Harvard Capstone - Choose your own - Health Insurance Cross Sell Prediction

Wesley Tay

5/10/2020

INTRODUCTION

This project is adapted from a challenge on Kaggle (https://www.kaggle.com/anmolkumar/health-insurance-cross-sell-prediction).

The client is an Insurance company that has provided Health Insurance to its customers and is looking to build a model to predict whether the policyholders (customers) from past year will also be interested in Vehicle Insurance provided by the company.

An insurance policy is an arrangement by which a company undertakes to provide a guarantee of compensation for specified loss, damage, illness, or death in return for the payment of a specified premium. A premium is a sum of money that the customer needs to pay regularly to an insurance company for this guarantee.

Just like medical insurance, there is vehicle insurance where every year customer needs to pay a premium of certain amount to insurance provider company so that in case of unfortunate accident by the vehicle, the insurance provider company will provide a compensation (called 'sum assured') to the customer.

Building a model to predict whether a customer would be interested in Vehicle Insurance is extremely helpful for the company because it can then accordingly plan its communication strategy to reach out to those customers and optimise its business model and revenue.

In this project, we aim to build supervised machine learningmodel to predict if a customer would be interested in Vehicle insurance, based on information about demographics (gender, age, region code type), Vehicles (Vehicle Age, Damage), Policy (Premium, sourcing channel) etc.

METHODOLOGY AND ANALYSIS

Libraries Used

The libraries used for this project are as follows:

```r
library(tidyverse)
library(RCurl)
library(summarytools)
library(RColorBrewer)
library(ggplot2)
library(caret)
library(randomForest)
library(corrplot)
library(pROC)
library(knitr)
```

Loading the Data

The original dataset from Kaggle includes both test and training datasets. However, as the files were taken from a kaggle competition, the test dataset did not contain the dependent variable. As such, I split the training dataset into my own testing and training sets so that the resulting prediction of the machine learning model could be displayed.

The total number of observations in the set was 381,109, which was quite big and difficult for my computer to process. I reduced the total dataset to 150000 and split the dataset 80:20 such that caret training models could be run effectively.

The data has been uploaded onto github and is accessible with the following code.

```r
data_source <- getURL("https://raw.githubusercontent.com/birddropping/Harvard-Capstone-Health-Insurance
data <- read.csv(text = data_source)

data <- data[1:150000, ]
```

The dataset contains the following features and dependent variable:

Features

Id - Unique ID for the customer

Gender - Gender of the customer

Age - Age of the customer

Driving_License - 0: Customer does not have driving license; 1: Customer already has driving license

Region_Code - Unique code for the region of the customer

Previously_Insured - 1: Customer already has Vehicle Insurance; 0: Customer doesn't have Vehicle Insurance

Vehicle_Age - Age of the vehicle

Vehicle_Damage - 1: Customer got his/her vehicle damaged in the past; 0: Customer didn't get his/her vehicle damaged in the past

Annual_Premium - The amount the customer needs to pay as premium in the year

Policy_Sales_Channel - Anonymized code for the channel of outreaching to the customer e.g. Different agents, over mail, over phone, in person, etc.

Vintage - Number of days that the customer has been associated with the company

Dependent Variable

Response - 1: Customer is interested; 0: Customer is not interested

The first few lines of the dataset is shown below:

```
head(data)
```

```
##   id Gender Age Driving_License Region_Code Previously_Insured Vehicle_Age
## 1  1   Male  44               1          28                  0   > 2 Years
## 2  2   Male  76               1           3                  0    1-2 Year
## 3  3   Male  47               1          28                  0   > 2 Years
## 4  4   Male  21               1          11                  1    < 1 Year
## 5  5 Female  29               1          41                  1    < 1 Year
## 6  6 Female  24               1          33                  0    < 1 Year
##   Vehicle_Damage Annual_Premium Policy_Sales_Channel Vintage Response
## 1            Yes          40454                   26     217        1
## 2             No          33536                   26     183        0
## 3            Yes          38294                   26      27        1
## 4             No          28619                  152     203        0
## 5             No          27496                  152      39        0
## 6            Yes           2630                  160     176        0
```

Data Cleaning

The loaded data was codified, categorical data except for the response was converted into integers. Responses were converted from binary numerals 0 and 1 to “N” and “Y” factors because the caret package was not able to perform training on the “0” integer.

```
# Converting all inputs to numeric values

# Gender #
data$Gender[data$Gender == "Male"] <- 1
data$Gender[data$Gender == "Female"] <- 2

# Vehicle Age #
data$Vehicle_Age[data$Vehicle_Age == "< 1 Year"] <- 1
data$Vehicle_Age[data$Vehicle_Age == "1-2 Year"] <- 2
data$Vehicle_Age[data$Vehicle_Age == "> 2 Years"] <- 3

# Vehicle Damage #
data$Vehicle_Damage[data$Vehicle_Damage == "No"] <- 0
data$Vehicle_Damage[data$Vehicle_Damage == "Yes"] <- 1

# Converting Responses #
data$Response <- as.numeric(data$Response)
```

```r
data$Response[data$Response == 0] <- "N"
data$Response[data$Response == 1] <- "Y"

# Converting all columns to factor type

data <- lapply(data, as.factor)
data <- as.data.frame(data)

# Change Age, Annual Premiums and Vintage to numeric as they
# are continuous data

data$Age <- as.numeric(data$Age)
data$Annual_Premium <- as.numeric(data$Annual_Premium)
data$Region_Code <- as.numeric(data$Region_Code)
data$Policy_Sales_Channel <- as.numeric(data$Policy_Sales_Channel)
data$Vintage <- as.numeric(data$Vintage)
```

Data Partitioning

Once the dataset was converted into respective factors and numerical data, the data was partitioned out into a training set (edx), as well as a validation set (validation). The validation set would not be used until the final testing of the machine learning model.

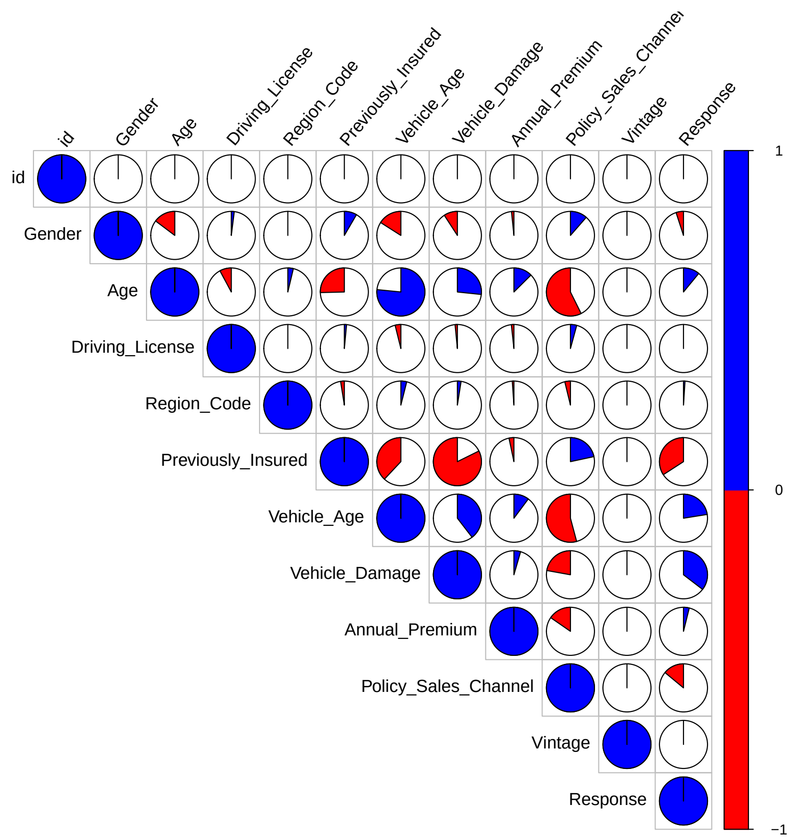
To test the model during the developmental phase, the training set was partitioned a second time into another training and test set: edx_train and edx_test.

All sets were semi-joined with the training data such that none of the features captured as part of the machine learning model would be out of range.

```r
set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(data$Response, times = 1, p = 0.2,
    list = FALSE)

validation <- data[test_index, ]
edx <- data[-test_index, ]

validation <- validation %>% semi_join(edx, by = "Gender") %>%
    semi_join(edx, by = "Driving_License") %>% semi_join(edx,
    by = "Region_Code") %>% semi_join(edx, by = "Previously_Insured") %>%
    semi_join(edx, by = "Vehicle_Damage") %>% semi_join(edx,
    by = "Policy_Sales_Channel")


set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(edx$Response, times = 1, p = 0.2,
    list = FALSE)

edx_test <- edx[test_index, ]

edx_train <- edx[-test_index, ]

edx_test <- edx_test %>% semi_join(edx_train, by = "Gender") %>%
    semi_join(edx_train, by = "Driving_License") %>% semi_join(edx_train,
    by = "Region_Code") %>% semi_join(edx_train, by = "Previously_Insured") %>%
    semi_join(edx_train, by = "Vehicle_Damage") %>% semi_join(edx_train,
    by = "Policy_Sales_Channel")

rm(data_source, test_index, data)
```

Data Exploration

Correlation Plot



From this correlation plot, we see that Gender, Age and the Policy Sales Channel used to attract the customer has a mild correlation with the response of the customer, and Vehicle Age, previous Vehicle Damage and whether they have been previously insured has a moderate correlation with the response. We will examine these factors in the next few steps.

Overall proportion of people taking up insurance

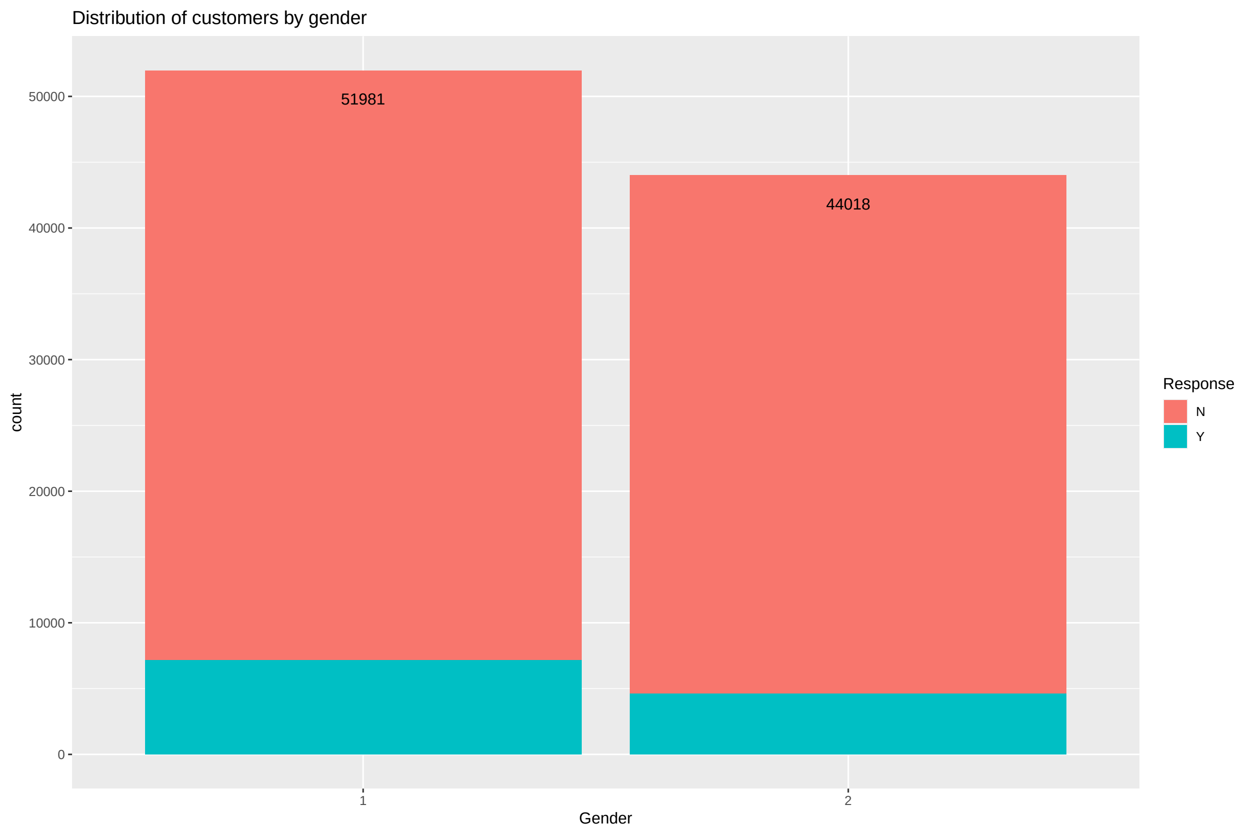
To begin, the proportion of customers who eventually take up the vehicle insurance is assessed.

```
## # A tibble: 2 x 2
## # Groups:   Response [2]
##   Response     n
##   <fct>    <int>
## 1 N        84227
## 2 Y        11772
```

We see that only 13.9% of customers were eventually took up the vehicle insurance, which suggests that the data is imbalanced.

Gender

To see the differences in the conversions between genders, we plot a bar chart of the 2 genders, and highlight the number of customers who eventually responded positively.



```
## ‘summarise()‘ regrouping output by ’Gender’ (override with ‘.groups‘ argument)
```

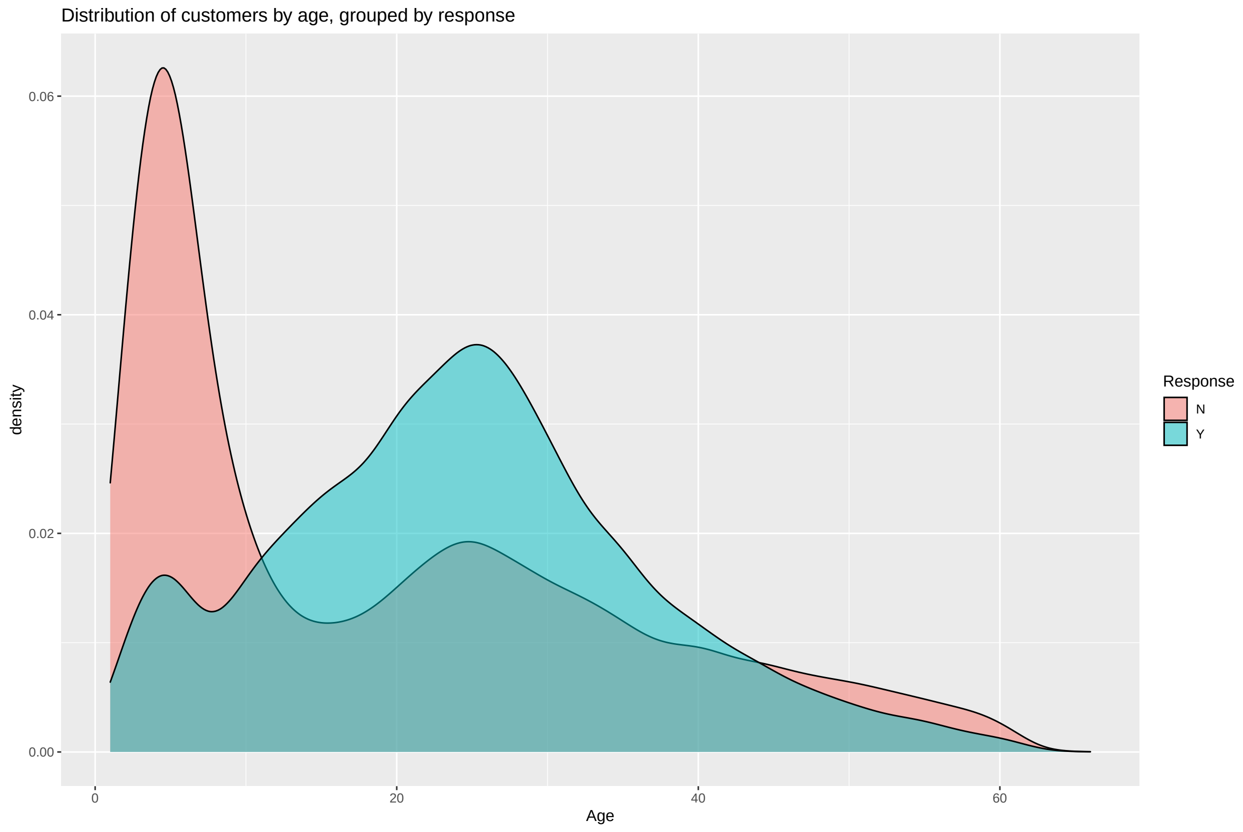
```
## # A tibble: 4 x 4
## # Groups:   Gender [2]
##   Gender Response     n percentage
##   <fct>  <fct>    <int> <chr>
## 1 1      N        44822 86%
## 2 1      Y         7159 14%
## 3 2      N        39405 90%
## 4 2      Y         4613 10%
```

As can be seen in the graph, as well as from the percentage summary, males seem to be slightly more inclined to take up the insurance than females.

Age

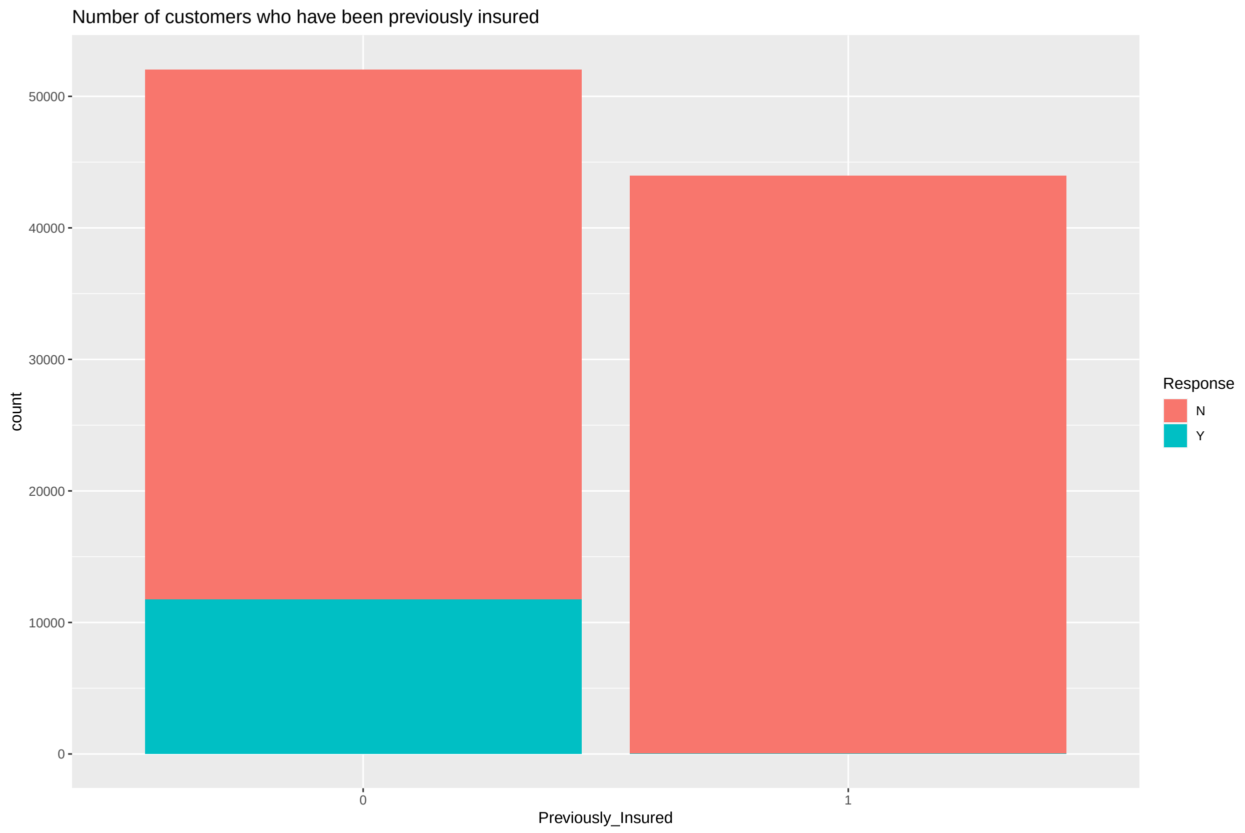
Age may also play a factor in the decision to take up vehicle insurance. The case does not make it clear, and as there are some customers who are of a very young age holding on to a driving license, I make the assumption that age in this case indicates the driver's age starting from the legal age for driving (e.g. 1 = 19 assuming the legal age to drive is 18).

We plot a density plot of the various ages and their response to the offer to take up vehicle insurance:



As we can see, there indeed is a distinction between younger and older drivers. younger drivers tend to not respond favourably, whereas older drivers tend to be more receptive to taking up vehicle insurance.

Previous Insurance



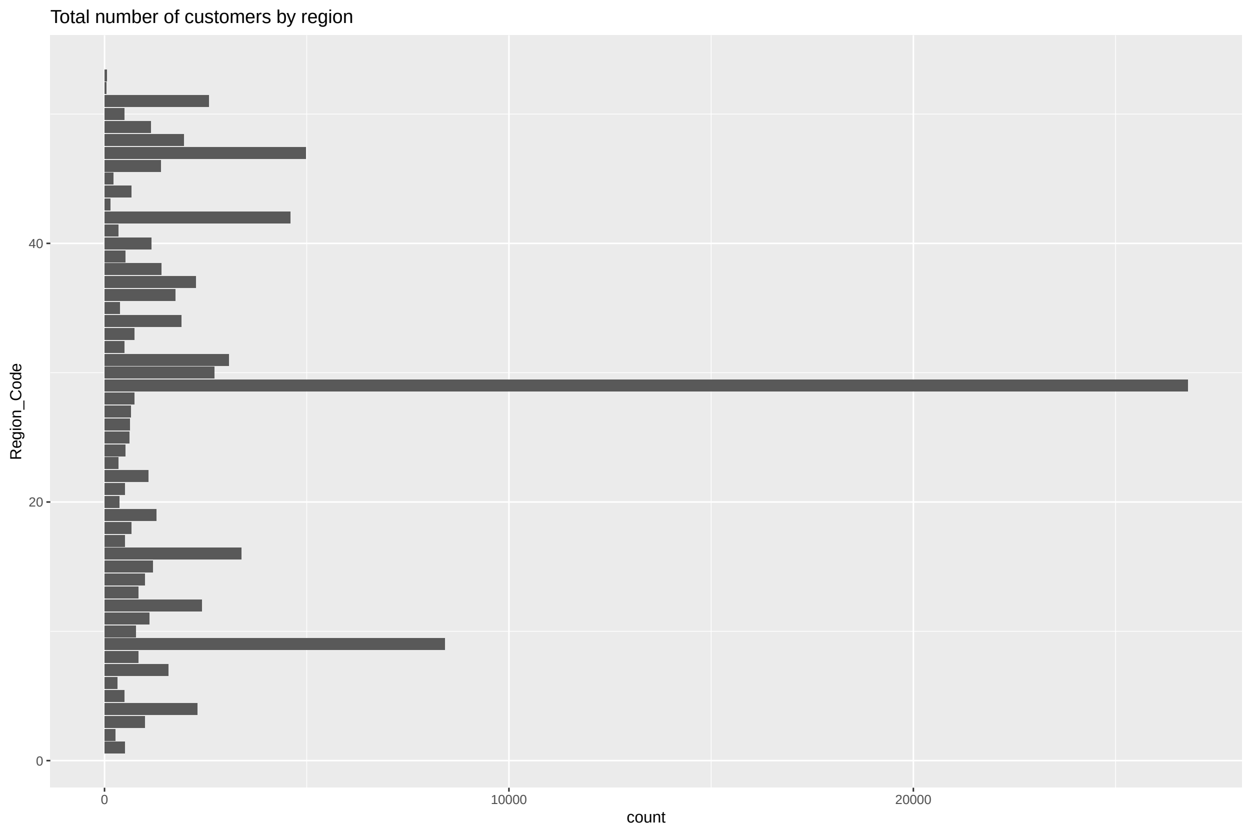
```
## `summarise()` regrouping output by 'Previously_Insured' (override with `.groups` argument)
```

```
## # A tibble: 4 x 4
## # Groups:   Previously_Insured [2]
##   Previously_Insured Response     n percentage
##   <fct>              <fct>    <int> <chr>
## 1 0                  N        40304 77%
## 2 0                  Y        11736 23%
## 3 1                  N        43923 100%
## 4 1                  Y           36 0%
```

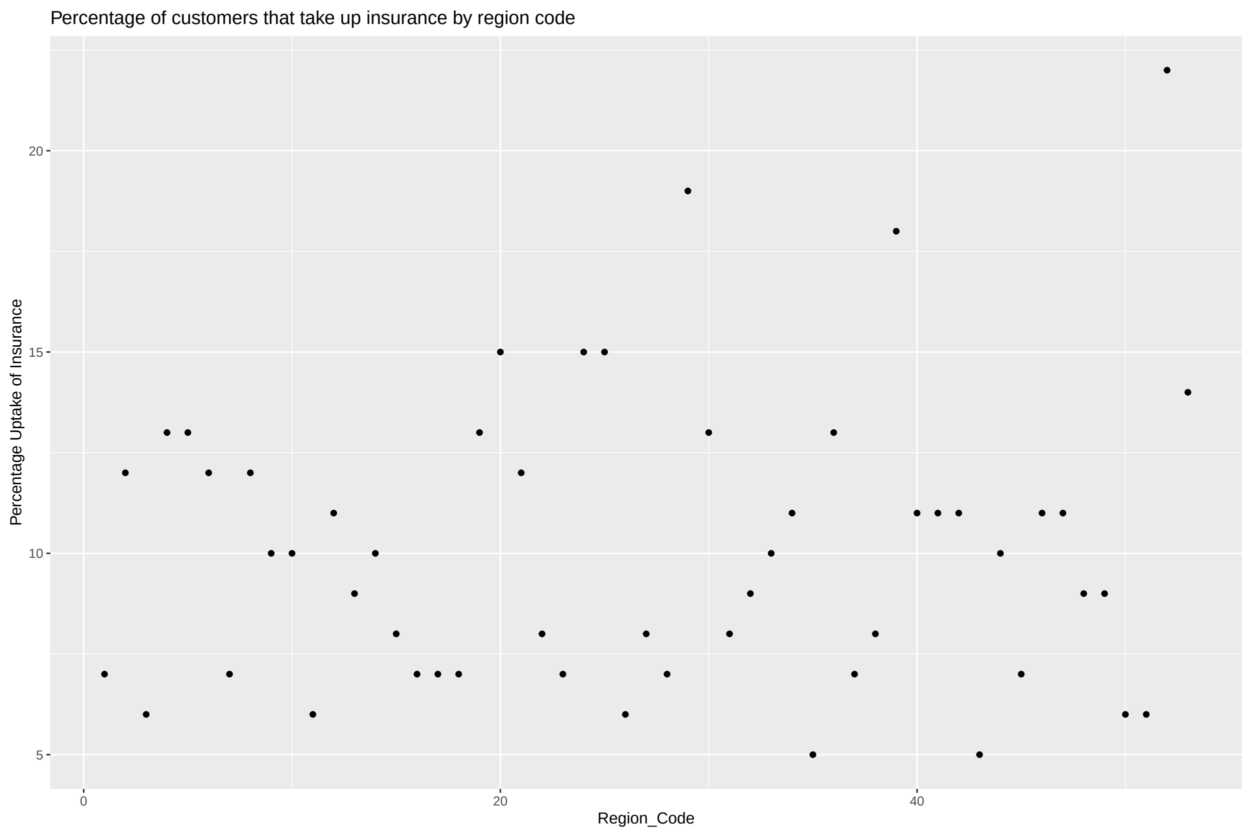
The aspect of being previously insured seems to be an important deciding factor in determining if someone will take up the vehicle insurance. From this plot, we see that most people who have been previously insured chose not to take up the insurance, with only 36 out of 43959 individuals chose to take up the insurance. However, on the other hand, 23% of people who were not previously insured chose to take up the insurance. This may be the case because people who were previously insured would probably have existing plans still in place and would not require additional vehicle insurance.

Regions

We explore the distribution of the customers across the various regions.



```
## ‘summarise()‘ regrouping output by ’Region_Code’ (override with ‘.groups‘ argument)
```



We can see from these plots that that there is a huge variation in the number of customers hailing from each of the individual regions. There is also a difference in percentage of customers taking up the vehicle insurance by region. However, given that most regions have very few customers, percentage uptake rates may not be very accurate. To understand the extent of variation between region codes, we compare the two regions with the minimum and maximum number of customers.

```
## ‘summarise()‘ regrouping output by ’Region_Code’ (override with ‘.groups‘ argument)
```

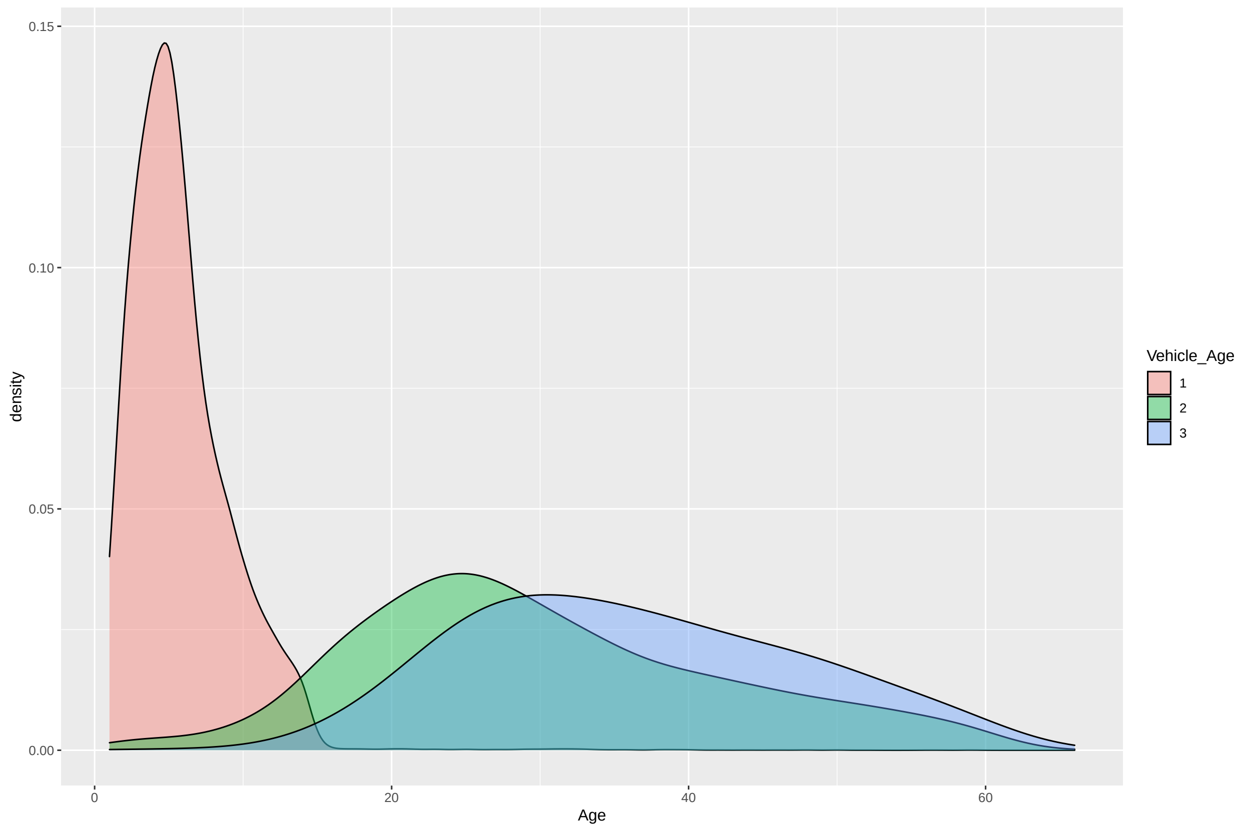
```
## [1] 21690
```

```
## [1] 8
```

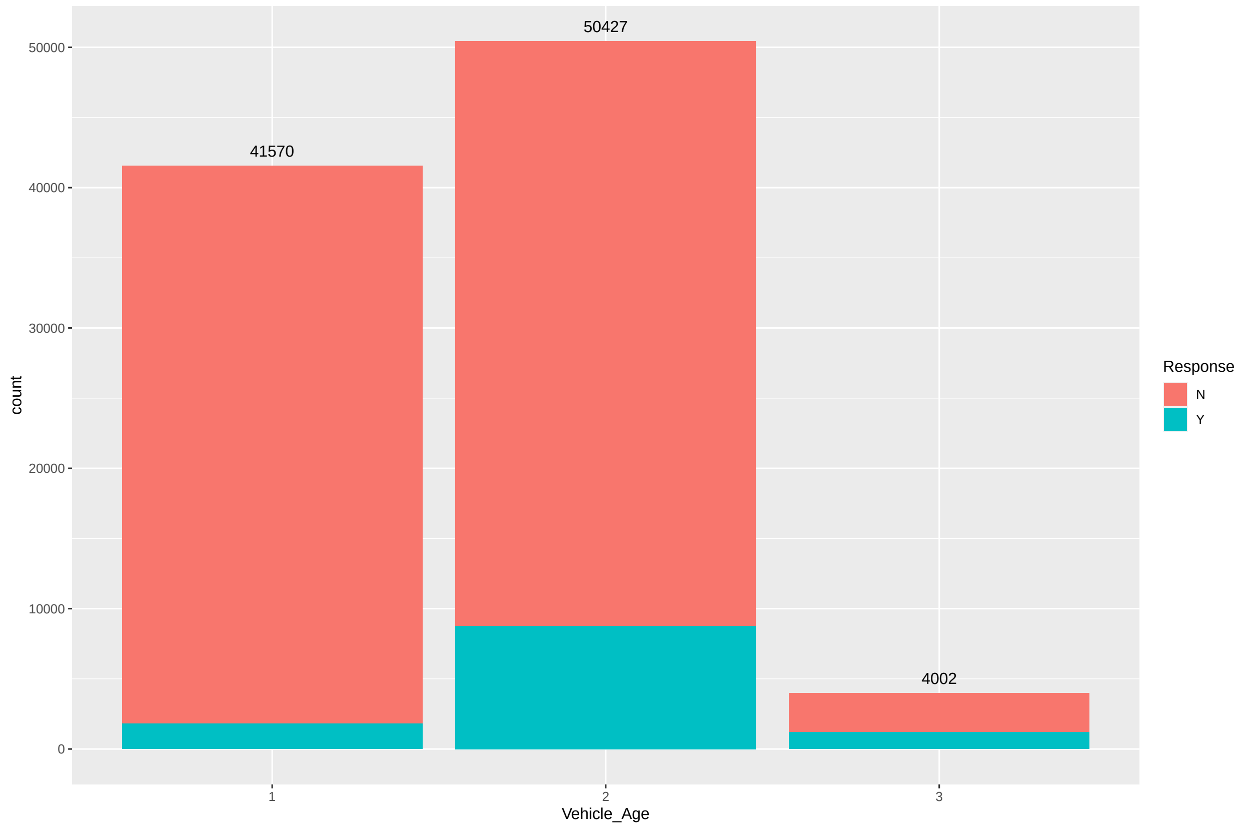
As we can see, the largest region had a total of 21690 customers, and the smallest region had only 8 customers. Given that region_code was not identified as a differential feature in the correlation plot, I decided not to use it in the development of the predictive model.

Length of Car Ownership

Next I explored the distribution of length of car ownership. I plotted a density plot of the ages of the drivers, grouped by the age of their vehicle.



As we can see, younger drivers tend to have newer cars. This is not surprising, as new drivers tend to buy new cars, and older drivers may have driven their cars for a period of time. We then explore the number of customers who took up the insurance based on the age of their car.

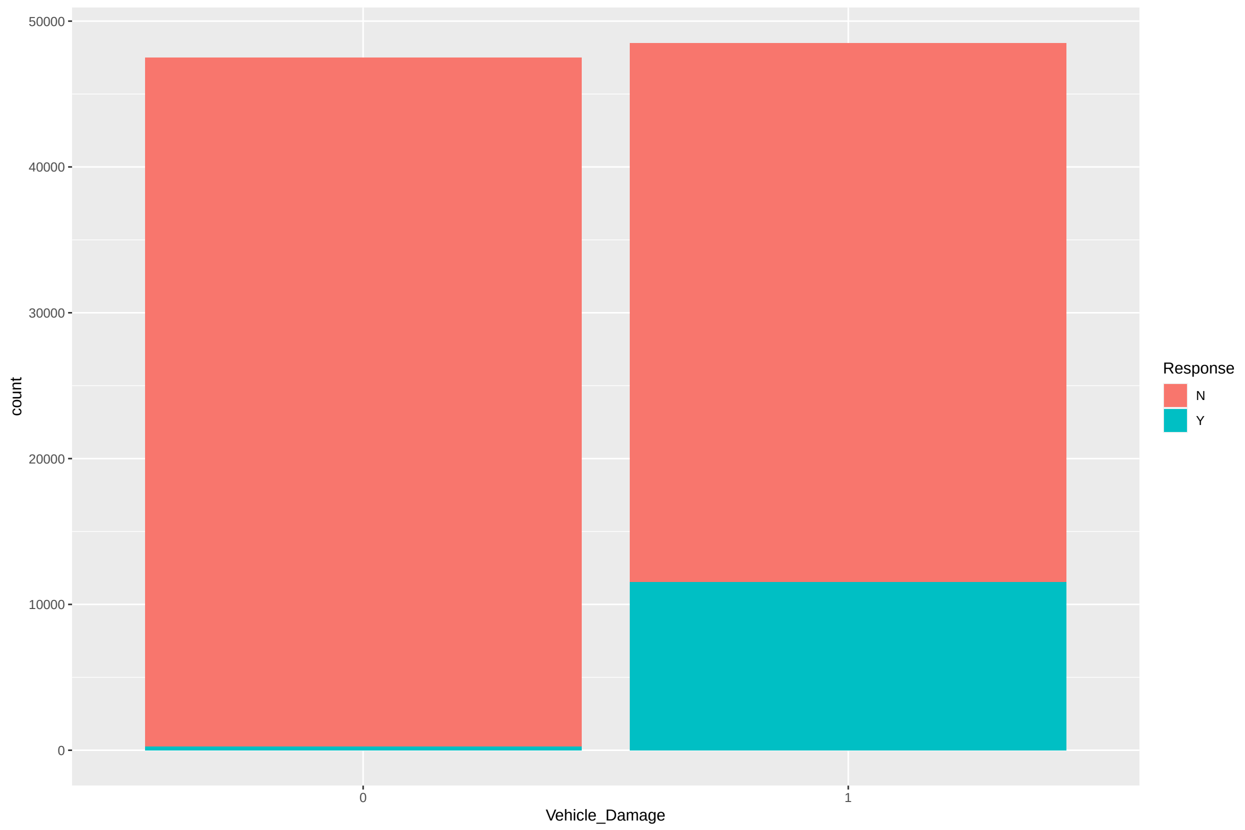


```
## `summarise()` regrouping output by 'Vehicle_Age' (override with `.groups` argument)

## # A tibble: 6 x 4
## # Groups:   Vehicle_Age [3]
##   Vehicle_Age Response     n percentage
##   <fct>       <fct>    <int>      <dbl>
## 1 1           N        39775         96
## 2 1           Y         1795          4
## 3 2           N        41654         83
## 4 2           Y         8773         17
## 5 3           N         2798         70
## 6 3           Y         1204         30
```

The graphs demonstrate that drivers with older vehicles are more likely to take up insurance. The percentage of drivers who choose to take up car insurance increases with the age of the car, with 4% of owners of new cars (<1 year old), 17% of owners of car between 1 and 2 years of age, and 30% of owners of cars at least 2 years old took up insurance

Past Vehicle Damage

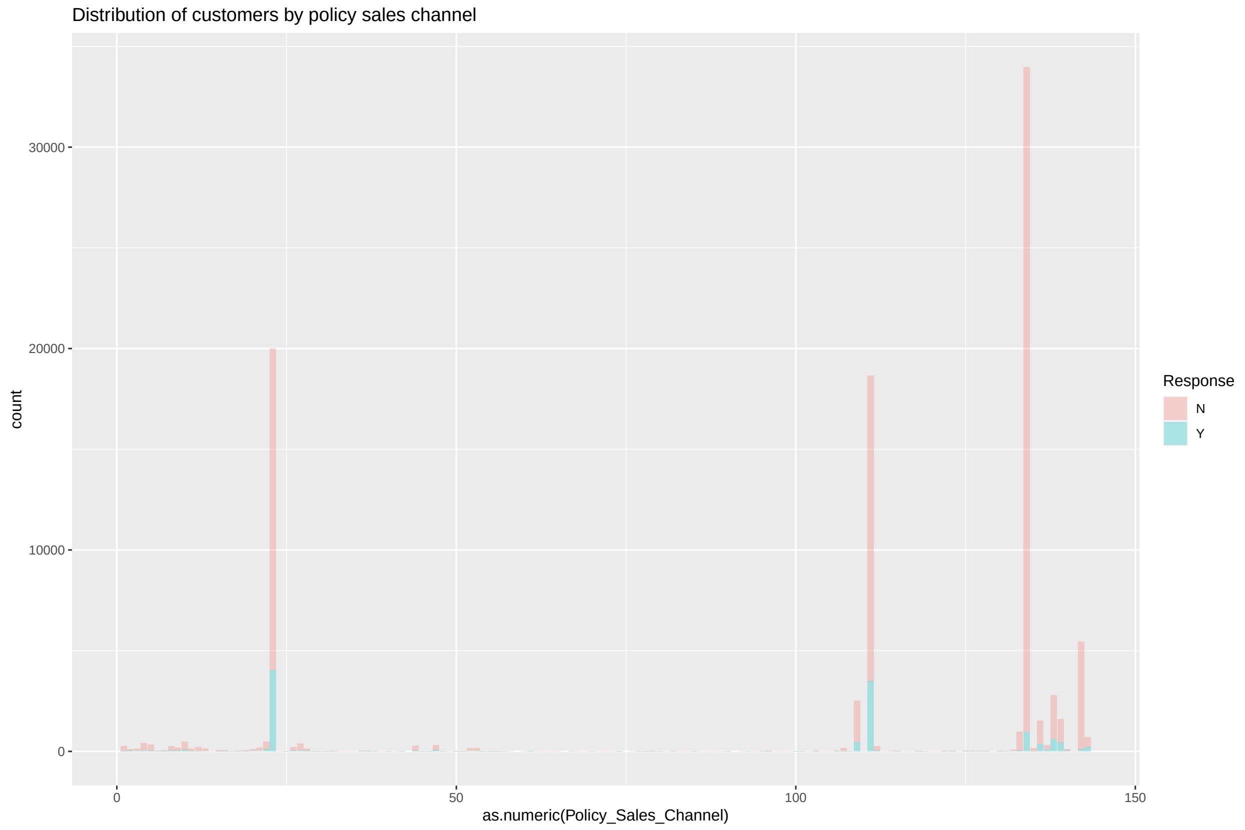
The number of customers with past vehicle damage is plotted, and the number of customers who took up vehicle insurance is highlighted.



We can see a clear distinction between those with a history of past vehicle damage. They are much more likely than the customers with no history of vehicle damage to take up vehicle insurance.

Distribution of Policy Sales Channels

The company utilises a large number of policy sales channels to connect with their customers. The following graph shows the distribution of customers across the various policy sales channels.



In this graph, we can see that there is a large variation in the customers obtained through the various sales channels, as well as in the success rates of the various channels.

To visualise the most successful sales channels, we sort the percentage of a positive responses in descending order:

```
## ‘summarise()‘ regrouping output by ’Policy_Sales_Channel’ (override with ‘.groups‘ argument)
```

```
## # A tibble: 6 x 4
## # Groups:   Policy_Sales_Channel [6]
##   Policy_Sales_Channel Response     n percentage
##                  <dbl> <fct>    <int>      <dbl>
## 1                   89 Y            1        100
## 2                  110 Y            1        100
## 3                   70 Y            2         67
## 4                   25 Y            1         50
## 5                   61 Y            1         50
## 6                  101 Y            3         50
```

Sorted in ascending order, the sales channels with the lowest rates of conversion are:

```
## ‘summarise()‘ regrouping output by ’Policy_Sales_Channel’ (override with ‘.groups‘ argument)
## # A tibble: 6 x 4
## # Groups:   Policy_Sales_Channel [6]
##   Policy_Sales_Channel Response     n percentage
##                  <dbl> <fct>    <int>      <dbl>
## 1                   15 Y            1          2
## 2                  142 Y          109          2
## 3                    1 Y            7          3
## 4                  133 Y           32          3
## 5                  134 Y          962          3
## 6                  100 Y            1          5
```

This shows that there are some policy sales channels that work better than others. Some of the data may be skewed because of the lack of customers in certain sales channels, but this feature shows significant differential potential and will be included into the machine learning model.

Model Building

Model Evaluation Metric

As the data is imbalanced and there are a lot less people who take up the insurance than those that do not, classification accuracy may not be the most suitable metric to use in evaluating between the various machine learning models. As such, we use the Area-under-curve (AUC) of the Receiver Operating Characteristic (ROC) Curve to distinguish between the various methods instead.

The ROC curve is a graphical plot that illustrates the diagnostic ability of the binary classifier system as its discrimination threshold is varied. It is created by plotting the true positive rate against the false positive rate at various threshold settings. Higher AUC values highlights a model's ability to correctly identify true positives with little misclassification of false positives.

In the case of actual prediction values generated by the machine learning model on the test dataset, the returned predictions are a probabilistic value between 0 to 1. To determine which values will be considered a positive or negative response, a probability threshold is required (e.g. Equating probabilities of > 0.3 to be a positive response). In such a scenario, it is important to find a threshold that balances between minimising false positives and maximising true positives.

Though setting a threshold for predictions can sometimes be regarded as over-training, and the Kaggle contest's aim was to determine the model with the highest AUC, I selected a probability threshold with the use of Youden's J statistic for the purpose of predicting outcomes in the validation dataset. Youden's J statistic is defined as the intercept of the ROC curve with the tangent at 45 degrees parallel to the no-discrimination line that is closest to the error-free point (0,1), and is employed as a criterion for selecting the optimum cut-off point when a diagnostic test gives a numeric rather than a dichotomous result.

With this evaluation metric in mind, I test a variety of various classification models on the edx_train dataset, and evaluate them based on the ROC-AUC and the J statistic.

Model Testing

The models in this section are tested through the following steps:

1. Tuning of the models are optimised for 5-fold cross-validation.
2. Train function from the caret package is utilised to produce the fit from the edx_train set. The model is trained with ROC as the metric of choice as previously explained. The features selected for training with the caret package include: Gender, Age, Previously_Insured, Vehicle_Age, Vehicle_Damage, and the Policy_Sales_Channel.
3. We index the predictions that utilise the best identified hyperparameters, and produce the ROC plot. This allows us to then calculate the coordinates of Youden's J statistic.
4. Predictions are made with the predict function by applying the fit to edx_test is calculated using the probability threshold identified with the J-statistic, and the confusion matrix is generated.
5. The ROC-AUC and the J statistic are noted, and consolidated into a list for comparison with other models.
6. Model with the best ROC-AUC and J-statistic will be selected for testing on the final validation set.

The algorithms used include: Naive Bayes, k-Nearest Neighbours, Generalized Linear Model, Linear Discriminatory Analysis, Boosted classification trees, Stochastic Gradient Boosting, Quadratic Discriminatory Analysis, Random forest, Boosted Logistic Regression, and eXtreme Gradient Boosting.

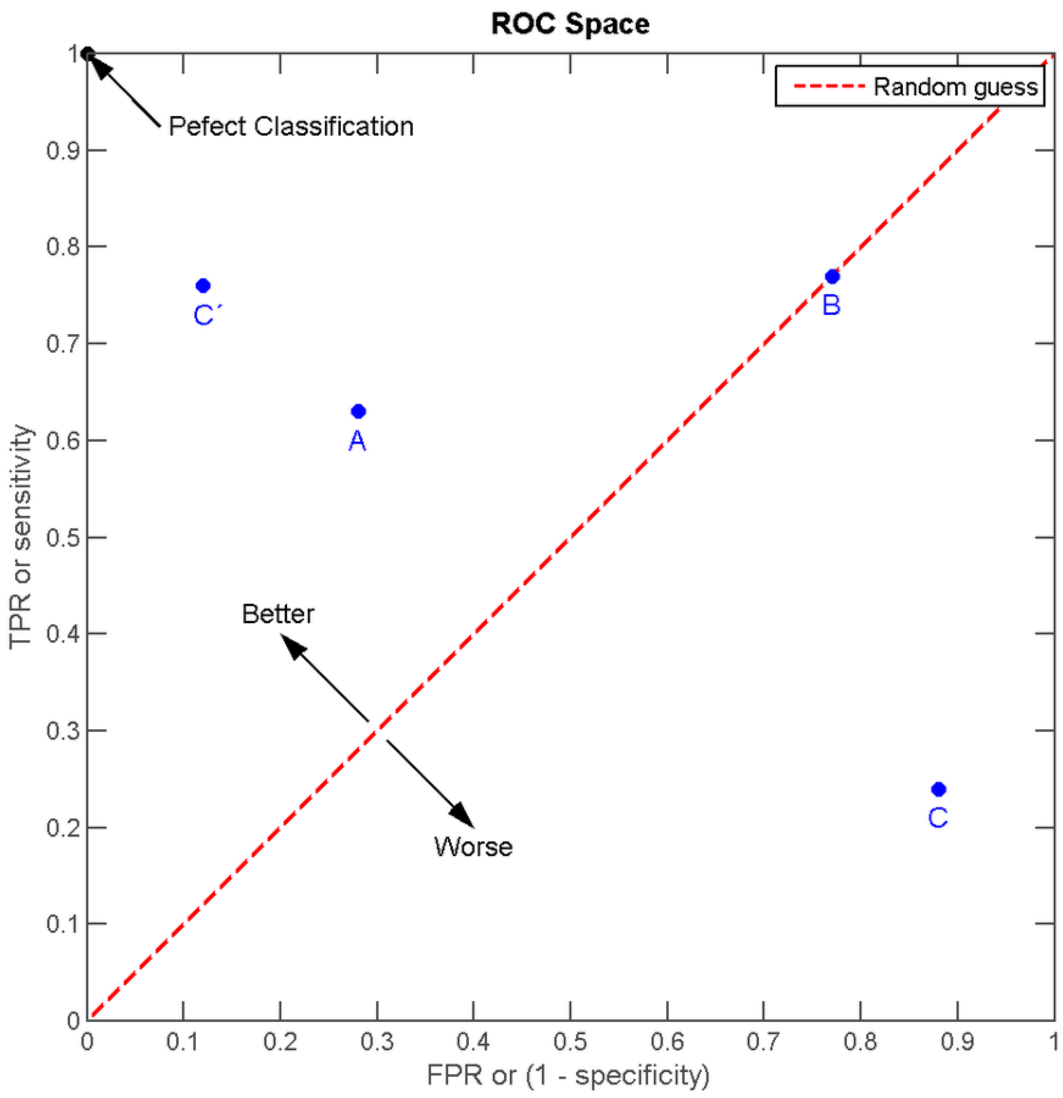


Figure 1: Basic characteristics of an ROC curve, taken from Wikipedia

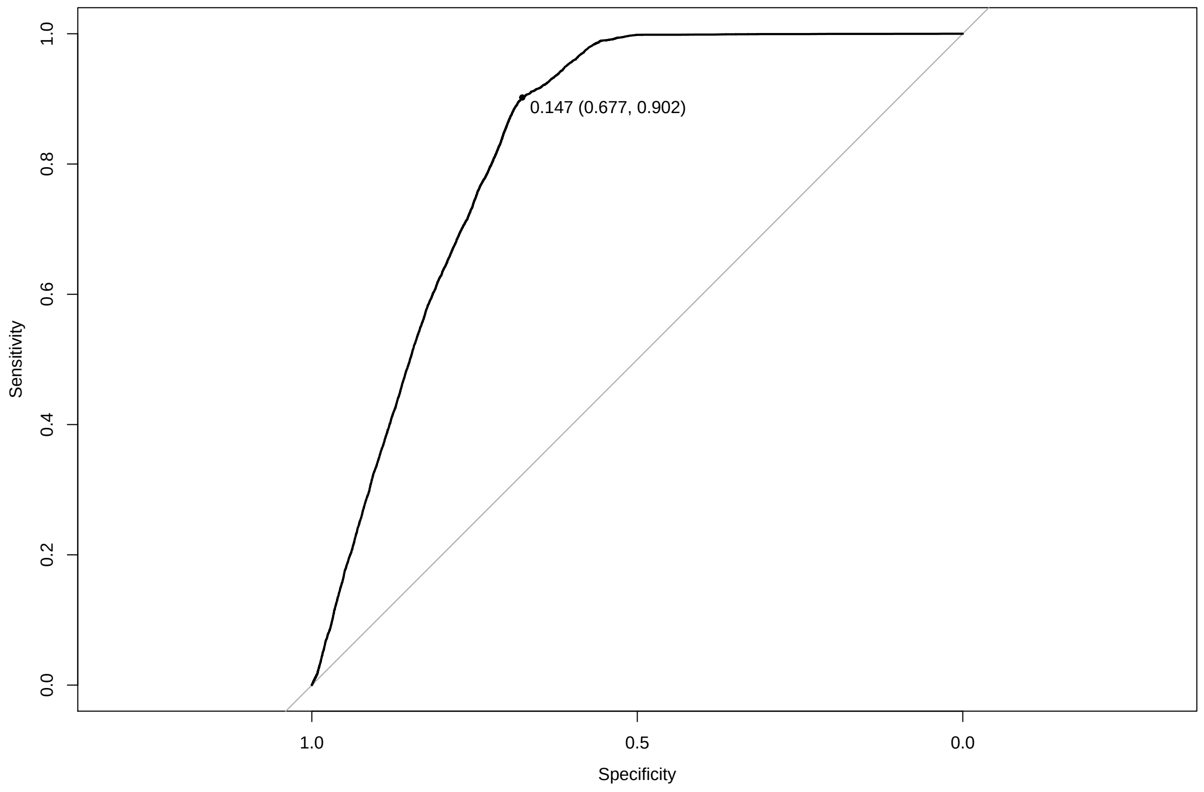
Naive Bayes

```r
# Set control parameters for cross validation
control_bayes <- trainControl(method = "cv", number = 5, savePredictions = TRUE,
    classProbs = TRUE, verboseIter = TRUE, summaryFunction = twoClassSummary)

# Train fit using caret package. Metric set to ROC
fit_bayes <- train(Response ~ Gender + Age + Previously_Insured +
    Vehicle_Age + Vehicle_Damage + Policy_Sales_Channel, method = "naive_bayes",
    data = edx_train, trControl = control_bayes, metric = "ROC")
```

```
## + Fold1: usekernel= TRUE, laplace=0, adjust=1
## - Fold1: usekernel= TRUE, laplace=0, adjust=1
## + Fold1: usekernel=FALSE, laplace=0, adjust=1
## - Fold1: usekernel=FALSE, laplace=0, adjust=1
## + Fold2: usekernel= TRUE, laplace=0, adjust=1
## - Fold2: usekernel= TRUE, laplace=0, adjust=1
## + Fold2: usekernel=FALSE, laplace=0, adjust=1
## - Fold2: usekernel=FALSE, laplace=0, adjust=1
## + Fold3: usekernel= TRUE, laplace=0, adjust=1
## - Fold3: usekernel= TRUE, laplace=0, adjust=1
## + Fold3: usekernel=FALSE, laplace=0, adjust=1
## - Fold3: usekernel=FALSE, laplace=0, adjust=1
## + Fold4: usekernel= TRUE, laplace=0, adjust=1
## - Fold4: usekernel= TRUE, laplace=0, adjust=1
## + Fold4: usekernel=FALSE, laplace=0, adjust=1
## - Fold4: usekernel=FALSE, laplace=0, adjust=1
## + Fold5: usekernel= TRUE, laplace=0, adjust=1
## - Fold5: usekernel= TRUE, laplace=0, adjust=1
## + Fold5: usekernel=FALSE, laplace=0, adjust=1
## - Fold5: usekernel=FALSE, laplace=0, adjust=1
## Aggregating results
## Selecting tuning parameters
## Fitting laplace = 0, usekernel = TRUE, adjust = 1 on full training set
```

```r
# Selecting best hyperparameters for Naive Bayes model to
# plot the ROC curve
index_bayes <- fit_bayes$pred$usekernel == as.logical(fit_bayes$bestTune[2])

roc_bayes <- roc(fit_bayes$pred$obs[index_bayes], fit_bayes$pred$Y[index_bayes])
```

```r
# Plotting ROC curve with Youden's index highlighted
plot(roc_bayes, print.thres = "best", print.thres.best.method = "youden")
```



```r
J_bayes <- coords(roc_bayes, x = "best", input = c("threshold",
    "specificity", "sensitivity"), best.method = "youden", transpose = TRUE)

# Generating predictions on the edx_test set with the fit,
# optimised for probabilistic threshold set by Youden's
# statistic
y_hat_bayes <- ifelse(predict(fit_bayes, edx_test, type = "prob") >
    J_bayes[1], "Y", "N")
y_hat_bayes <- as.factor(y_hat_bayes[, 2])
confusionMatrix(y_hat_bayes, edx_test$Response, positive = "Y",
    mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     N     Y
##          N 14273   270
##          Y  6779  2674
##
##                Accuracy : 0.7062
##                  95% CI : (0.7004, 0.712)
##     No Information Rate : 0.8773
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3005
##
```

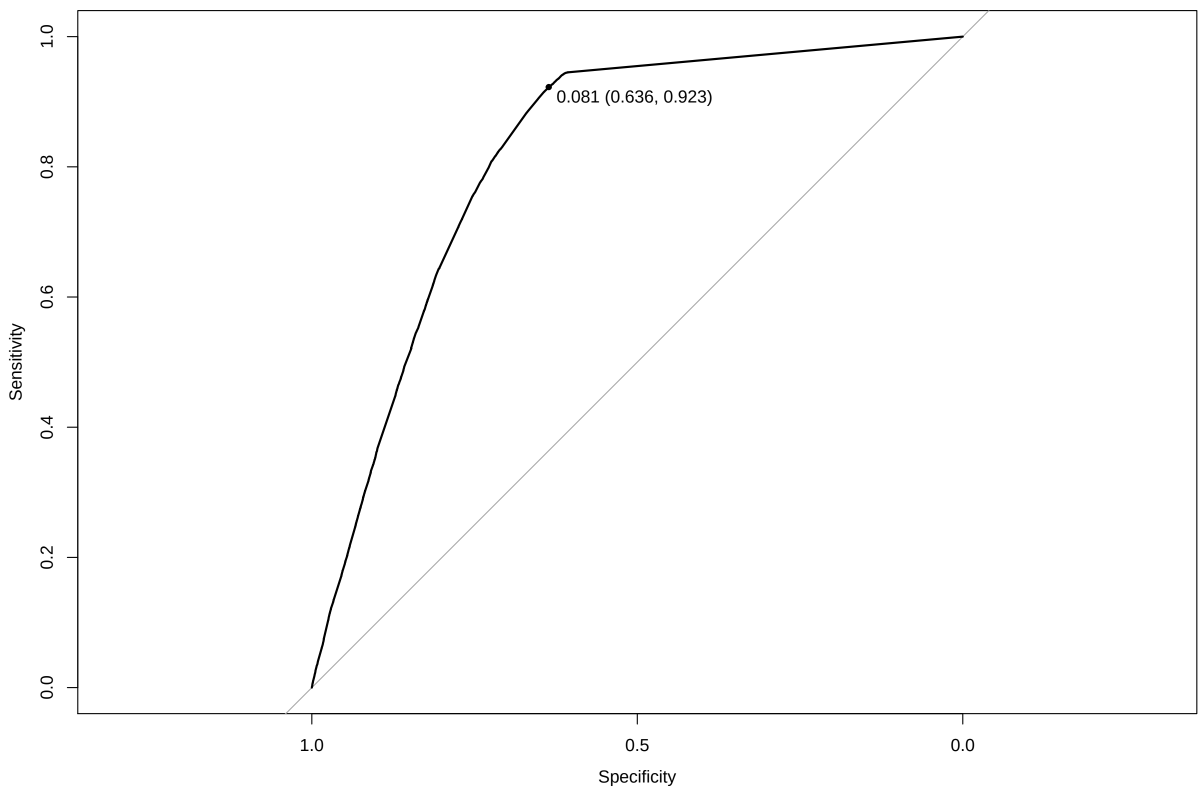
```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9083
##             Specificity : 0.6780
##          Pos Pred Value : 0.2829
##          Neg Pred Value : 0.9814
##               Precision : 0.2829
##                  Recall : 0.9083
##                      F1 : 0.4314
##              Prevalence : 0.1227
##          Detection Rate : 0.1114
##    Detection Prevalence : 0.3939
##       Balanced Accuracy : 0.7931
##
##        'Positive' Class : Y
##
```

```
# Results of the model are added to a rolling list to compare
# models
model_AUC <- data.frame("Naive Bayes", round(roc_bayes$auc, digits = 4),
    J = round((as.numeric(J_bayes[3]) + as.numeric(J_bayes[2]) -
        1), digits = 4))
colnames(model_AUC) <- c("Model", "AUC", "J-Statistic")
model_AUC
```

```
##         Model    AUC J-Statistic
## 1 Naive Bayes 0.8304      0.5788
```

```
rm(control_bayes, fit_bayes, roc_bayes, J_bayes, y_hat_bayes)
```

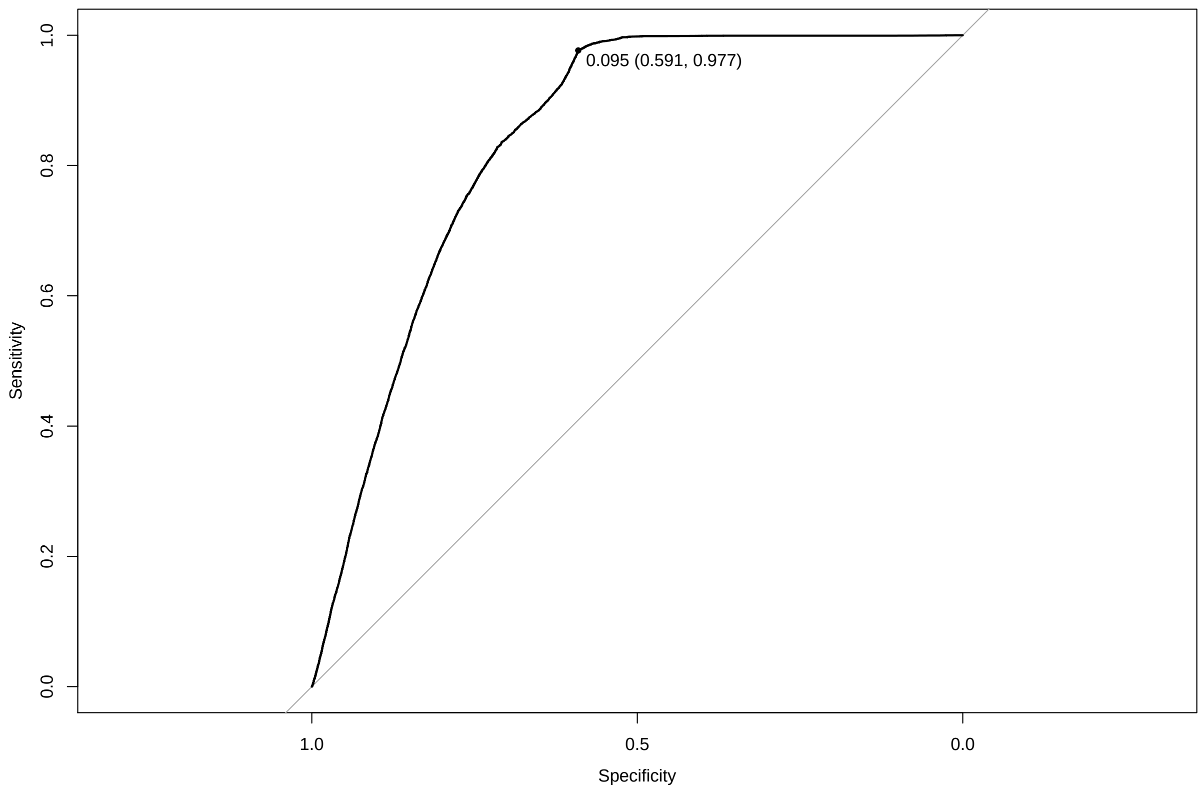
KNN - K-Nearest Neighbours



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     N     Y
##          N 13378   227
##          Y  7674  2717
##
##                Accuracy : 0.6707
##                  95% CI : (0.6647, 0.6767)
##     No Information Rate : 0.8773
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2674
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9229
##             Specificity : 0.6355
##          Pos Pred Value : 0.2615
##          Neg Pred Value : 0.9833
##               Precision : 0.2615
##                  Recall : 0.9229
##                      F1 : 0.4075
##              Prevalence : 0.1227
##          Detection Rate : 0.1132
##    Detection Prevalence : 0.4330
##       Balanced Accuracy : 0.7792
##
##        'Positive' Class : Y
##
```

```
##         Model    AUC J-Statistic
## 1 Naive Bayes 0.8304      0.5788
## 2         kNN 0.8197      0.5585
```

GLM - Generalized Linear Model

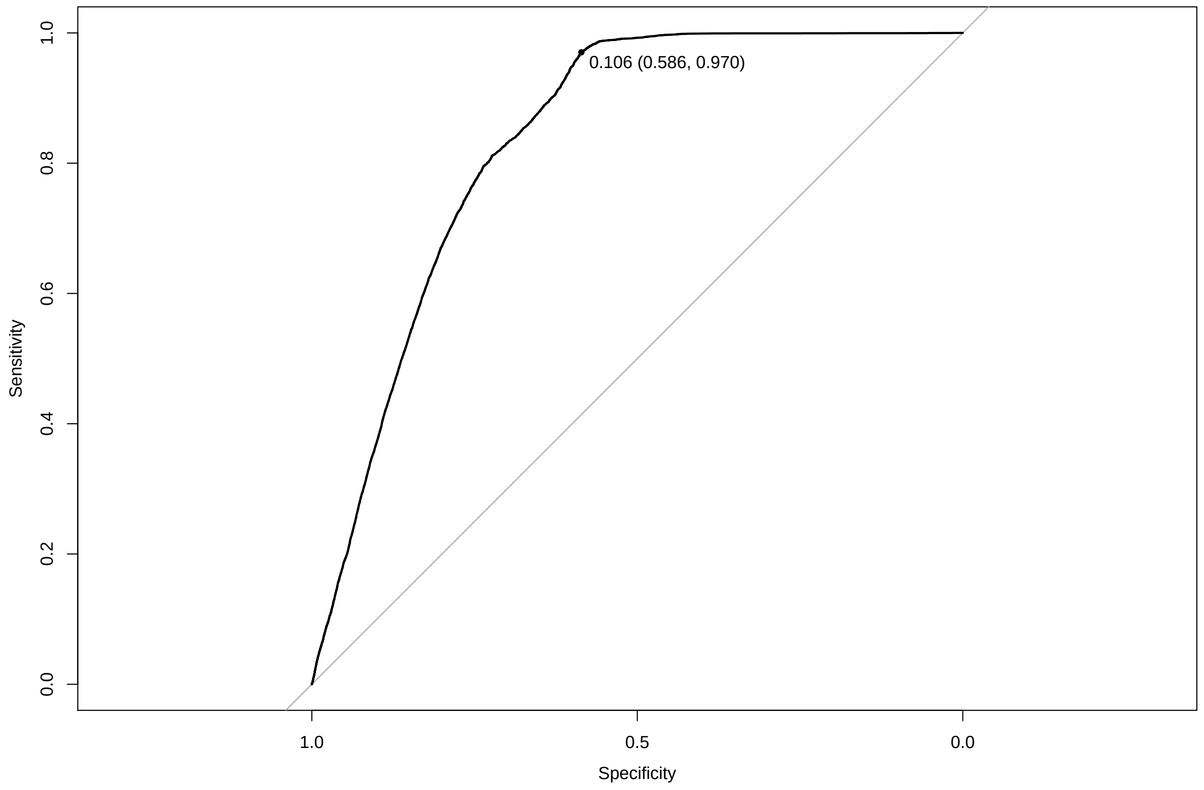


```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     N     Y
##          N 12393    53
##          Y  8659  2891
##
##                Accuracy : 0.6369
##                  95% CI : (0.6308, 0.643)
##     No Information Rate : 0.8773
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2528
##
```

```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9820
##             Specificity : 0.5887
##          Pos Pred Value : 0.2503
##          Neg Pred Value : 0.9957
##               Precision : 0.2503
##                  Recall : 0.9820
##                      F1 : 0.3989
##              Prevalence : 0.1227
##          Detection Rate : 0.1205
##    Detection Prevalence : 0.4813
##       Balanced Accuracy : 0.7853
##
##        'Positive' Class : Y
##
```

```
##                      Model    AUC J-Statistic
## 1              Naive Bayes 0.8304      0.5788
## 2                      kNN 0.8197      0.5585
## 3 Generalized Linear Model 0.8377      0.5674
```

LDA - Linear Discriminatory Analysis

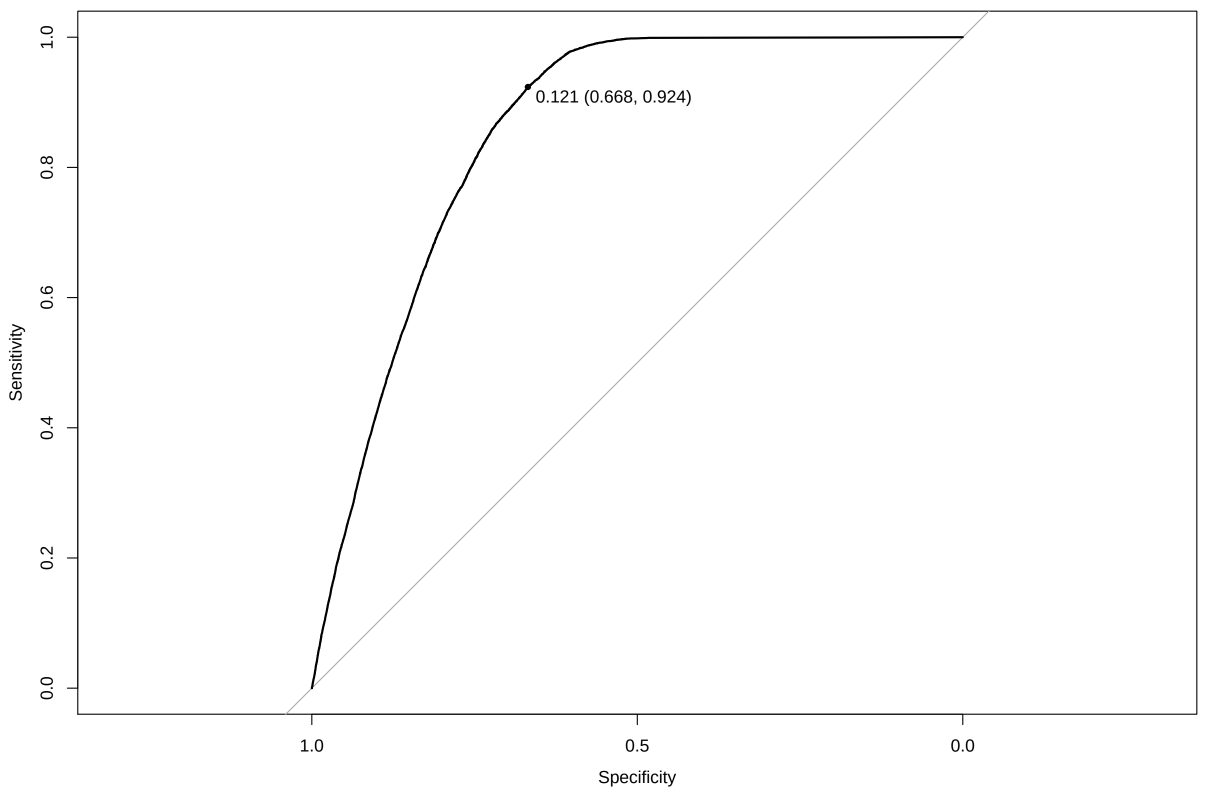


```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction     N     Y
##          N 12302    70
##          Y  8750  2874
##
##                Accuracy : 0.6324
##                  95% CI : (0.6263, 0.6385)
##     No Information Rate : 0.8773
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2472
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9762
##             Specificity : 0.5844
##          Pos Pred Value : 0.2472
##          Neg Pred Value : 0.9943
##               Precision : 0.2472
##                  Recall : 0.9762
##                      F1 : 0.3946
##              Prevalence : 0.1227
##          Detection Rate : 0.1198
##    Detection Prevalence : 0.4844
##       Balanced Accuracy : 0.7803
##
##        'Positive' Class : Y
##
```

```
##                             Model    AUC J-Statistic
## 1                     Naive Bayes 0.8304      0.5788
## 2                             kNN 0.8197      0.5585
## 3        Generalized Linear Model 0.8377      0.5674
## 4 Quadratiic Discriminant Analysis 0.8345      0.5562
```

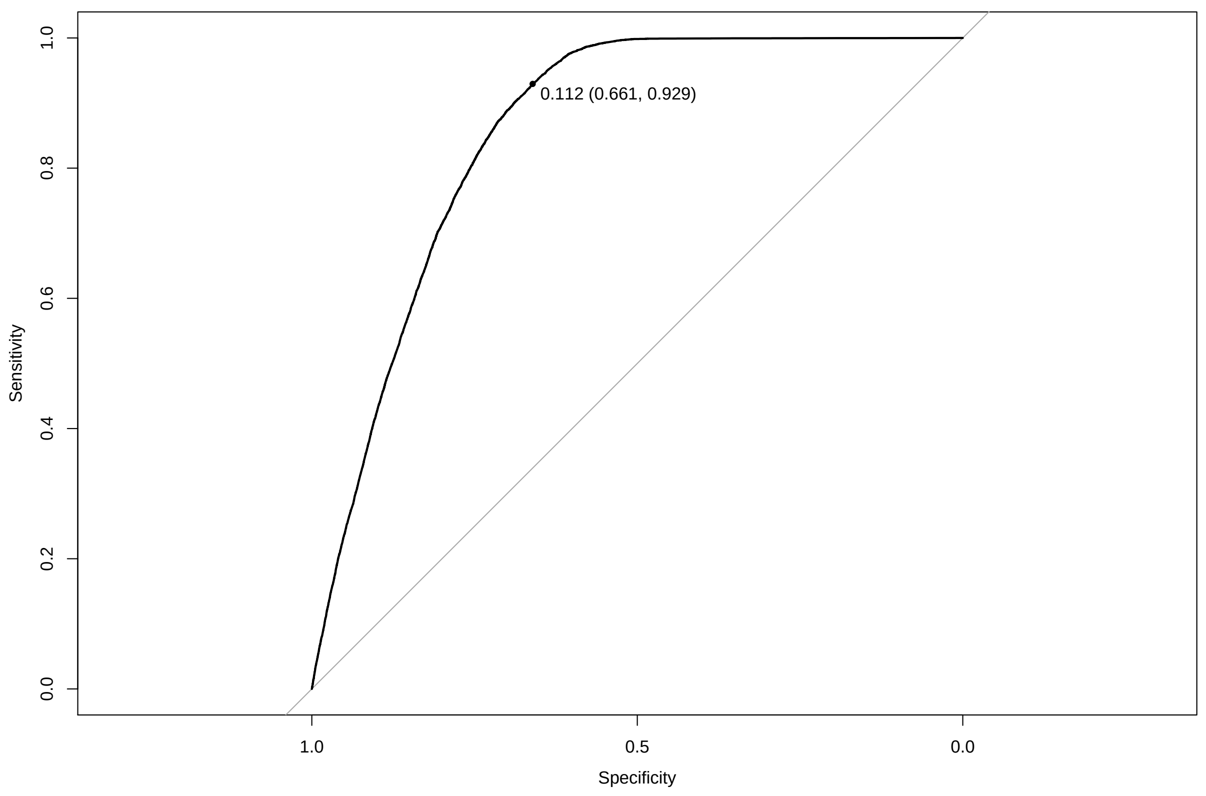
ada - Boosted classification trees



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     N     Y
##          N 13857   181
##          Y  7195  2763
##
##                Accuracy : 0.6926
##                  95% CI : (0.6867, 0.6985)
##     No Information Rate : 0.8773
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2947
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9385
##             Specificity : 0.6582
##          Pos Pred Value : 0.2775
##          Neg Pred Value : 0.9871
##               Precision : 0.2775
##                  Recall : 0.9385
##                      F1 : 0.4283
##              Prevalence : 0.1227
##          Detection Rate : 0.1151
##    Detection Prevalence : 0.4150
##       Balanced Accuracy : 0.7984
##
##        'Positive' Class : Y
##
```

```
##                               Model    AUC J-Statistic
## 1                       Naive Bayes 0.8304      0.5788
## 2                               kNN 0.8197      0.5585
## 3         Generalized Linear Model 0.8377      0.5674
## 4 Quadratiic Discriminant Analysis 0.8345      0.5562
## 5     Boosted Classification Trees 0.8537      0.5916
```

GBM - Stochastic Gradient Boosting

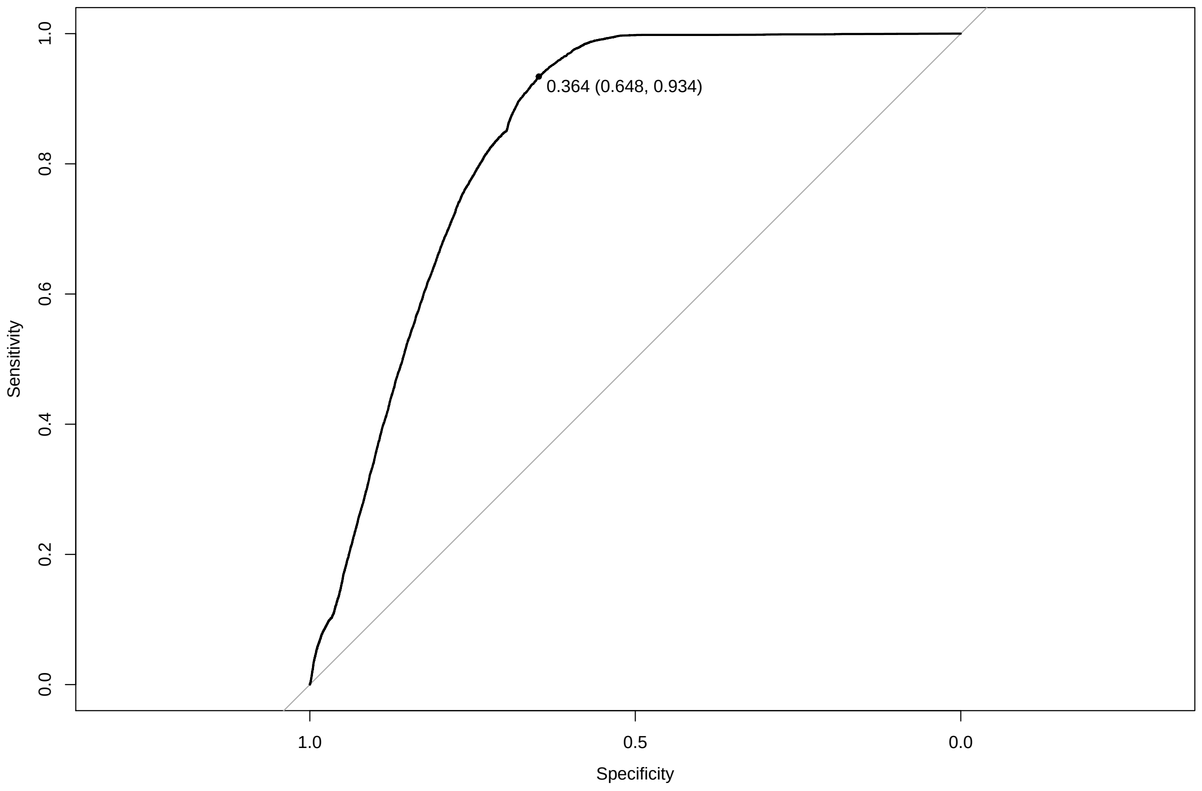


```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     N     Y
##          N 13725   170
##          Y  7327  2774
##
##                Accuracy : 0.6876
##                  95% CI : (0.6817, 0.6934)
##     No Information Rate : 0.8773
##     P-Value [Acc > NIR] : 1
```

```
##
##                   Kappa : 0.2905
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9423
##             Specificity : 0.6520
##          Pos Pred Value : 0.2746
##          Neg Pred Value : 0.9878
##               Precision : 0.2746
##                  Recall : 0.9423
##                      F1 : 0.4253
##              Prevalence : 0.1227
##          Detection Rate : 0.1156
##    Detection Prevalence : 0.4209
##       Balanced Accuracy : 0.7971
##
##        'Positive' Class : Y
##
```

```
##                             Model    AUC J-Statistic
## 1                     Naive Bayes 0.8304      0.5788
## 2                             kNN 0.8197      0.5585
## 3         Generalized Linear Model 0.8377      0.5674
## 4 Quadratiic Discriminant Analysis 0.8345      0.5562
## 5     Boosted Classification Trees 0.8537      0.5916
## 6    Stochastic Gradient Boostiing  0.854      0.5901
```

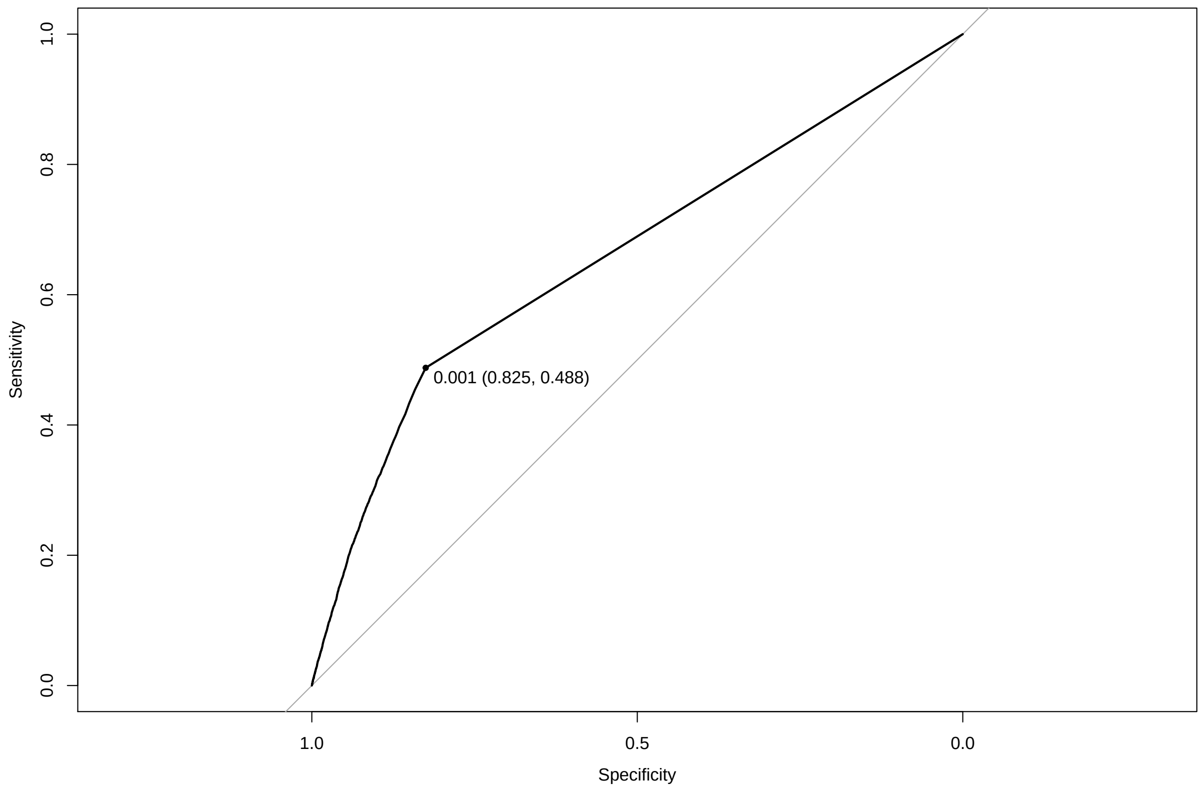
QDA - Quadratic Discriminatory Analysis



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     N     Y
##          N 13526   167
##          Y  7526  2777
##
##                Accuracy : 0.6794
##                  95% CI : (0.6735, 0.6853)
##     No Information Rate : 0.8773
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2823
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9433
##             Specificity : 0.6425
##          Pos Pred Value : 0.2695
##          Neg Pred Value : 0.9878
##               Precision : 0.2695
##                  Recall : 0.9433
##                      F1 : 0.4193
##              Prevalence : 0.1227
##          Detection Rate : 0.1157
##    Detection Prevalence : 0.4294
##       Balanced Accuracy : 0.7929
##
##        'Positive' Class : Y
##
```

```
##                              Model    AUC J-Statistic
## 1                      Naive Bayes 0.8304      0.5788
## 2                              kNN 0.8197      0.5585
## 3         Generalized Linear Model 0.8377      0.5674
## 4 Quadratiic Discriminant Analysis 0.8345      0.5562
## 5     Boosted Classification Trees 0.8537      0.5916
## 6    Stochastic Gradient Boostiing  0.854      0.5901
## 7 Quadratiic Discriminant Analysis 0.8372      0.5823
```

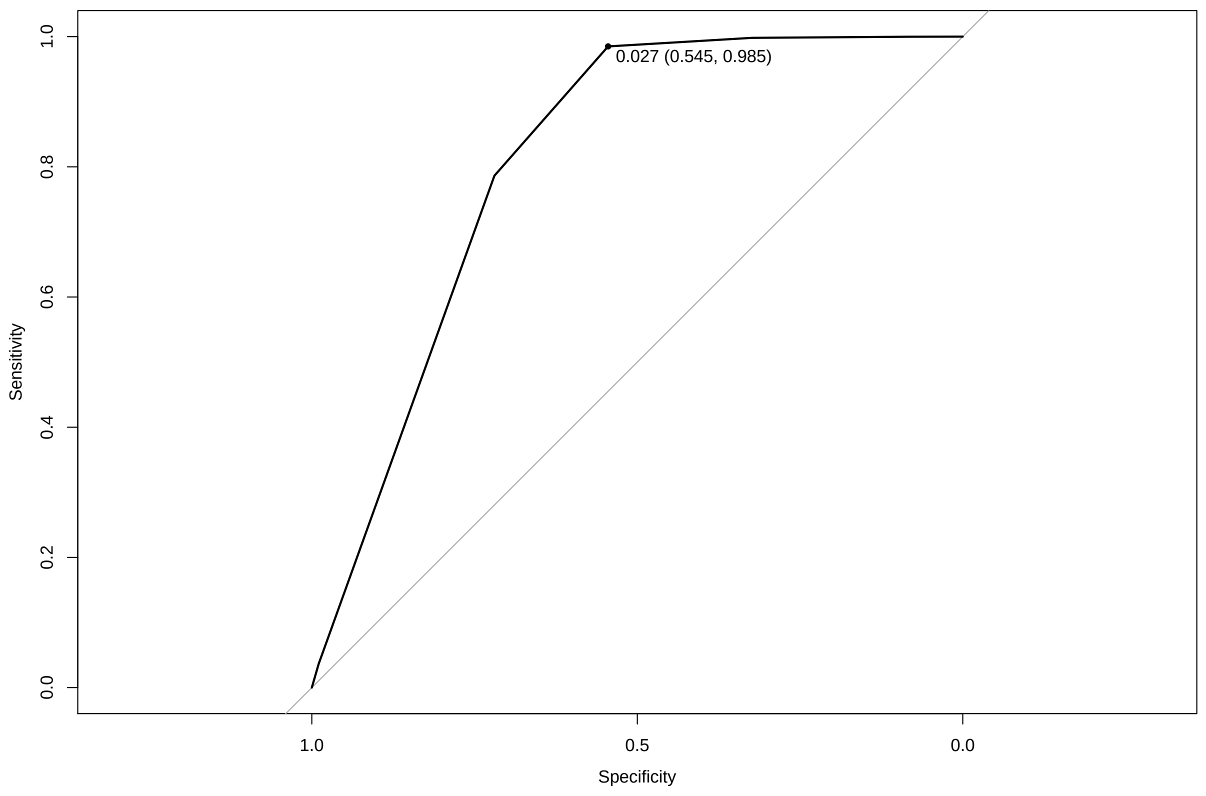
RF - Random forest



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     N     Y
##          N 17665  1645
##          Y  3387  1299
##
##                Accuracy : 0.7903
##                  95% CI : (0.7851, 0.7954)
##     No Information Rate : 0.8773
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2235
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.44124
##             Specificity : 0.83911
##          Pos Pred Value : 0.27721
##          Neg Pred Value : 0.91481
##               Precision : 0.27721
##                  Recall : 0.44124
##                      F1 : 0.34050
##              Prevalence : 0.12269
##          Detection Rate : 0.05413
##    Detection Prevalence : 0.19528
##       Balanced Accuracy : 0.64017
##
##        'Positive' Class : Y
##
```

```
##                              Model    AUC J-Statistic
## 1                      Naive Bayes 0.8304      0.5788
## 2                              kNN 0.8197      0.5585
## 3         Generalized Linear Model 0.8377      0.5674
## 4 Quadratiic Discriminant Analysis 0.8345      0.5562
## 5     Boosted Classification Trees 0.8537      0.5916
## 6    Stochastic Gradient Boostiing  0.854      0.5901
## 7 Quadratiic Discriminant Analysis 0.8372      0.5823
## 8                   Random Forests 0.6609      0.3128
```

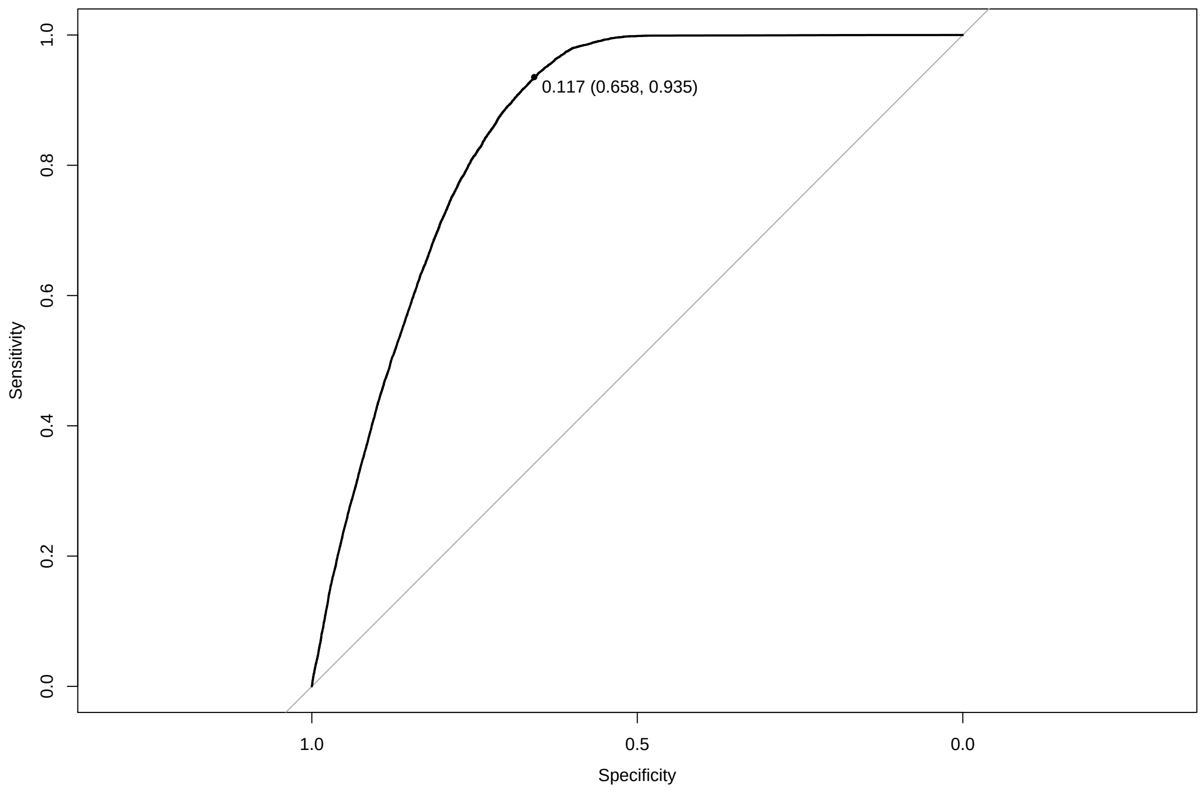
LogitBoost - Boosted Logistic Regression



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     N     Y
##          N 11882    58
##          Y  9170  2886
##
##                Accuracy : 0.6154
##                  95% CI : (0.6092, 0.6216)
##     No Information Rate : 0.8773
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2337
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9803
##             Specificity : 0.5644
##          Pos Pred Value : 0.2394
##          Neg Pred Value : 0.9951
##               Precision : 0.2394
##                  Recall : 0.9803
##                      F1 : 0.3848
##              Prevalence : 0.1227
##          Detection Rate : 0.1203
##    Detection Prevalence : 0.5024
##       Balanced Accuracy : 0.7724
##
##        'Positive' Class : Y
##
```

```
##                             Model    AUC J-Statistic
## 1                     Naive Bayes 0.8304      0.5788
## 2                             kNN 0.8197      0.5585
## 3         Generalized Linear Model 0.8377      0.5674
## 4 Quadratiic Discriminant Analysis 0.8345      0.5562
## 5     Boosted Classification Trees 0.8537      0.5916
## 6    Stochastic Gradient Boostiing  0.854      0.5901
## 7 Quadratiic Discriminant Analysis 0.8372      0.5823
## 8                   Random Forests 0.6609      0.3128
## 9      Boosted Logistic Regression 0.8087      0.5298
```

XGB - eXtreme Gradient Boosting



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     N     Y
##          N 13925   181
##          Y  7127  2763
##
```

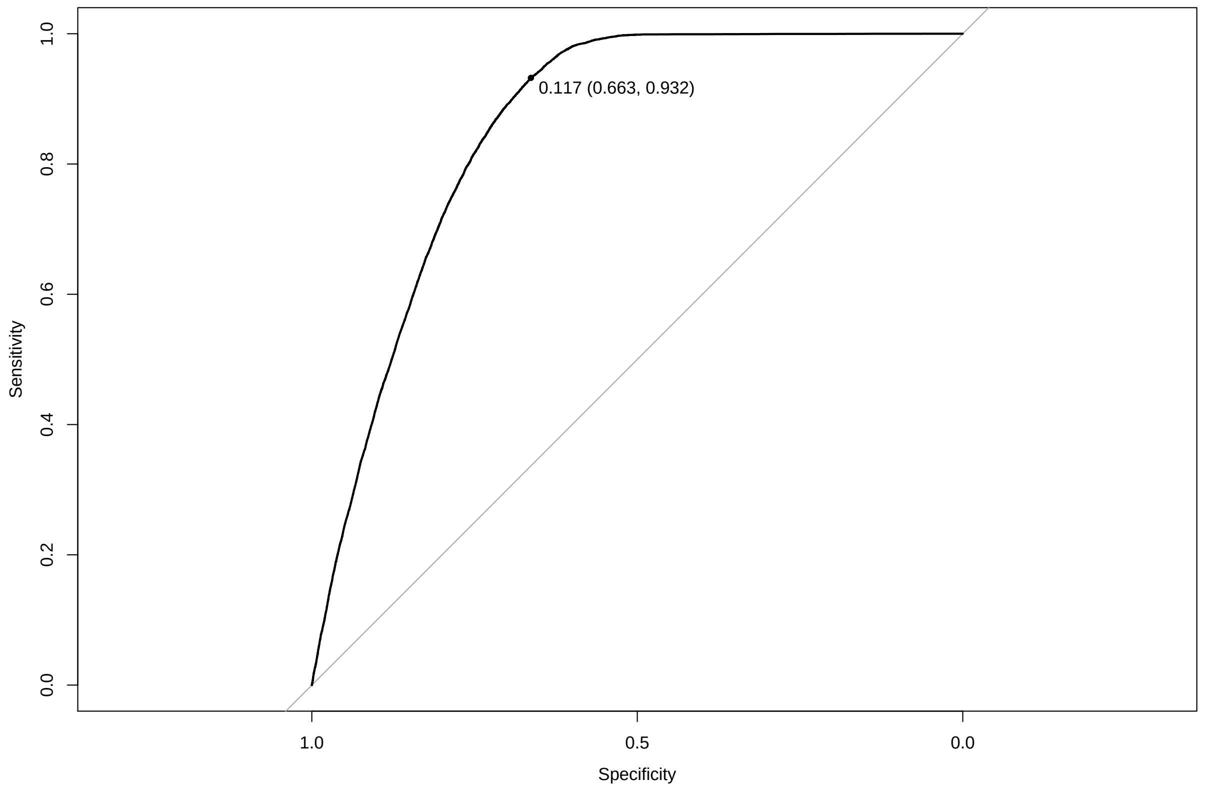
```
##                Accuracy : 0.6954
##                  95% CI : (0.6896, 0.7013)
##     No Information Rate : 0.8773
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2978
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9385
##             Specificity : 0.6615
##          Pos Pred Value : 0.2794
##          Neg Pred Value : 0.9872
##               Precision : 0.2794
##                  Recall : 0.9385
##                      F1 : 0.4306
##              Prevalence : 0.1227
##          Detection Rate : 0.1151
##    Detection Prevalence : 0.4122
##       Balanced Accuracy : 0.8000
##
##        'Positive' Class : Y
##
```

```
##                              Model    AUC J-Statistic
## 1                      Naive Bayes 0.8304      0.5788
## 2                              kNN 0.8197      0.5585
## 3          Generalized Linear Model 0.8377      0.5674
## 4  Quadratiic Discriminant Analysis 0.8345      0.5562
## 5      Boosted Classification Trees 0.8537      0.5916
## 6     Stochastic Gradient Boostiing  0.854      0.5901
## 7  Quadratiic Discriminant Analysis 0.8372      0.5823
## 8                    Random Forests 0.6609      0.3128
## 9       Boosted Logistic Regression 0.8087      0.5298
## 10        eXtreme Gradient Boosting 0.8551      0.5937
```

Results

Final Model

The best results for both ROC-AUC and the J statistic were achieved with the extreme gradient boosting method. As such, I will proceed to train this model with the full edx dataset, and test the model on the final validation set.



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     N     Y
##          N 17520   278
##          Y  8800  3400
##
##                Accuracy : 0.6974
##                  95% CI : (0.6921, 0.7026)
##     No Information Rate : 0.8774
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2955
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9244
##             Specificity : 0.6657
##          Pos Pred Value : 0.2787
##          Neg Pred Value : 0.9844
##               Precision : 0.2787
##                  Recall : 0.9244
##                      F1 : 0.4283
##              Prevalence : 0.1226
##          Detection Rate : 0.1133
##    Detection Prevalence : 0.4067
##       Balanced Accuracy : 0.7950
##
##        'Positive' Class : Y
##
```

```
##                                                 Model    AUC J-Statistic
## 1                                         Naive Bayes 0.8304      0.5788
## 2                                                 kNN 0.8197      0.5585
## 3                           Generalized Linear Model 0.8377      0.5674
## 4                    Quadratiic Discriminant Analysis 0.8345      0.5562
## 5                       Boosted Classification Trees 0.8537      0.5916
## 6                        Stochastic Gradient Boostiing  0.854      0.5901
## 7                    Quadratiic Discriminant Analysis 0.8372      0.5823
## 8                                      Random Forests 0.6609      0.3128
## 9                       Boosted Logistic Regression 0.8087      0.5298
## 10                         eXtreme Gradient Boosting 0.8551      0.5937
## 11 Final Validation Set - eXtreme Gradient Boosting 0.8556      0.5956
```

With this final model, we were able to achieve an AUC of 0.855, and a J statistic of 0.595. This was similar to the performance achieved during the model selection phase and proved that the model was robust when applied to a larger dataset. The model classified 92.69% of positive responders correctly, while only misclassifying 33.75% of negative responders. Considering the imbalanced dataset that where only 13.9% of customers actually responded positively to the vehicle insurance, this is a great improvement and will reduce a significant proportion of unnecessary marketing spend.

Conclusion

This project was a supervised learning, classification problem that required the maximising of true positives, while minimising the number of false positives classified. The extreme gradient boost method was able to achieve the best performance in this project and was able to significantly reduce the number of false positives, while maintaining a high level of specificity predicting true positives. The model achieved an AUC of 0.855, as well as a J statistic of 0.595. 92.69% of positive responders were classified correctly, and only 33.75% of negative responders were classified as false positives. The project highlighted the importance of utilising metrics other than overall accuracy, especially in the case of imbalanced datasets.

There were several limitations faced in this project. Firstly, the dataset was too large and had to be reduced substantially because of the lack of processing power. As a result, the model may not be fully optimised and can be improved with the use of the full dataset. Secondly, the resulting predictions were also calibrated based on a threshold specified to maximise both specificity and sensitivity. This threshold can be changed to improve sensitivity such that they can pick up on the remainder of the positive cases should the gains of doing so outweigh the costs.

Overall, the use of machine learning models was incredibly beneficial in this case. With the information captured by the prediction model, the insurance company would be able to reduce their marketing expenditure by only reaching out to customers who were more likely to purchase the vehicle insurance.