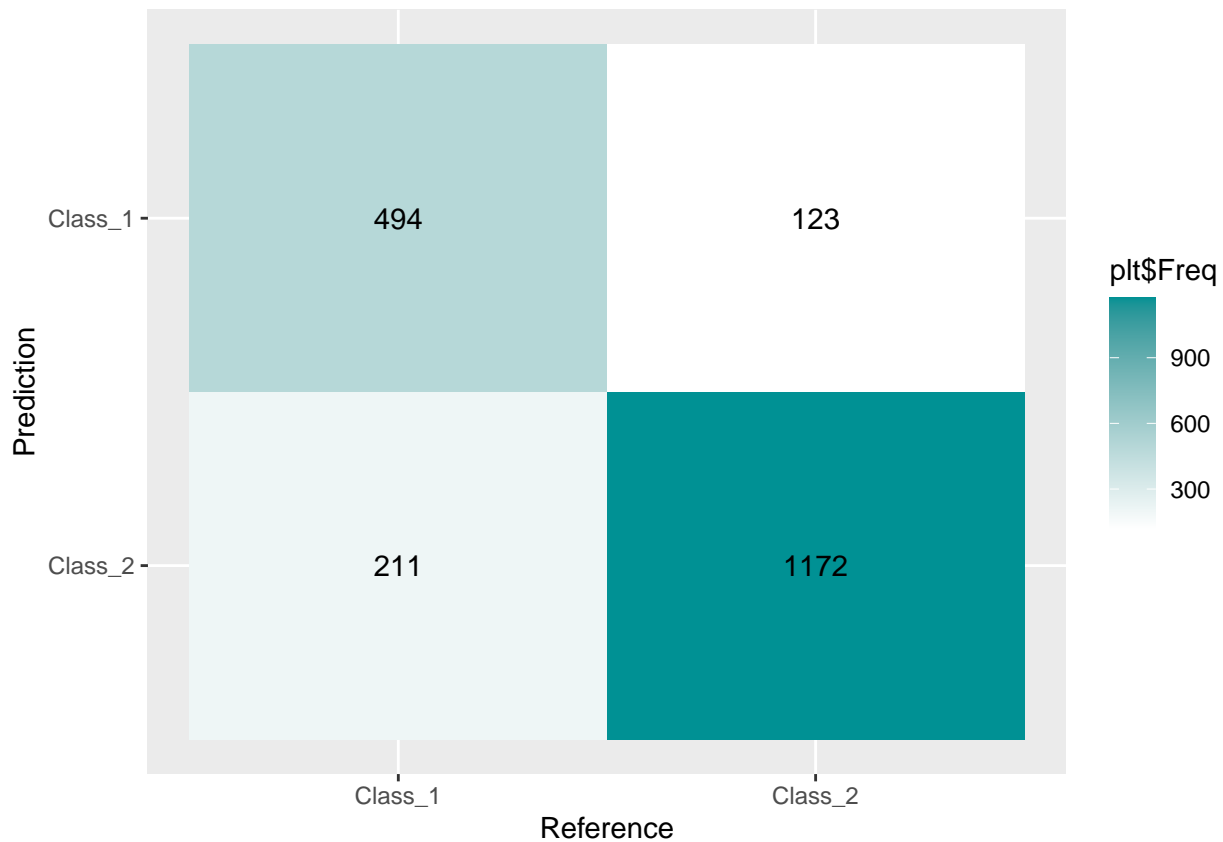
```
## Warning: Use of 'plt$Reference' is discouraged.  
## i Use 'Reference' instead.
```

```
## Warning: Use of 'plt$Freq' is discouraged.  
## i Use 'Freq' instead.  
## Use of 'plt$Freq' is discouraged.  
## i Use 'Freq' instead.
```

```
## Warning: Use of 'plt$Prediction' is discouraged.  
## i Use 'Prediction' instead.
```

```
## Warning: Use of 'plt$Reference' is discouraged.  
## i Use 'Reference' instead.
```

```
## Warning: Use of 'plt$Freq' is discouraged.  
## i Use 'Freq' instead.
```



Extra Material: Decoding One-Hot-Encoding

```
col_list <- names(insurance_test)

cols_list_enc <- grep("\\.", col_list, value = TRUE, perl = TRUE)

pattern <- "([a-zA-Z0-9_-]*)(.)"
common_names <- list()
for (col in cols_list_enc)
{
  common_names <- append(common_names, str_match(col, pattern)[2])
}
common_names <- common_names |> unique()

insurance_test_revert <-
insurance_test[, !(colnames(insurance_test) %in% cols_list_enc)]

for (name in common_names){
  cols_revert <- lapply(cols_list_enc,
    function(x) x[grepl(name, x)])
  cols_revert <- cols_revert[lengths(cols_revert)!=0]
  #cols_revert <- lapply(cols_list_enc,
  #  function(x) str_match(x,"([\\.])\\.") [3])

  cols_revert <- cols_revert |> unlist()

  w <- which(insurance_test[cols_revert]==1, arr.ind = T)

  insurance_test_revert_name <- names(insurance_test[cols_revert])[w[order(w[,1]),2]]

  insurance_test_revert[name] <- insurance_test_revert_name
}
```

```
insurance_test[,0:4] |> tidyr::pivot_longer(col=starts_with("age"), names_to = "age", names_prefix="ag
```

```
## # A tibble: 8,000 x 1
##   age
##   <chr>
## 1 16.25
## 2 26.39
## 3 40.64
## 4 65.
## 5 16.25
## 6 26.39
## 7 40.64
## 8 65.
## 9 16.25
## 10 26.39
```

```
## # ... with 7,990 more rows
```

```
insurance_test_revert |> dplyr::glimpse()
```

```
## Rows: 2,000
## Columns: 18
## $ credit_score      <dbl> 0.6234644, 0.6248241, 0.7454570, 0.6432089, 0.7817~
## $ vehicle_ownership <int> 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,~
## $ married           <int> 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,~
## $ children          <int> 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0,~
## $ postal_code       <dbl> 0.0000000, 0.0000000, 0.2751793, 0.0000000, 0.0000~
## $ annual_mileage    <dbl> 0.55, 0.40, 0.50, 0.30, 0.50, 0.50, 0.50, 0.80, 0.~
## $ speeding_violations <dbl> 0.13636364, 0.27272727, 0.18181818, 0.18181818, 0.~
## $ dui               <dbl> 0.0000000, 0.3333333, 0.0000000, 0.1666667, 0.0000~
## $ past_accidents    <dbl> 0.20000000, 0.46666667, 0.00000000, 0.13333333, 0.~
## $ outcome           <int> 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1,~
## $ age               <chr> "age.40.64", "age.65.", "age.65.", "age.40.64", "a~
## $ gender            <chr> "gender.female", "gender.male", "gender.female", "~
## $ race              <chr> "race.majority", "race.majority", "race.majority",~
## $ driving_experience <chr> "driving_experience.20.29y", "driving_experience.3~
## $ education         <chr> "education.high.school", "education.high.school", ~
## $ income             <chr> "income.upper.class", "income.upper.class", "incom~
## $ vehicle_year      <chr> "vehicle_year.after.2015", "vehicle_year.after.201~
## $ vehicle_type      <chr> "vehicle_type.sedan", "vehicle_type.sedan", "vehic~
```