Insurance Cross-sell Project

Pragya Bansal

2021-01-03

Contents

1 1 Introduction

This report is part of the capstone project of HarvardX's Data Science Professional Certificate program. As we had to choose our own dataset, I picked up the dataset from Kaggle which requires a prediction of customers' response for cross selling a different financial product.

The project aims at building a model to predict whether the customers who bought health insurance during the year will also be interested in vehicle Insurance provided by the company. Such a model can help the company plan its communication strategy to reach out to those existing customers and thus reduce marketing costs by tapping captive customer base.

This report is organized in four sections wherein Section 1 describes the dataset and summarizes the goal of the project, Section 2 explains the process and techniques used such as data cleaning, data exploration and visualization and applying those to the modelling approach, Section 3 presents the modeling results and the model performance and Section 4 concludes with a brief summary of the report, its limitations and future work.

1.1 1.1 Relevant Dataset

This project paper is based on Health Insurance Cross Sell dataset available at: https://www.kaggle.com/anmolkumar/health-insurance-cross-sell-prediction.

The file named train.csv contains 381,109 observations across 12 variables. It contains customer information about demographics (gender, age, region code type, previously insured), Vehicles (Vehicle Age, Damage), Policy (Premium, sourcing channel).

1.2 1.2 Evaluation Metrics

The evaluation of machine learning algorithms is comprised of comparing the predicted outcome with the actual outcome. Since this is a classification problem, wherein outcome is a two factor variable whether the customer is interested or not, we need to measure the accuracy of the prediction as measured by overall accuracy.

However, classification algorithms are typically impacted by the problems of false positive (ie Predict an event when there was no event) and false negative (Predict no event when in fact there was an event). Thus, besides accuracy, we need to balance the trade-off between the true positive rate and false positive rate for our predictive model. In this case, we don't wish to miss out on any potential customer thus need to identify all customers which say yes and will do ok with targeting some customers who are not interested but model classifies them as interested.

Therefore, we shall also look at area under ROC curve or AUC. It is a plot of the false positive rate ie Specificity (x-axis) versus the true positive rate ie Sensitivity (y-axis) for a number of different candidate threshold values between 0.0 and 1.0. This can be obtained from ROCR package available in CRAN library.

1.3 1.3 Methodology and Workflow

We followed below key steps:

• Data preparation: We download train.csv dataset from Kaggle website and split into subsets for training and testing set. There is no specific requirement for cross validation, thus we split it into two subsets. The already given test set does not have response column and thus cannot be used for validation.

• Data exploration: We run some data exploration to see how the features impact the outcome. The information and insights obtained during exploration will help to build the machine learning model.

• Data cleaning: We remove unnecessary information which can slow the processing and convert the features into numeric variables.

• Data modeling: This step involves identification of the most important features that help to predict the outcome and measuring the efficacy of such prediction against identified project goal. We started with very simple models of random prediction and then moved to more complex models of logistic regression and XGBoost. While some of the models gave high accuracy, they did not balance the specificity and sensitivity issues. Thus, our final choice of model is based on the one which helps achieve higher ROC score without compromising the accuracy much.

2 2 Process and Technique

2.1 2.1 Data Preparation

We split dataset in two parts, the training set and the testing set with 80% and 20% of the original dataset respectively. We download the requisite R packages from CRAN library. The models shall be run on train set and tested on test set to identify the most apt model.

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ------------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ---------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
## Loading required package: rpart
```

```
## Loading required package: caret
```

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

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

## Loading required package: PRROC

## Warning: package 'PRROC' was built under R version 4.0.3

## Loading required package: ROCR

## Warning: package 'ROCR' was built under R version 4.0.3

## Loading required package: xgboost

## Warning: package 'xgboost' was built under R version 4.0.3

##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice

## Loading required package: corrr

## Warning: package 'corrr' was built under R version 4.0.3

## Parsed with column specification:
## cols(
##   id = col_double(),
##   Gender = col_character(),
##   Age = col_double(),
##   Driving_License = col_double(),
##   Region_Code = col_double(),
##   Previously_Insured = col_double(),
##   Vehicle_Age = col_character(),
##   Vehicle_Damage = col_character(),
##   Annual_Premium = col_double(),
##   Policy_Sales_Channel = col_double(),
##   Vintage = col_double(),
##   Response = col_double()
## )

## tibble [381,109 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ id                  : num [1:381109] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Gender              : chr [1:381109] "Male" "Male" "Male" "Male" ...
##  $ Age                 : num [1:381109] 44 76 47 21 29 24 23 56 24 32 ...
##  $ Driving_License     : num [1:381109] 1 1 1 1 1 1 1 1 1 1 ...
##  $ Region_Code         : num [1:381109] 28 3 28 11 41 33 11 28 3 6 ...
##  $ Previously_Insured  : num [1:381109] 0 0 0 1 1 0 0 0 1 1 ...
##  $ Vehicle_Age         : chr [1:381109] "> 2 Years" "1-2 Year" "> 2 Years" "< 1 Year" ...
##  $ Vehicle_Damage      : chr [1:381109] "Yes" "No" "Yes" "No" ...
##  $ Annual_Premium      : num [1:381109] 40454 33536 38294 28619 27496 ...
##  $ Policy_Sales_Channel: num [1:381109] 26 26 26 152 152 160 152 26 152 152 ...
##  $ Vintage             : num [1:381109] 217 183 27 203 39 176 249 72 28 80 ...
##  $ Response            : num [1:381109] 1 0 1 0 0 0 0 1 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   id = col_double(),
##   ..   Gender = col_character(),
##   ..   Age = col_double(),
##   ..   Driving_License = col_double(),
##   ..   Region_Code = col_double(),
##   ..   Previously_Insured = col_double(),
##   ..   Vehicle_Age = col_character(),
##   ..   Vehicle_Damage = col_character(),
##   ..   Annual_Premium = col_double(),
##   ..   Policy_Sales_Channel = col_double(),
##   ..   Vintage = col_double(),
##   ..   Response = col_double()
##   .. )

## Warning: Unknown or uninitialised column: ‘gender‘.

## Warning: Unknown or uninitialised column: ‘damage‘.

## Warning: Unknown or uninitialised column: ‘veh_age‘.

## tibble [381,109 x 8] (S3: tbl_df/tbl/data.frame)
##  $ gender              : num [1:381109] 1 1 1 1 0 0 1 0 0 0 ...
##  $ Age                 : num [1:381109] 44 76 47 21 29 24 23 56 24 32 ...
##  $ Previously_Insured  : num [1:381109] 0 0 0 1 1 0 0 0 1 1 ...
##  $ veh_age             : num [1:381109] 2 1 2 0 0 0 0 1 0 0 ...
##  $ damage              : num [1:381109] 1 0 1 0 0 1 1 1 0 0 ...
##  $ Annual_Premium      : num [1:381109] 40454 33536 38294 28619 27496 ...
##  $ Policy_Sales_Channel: num [1:381109] 26 26 26 152 152 160 152 26 152 152 ...
##  $ Response            : num [1:381109] 1 0 1 0 0 0 0 1 0 0 ...

## Warning in set.seed(1234, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used

## Warning: The ‘i‘ argument of ‘‘[‘()‘ can’t be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_warnings()‘ to see where this warning was generated.
```

2.2 2.2 Data Exploration

We first need to understand the structure of the data, the distribution of responses and the relationship of the predictors to help build a better model. The data has 381,109 observations with 8 variables which have been converted to numeric.

We are aiming to predict the Response based on other variables. The response is a two factor numeric variable with a fair degree of imbalance as the proportion of affirmative responses are about 12%.

We thus try to find some variables which can have higher impact on the response being Yes. While there is not much variation in responses of males or females, the absence of insurance previously or having incurred a damage previously results in higher response as Yes.

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

```
## # A tibble: 7 x 2
##   term                 Response
##   <chr>                   <dbl>
## 1 gender                 0.0524
## 2 Age                    0.111
## 3 Previously_Insured    -0.341
## 4 veh_age                0.222
## 5 damage                 0.354
## 6 Annual_Premium         0.0226
## 7 Policy_Sales_Channel  -0.139
```

```
## [1] 0.1224814
```

```
## [1] 0.122891
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 2
##   gender Response_prop
##    <dbl>         <dbl>
## 1      0         0.104
## 2      1         0.139
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 2
##   Previously_Insured Response_prop
##                <dbl>         <dbl>
## 1                  0      0.225
## 2                  1      0.000937
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 2
##   damage Response_prop
##    <dbl>         <dbl>
## 1      0       0.00534
## 2      1       0.238
```

2.3 2.3 Data Cleaning

Having more features increases the complexity and computing requirement, thus we could eliminate data features not needed.

```
train_set <- train_set %>% select(Age, Previously_Insured, veh_age, damage, Policy_Sales_Channel, Respon

test_set  <- test_set  %>% select(Age, Previously_Insured, veh_age, damage, Policy_Sales_Channel, Respon
```

2.4 2.4 Data Modelling

2.4.1 2.4.1 Random Prediction

This is essentially based on assigning responses randomly through sampling. As we shall see in the next section that it will give a poor accuracy and AUC.

2.4.2 2.4.2 Factor based Linear Models

The next improvisation over random prediction is linear models - the simplest one assigns the responses based on the correlation between the features. As we know there are two critical features - Previously Insured which has a relatively decent negative correlation to response and Damage Suffered which has a decent positive correlation. We start by assigning response based on these features separately and then aim to improve the accuracy and AUC by combining both these together.

These models provide poor accuracy as we shall see thus we first need to improve the accuracy followed by AUC.

2.4.3 2.4.3 Logistic Regression Models

Logistic regression models are used where the dependent variable is categorical which in our case is a two factor categorical outcome. We first use only two most relevant features - previous insurance and damage, and then try to predic the outcome based on all features.

As we shall see that while this gives high accuracy, it performs poorly on specificity and sensitivity with AUC being 0.5.

2.4.4 2.4.4 Boosting Models

Boosting models are widely used machine learning tool for improving the accuracy of classification models. It starts by fitting an initial model (e.g. a tree or linear regression) to the data. Then a second model is built that focuses on accurately predicting the cases where the first model performs poorly. The combination of these two models is expected to be better than either model alone. Then you repeat this process of boosting many times. Each successive model attempts to correct for the shortcomings of the combined boosted ensemble of all previous models.

There are varied boosting algoithms the most common being gradient boosting - gradient boosting is called so because target outcomes for each case are set based on the gradient of the error (which can be specified by users based on evaluation metric) with respect to the prediction. Each new model takes a step in the direction that minimizes prediction error, in the space of possible predictions for each training case.

We have used XGBoost package to fit the gradient boosting model due to its ability to offer better accuracy with speed.

3 3 Results

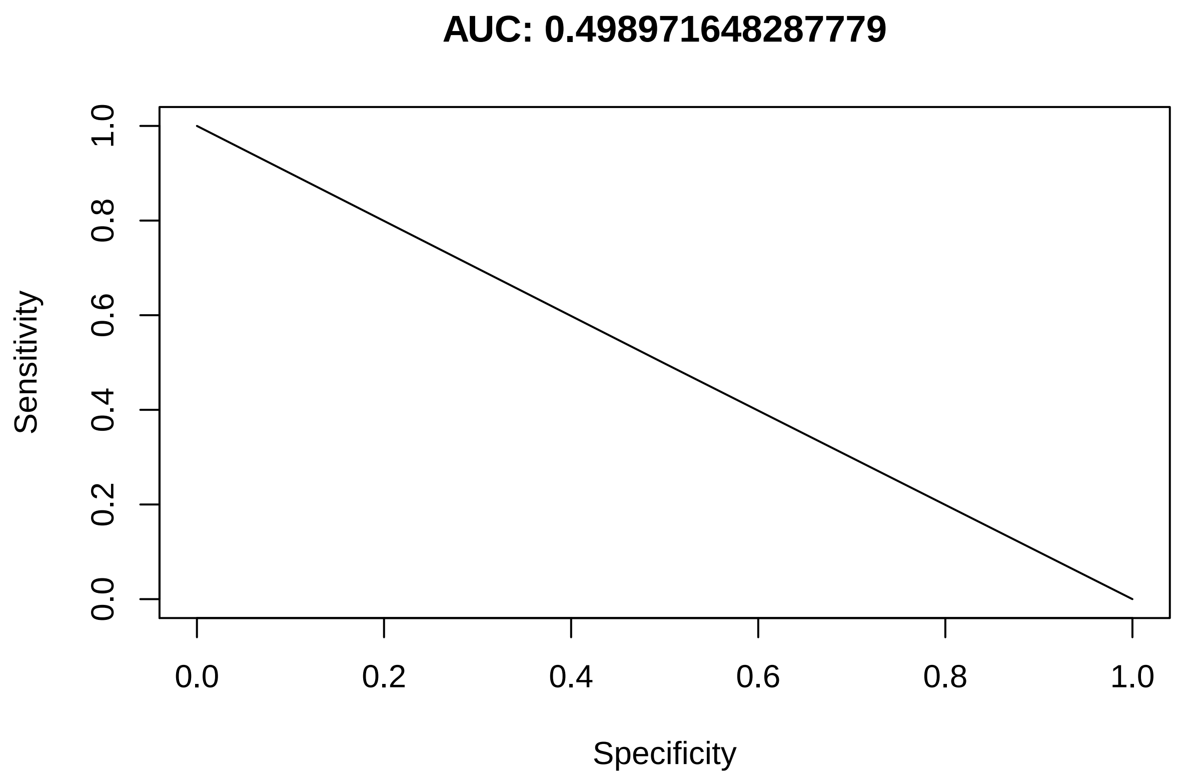
The models are evaluated using the overall accuracy and AUC. AUC ie area under ROC curve is a plot of the false positive rate ie Specificity (x-axis) versus the true positive rate ie Sensitivity (y-axis) for a number of different threshold values between 0.0 and 1.0.

3.1 3.1 Random Prediction

This is essentially based on assigning responses randomly. As we see that it gives a poor accuracy and AUC of about 50%.

```
## Warning in set.seed(3, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 33472  4709
##          1 33383  4658
##
##                Accuracy : 0.5002
##                  95% CI : (0.4967, 0.5038)
##     No Information Rate : 0.8771
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -9e-04
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5007
##             Specificity : 0.4973
##          Pos Pred Value : 0.8767
##          Neg Pred Value : 0.1224
##              Prevalence : 0.8771
##          Detection Rate : 0.4391
##    Detection Prevalence : 0.5009
##       Balanced Accuracy : 0.4990
##
##        'Positive' Class : 0
##
```



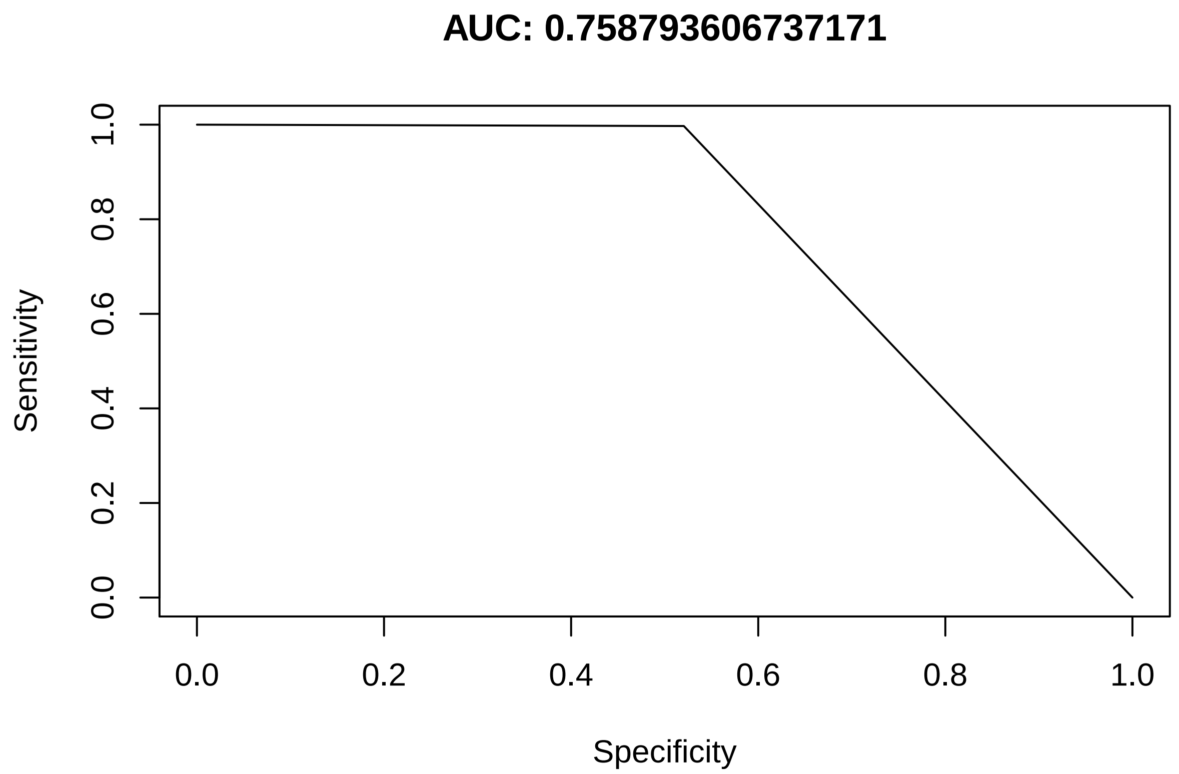
```
## # A tibble: 1 x 3
##   Method            Accuracy   AUC
##   <chr>                <dbl> <dbl>
## 1 random_prediction    0.500 0.499
```

3.2 3.2 Factor Based Linear Predictive Models

We tried predicting the outcome based on two critical features - previously insured which has negative correlation to response and damage suffered which has positive correlation. This will help us examine the accuracy of the prediction along with AUC (that captures specificity and sensitivity as well).

We improve the accuracy and AUC by combining both these together - this leads to a further improvement in accuracy and AUC to 64% and 78% respectively which is better than standalone accuracy of both the models. However, these are simplistic approaches and lets try regression techniques to evaluate the impact.

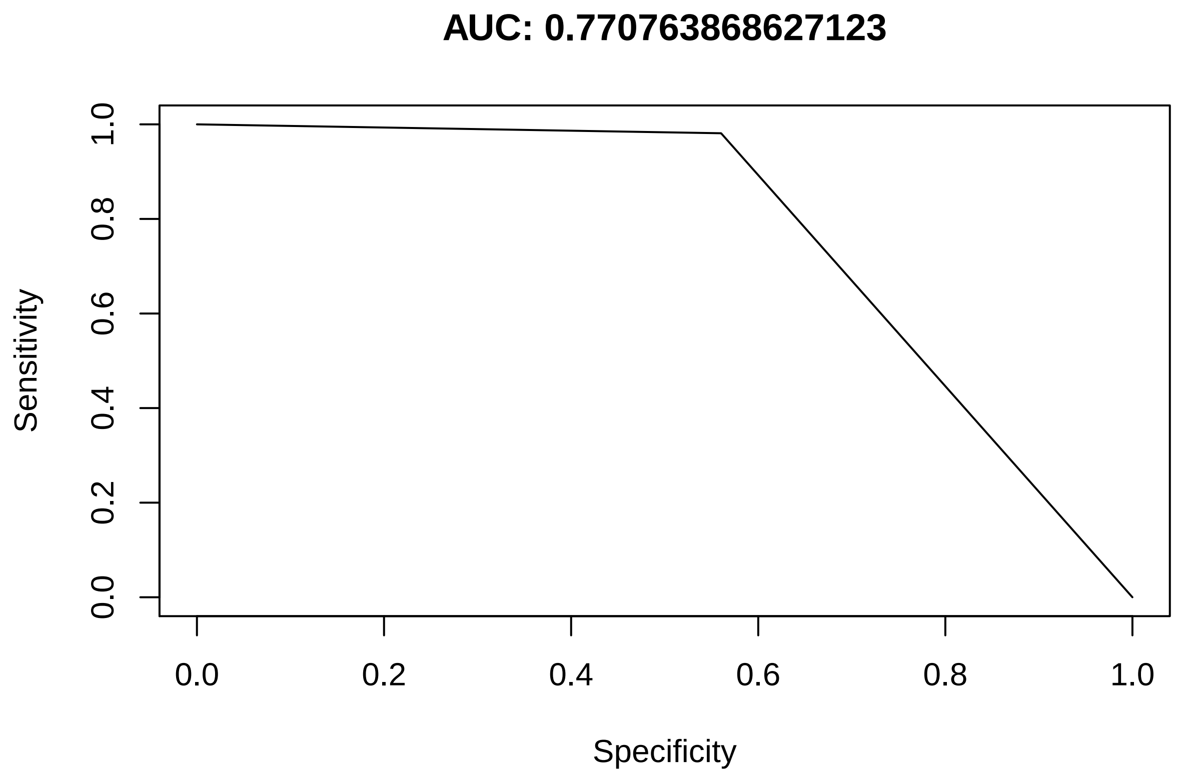
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 34796    27
##          1 32059  9340
##
##                Accuracy : 0.579
##                  95% CI : (0.5755, 0.5826)
##     No Information Rate : 0.8771
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2095
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5205
##             Specificity : 0.9971
##          Pos Pred Value : 0.9992
##          Neg Pred Value : 0.2256
##              Prevalence : 0.8771
##          Detection Rate : 0.4565
##    Detection Prevalence : 0.4569
##       Balanced Accuracy : 0.7588
##
##        'Positive' Class : 0
##
```



```
## # A tibble: 2 x 3
##   Method              Accuracy   AUC
##   <chr>                  <dbl> <dbl>
## 1 "random_prediction"    0.500 0.499
## 2 " Prev_Ins_model "     0.579 0.759
```

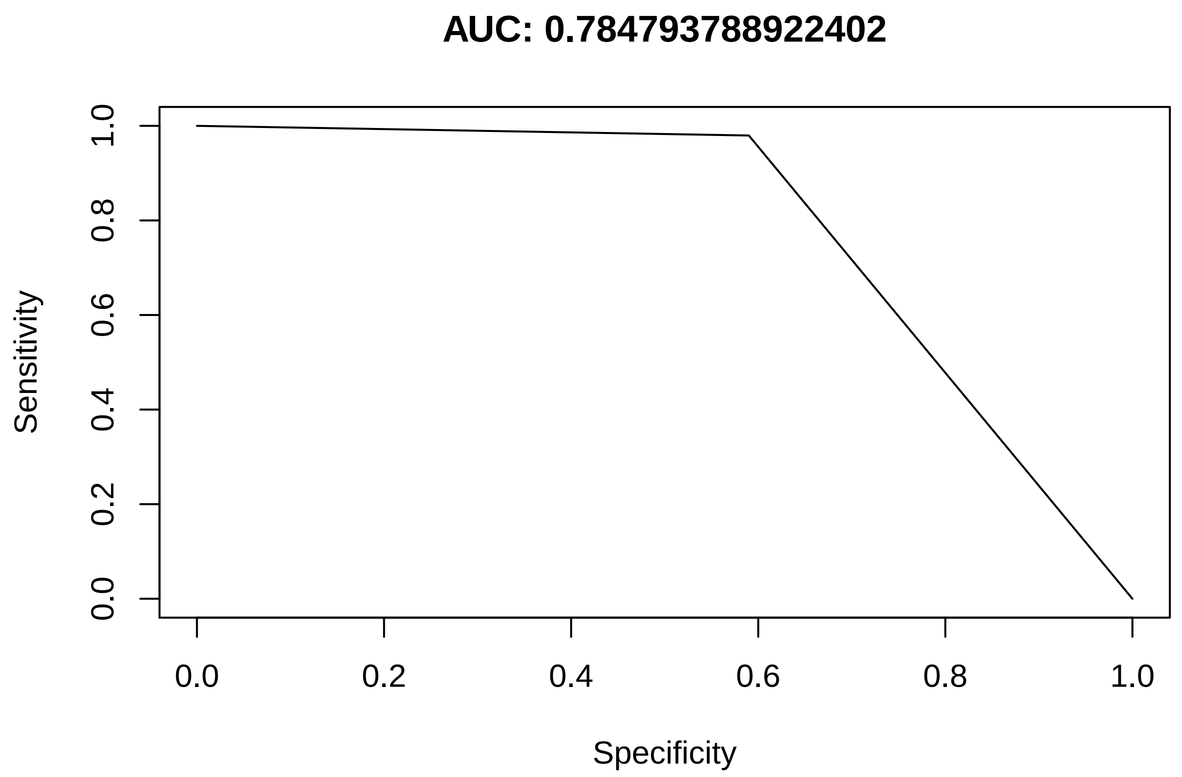
```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction     0     1
##          0 37460   176
##          1 29395  9191
##
##                Accuracy : 0.612
##                  95% CI : (0.6086, 0.6155)
##     No Information Rate : 0.8771
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2313
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5603
##             Specificity : 0.9812
##          Pos Pred Value : 0.9953
##          Neg Pred Value : 0.2382
##              Prevalence : 0.8771
##          Detection Rate : 0.4915
##    Detection Prevalence : 0.4938
##       Balanced Accuracy : 0.7708
##
##        'Positive' Class : 0
##
```



```
## # A tibble: 3 x 3
##   Method              Accuracy   AUC
##   <chr>                  <dbl> <dbl>
## 1 "random_prediction"    0.500 0.499
## 2 " Prev_Ins_model "     0.579 0.759
## 3 " Damage_model "       0.612 0.771
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 39443   191
##          1 27412  9176
##
##                Accuracy : 0.6379
##                  95% CI : (0.6344, 0.6413)
##     No Information Rate : 0.8771
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2532
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5900
##             Specificity : 0.9796
##          Pos Pred Value : 0.9952
##          Neg Pred Value : 0.2508
##              Prevalence : 0.8771
##          Detection Rate : 0.5175
##    Detection Prevalence : 0.5200
##       Balanced Accuracy : 0.7848
##
##        'Positive' Class : 0
##
```



```
## # A tibble: 4 x 3
##   Method                          Accuracy   AUC
##   <chr>                              <dbl> <dbl>
## 1 "random_prediction"                0.500 0.499
## 2 " Prev_Ins_model "                 0.579 0.759
## 3 " Damage_model "                   0.612 0.771
## 4 " Damage_Insurance_combine "       0.638 0.785
```

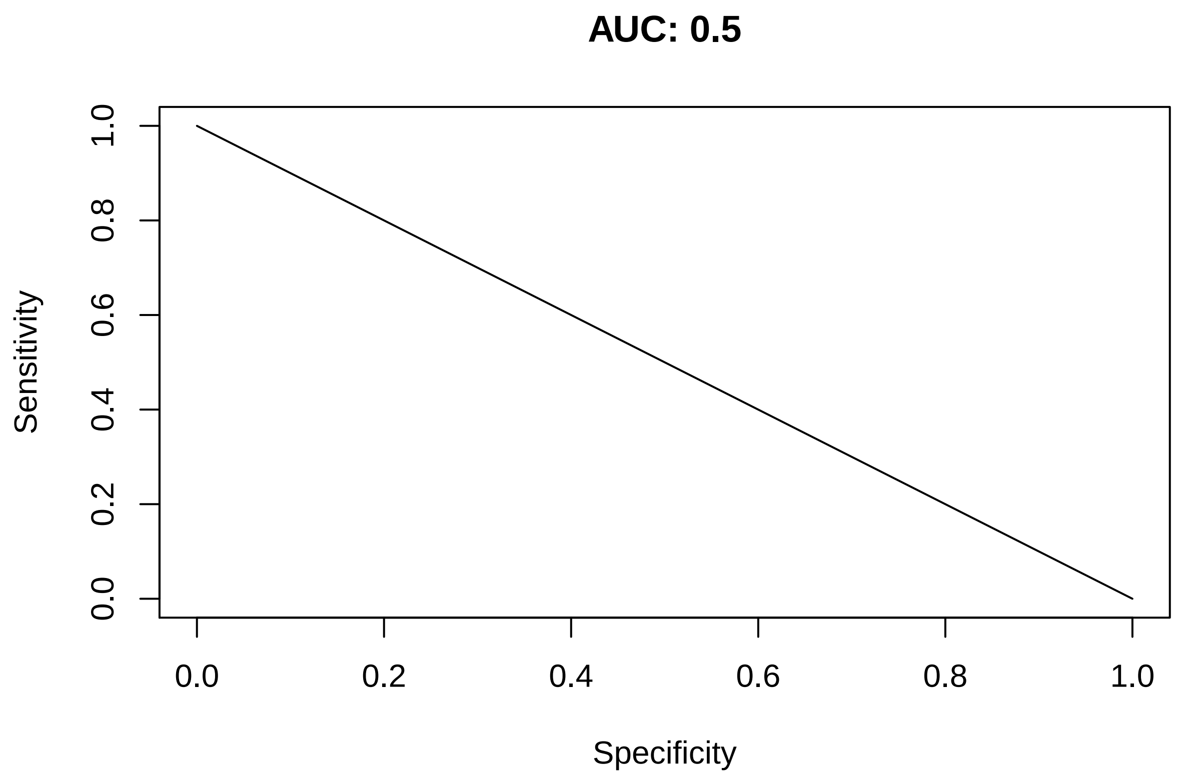
3.3 3.3 Logistic Regression Models

Logistic regression models are used where the dependent variable is categorical which in our case is a two factor categorical outcome. We first use only two most relevant features - previous insurance and damage. This while gives high accuracy, performs poorly on specificity and sensitivity with AUC being 0.5.

We then try to see if modeling the outcome on all variables improves the AUC but to no benefit. It remains 0.5 whereas we are targetting AUC of >0.80.

```
## Warning in set.seed(3, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## Warning in confusionMatrix.default(data = factor(glm_prediction), reference =
## factor(test_set$Response)): Levels are not in the same order for reference and
## data. Refactoring data to match.
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 66855  9367
##          1     0     0
##
##                Accuracy : 0.8771
##                  95% CI : (0.8748, 0.8794)
##     No Information Rate : 0.8771
##     P-Value [Acc > NIR] : 0.5028
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.8771
##          Neg Pred Value :    NaN
##              Prevalence : 0.8771
##          Detection Rate : 0.8771
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

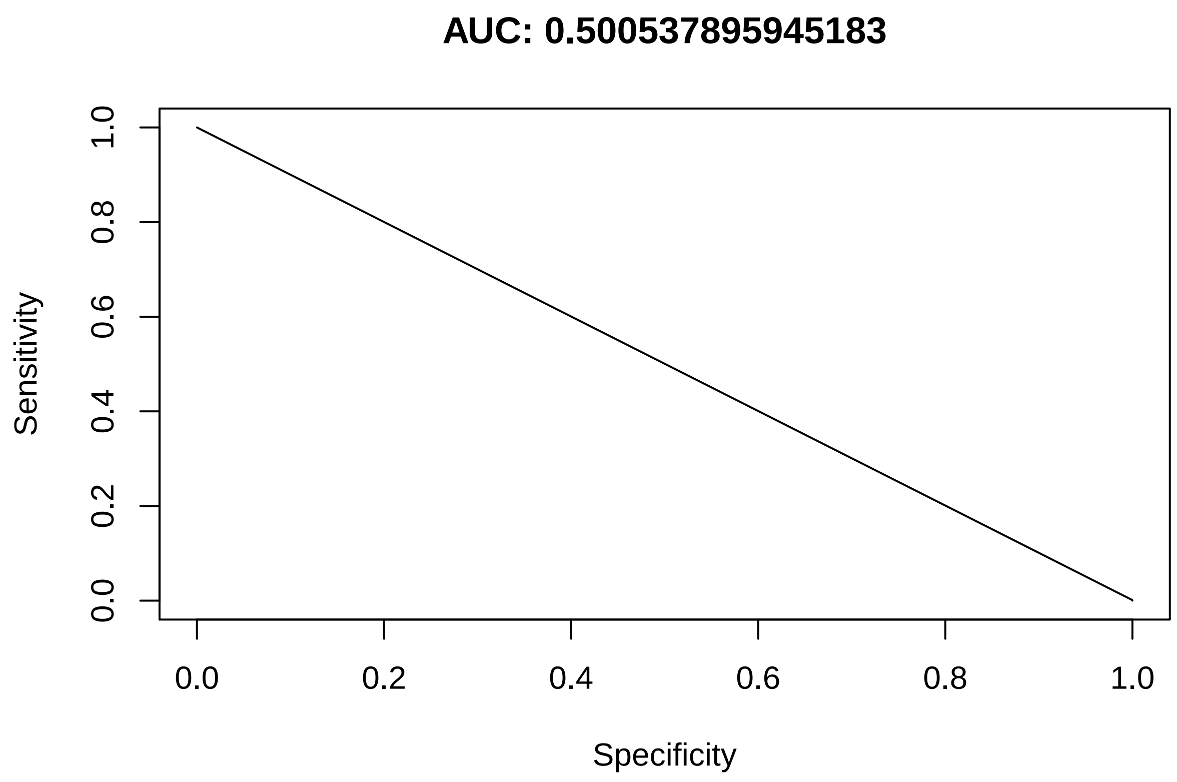


```
## Warning in confusionMatrix.default(data = factor(glm_prediction), reference =
## factor(test_set$Response)): Levels are not in the same order for reference and
## data. Refactoring data to match.
```

```
## # A tibble: 5 x 3
##   Method                       Accuracy   AUC
##   <chr>                           <dbl> <dbl>
## 1 "random_prediction"             0.500 0.499
## 2 " Prev_Ins_model "              0.579 0.759
## 3 " Damage_model "                0.612 0.771
## 4 " Damage_Insurance_combine "    0.638 0.785
## 5 " prediction_glm "              0.877 0.5
```

```
## Warning in set.seed(3, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 66827  9353
##          1    28    14
##
##                Accuracy : 0.8769
##                  95% CI : (0.8746, 0.8792)
##     No Information Rate : 0.8771
##     P-Value [Acc > NIR] : 0.5641
##
##                   Kappa : 0.0019
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.999581
##             Specificity : 0.001495
##          Pos Pred Value : 0.877225
##          Neg Pred Value : 0.333333
##              Prevalence : 0.877109
##          Detection Rate : 0.876742
##    Detection Prevalence : 0.999449
##       Balanced Accuracy : 0.500538
##
##        'Positive' Class : 0
##
```



```
## # A tibble: 6 x 3
##   Method                       Accuracy   AUC
##   <chr>                           <dbl> <dbl>
## 1 "random_prediction"             0.500 0.499
## 2 " Prev_Ins_model "              0.579 0.759
## 3 " Damage_model "                0.612 0.771
## 4 " Damage_Insurance_combine "    0.638 0.785
## 5 " prediction_glm "             0.877 0.5
## 6 " prediction_glm_all "         0.877 0.501
```

3.4 3.4 XG Boost Model

XGBoost is a high performing model which gets good results.XGBoost refers to Extreme Gradient Boosting, which is an efficient implementation of gradient boosting framework for tree learning algorithms.

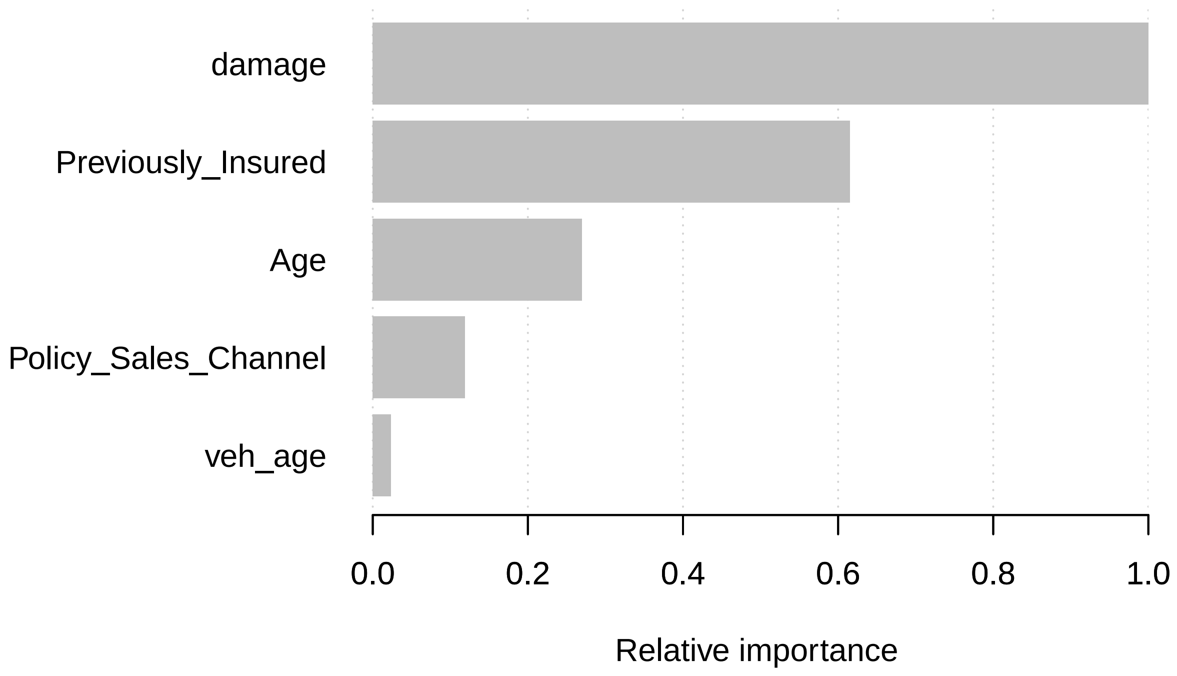
I tried the decision trees (using rpart) and random forest models separately, which took very long time without any better outcome than the glm models shown above. Thus these are not included as part of the project report and instead a faster XGBoost package is counted which can do parallel computation at a 10x faster speed.

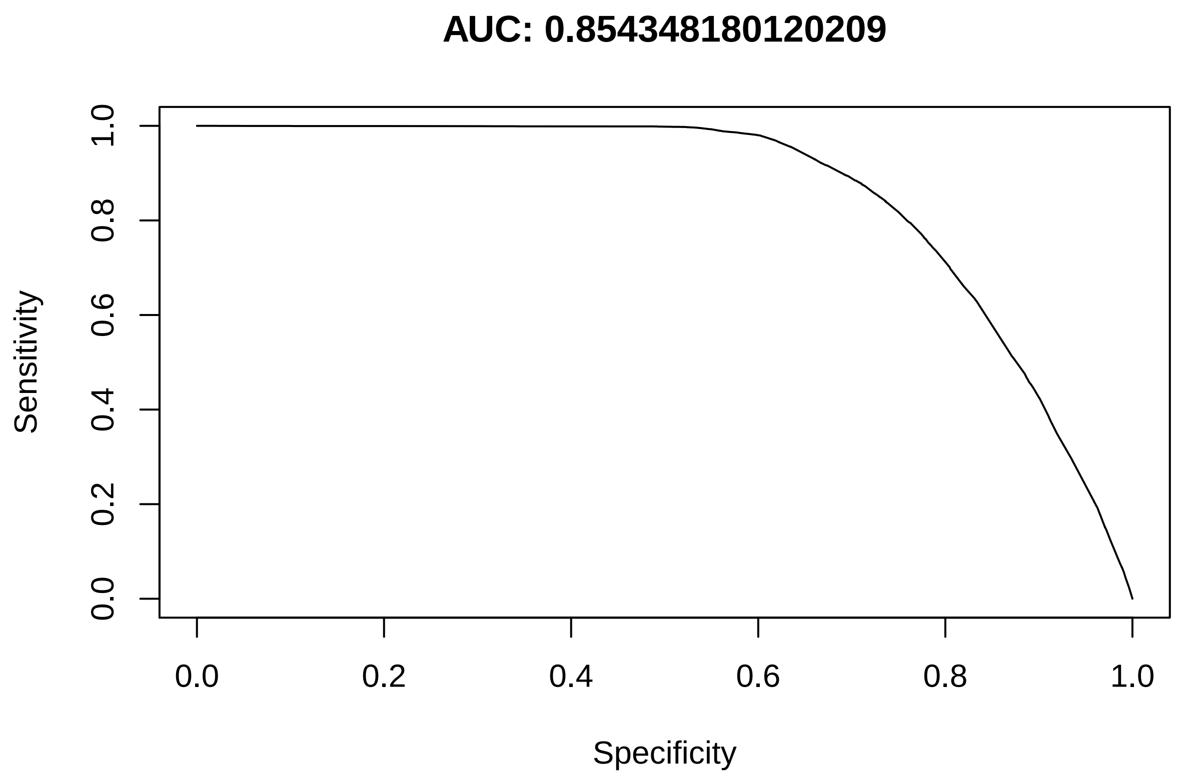
As explained earlier, it does repetitive modelling building upon the error of previous model until the best outcome on evaluation metric is achieved. In this case, we specified maximising accuracy or minimising the test error as our evaluation criteria to get the best fit model. This was then used to review the AUC.

At an accuracy of about 0.87, it improved the AUC to 0.85 which we aimed to achieve (>0.80).

```
## Warning in set.seed(1234, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used

## [1]  test-error:0.122891
## Will train until test_error hasn't improved in 50 rounds.
##
## Stopping. Best iteration:
## [25] test-error:0.122799
```





```
## # A tibble: 7 x 3
##   Method                          Accuracy   AUC
##   <chr>                              <dbl> <dbl>
## 1 "random_prediction"                0.500 0.499
## 2 " Prev_Ins_model "                 0.579 0.759
## 3 " Damage_model "                   0.612 0.771
## 4 " Damage_Insurance_combine "       0.638 0.785
## 5 " prediction_glm "                 0.877 0.5
## 6 " prediction_glm_all "             0.877 0.501
## 7 " predictions_xgb "                0.877 0.854
```

4 4 Conclusion

There are various pre-steps before running the models which are extremely important for the models to show correct results. These include data preparation, data exploration and data analysis.

Further its important to start building simple models before getting to more complex ones as it helps understand the improvements brought about by using a stepped-up approach. We gather more insights into the data as we start with simple models.

In our case we started with a random model that predicts the response through random sampling. At this stage we started adding other relevant features to our prediction model such as previously insured, damage etc and then With logistic regression aim to predict the response. While the logistic regression model increased the overall accuracy, it lead to high degree of false negatives and false positives. Thus, we implemented the XG Boost package to help improve the performance across the parameters of specificity and sensitivity.

4.1 4.1 Limitations

One of the limitation of our classification model is that we have not used any technique to improve to balance within dataset.The outcome of model is impacted by the prevalence where we have higher proportion of negative responses and thus this problem can be addressed through oversampling by introducing higher positive values and undersampling by eliminating certain negative values. For over-sampling, we have to generate additional data points of smaller class in the training set after splitting.

Another aspect is that we can use ensemble technique to improve the performance of the classification model. Ensemble tries to combine various machine learning techniques to provide the best outcome. The three most popular methods for combining the predictions from different models are:

Bagging - Building multiple models (typically of the same type) from different subsamples of the training dataset.

Boosting - Building multiple models (typically of the same type) each of which learns to fix the prediction errors of a prior model in the chain.

Stacking - Building multiple models (typically of differing types) and supervisor model that learns how to best combine the predictions of the primary models.

4.2 4.2 Way Forward for Future

Some of the limitations discussed above can get addressed by building additional algorithms using packages available in R.

The most common technique is SMOTE - Synthetic Minority Over-Sampling Technique which we have not implemented. This can be done achieved by using SMOTE package in CRAN.

While we have used the boosting technique in our last model using XGBoosting method and also the bagging technique (using decision tree though not included here), we can try to implement stacking of algorithms using caretEnsemble package. When we combine the predictions of different models using stacking, it is desirable that the predictions made by the sub-models have low correlation. This would suggest that the models are skillful but in different ways, allowing a new classifier to figure out how to get the best from each model for an improved score.

5 Resource References

1. https://courses.edx.org/dashboard/programs/3c32e3e0-b6fe-4ee4-bd4f-210c6339e074/
2. https://rafalab.github.io/dsbook/
3. https://cran.r-project.org/web/packages/xgboost/xgboost.pdf
4. https://statinfer.com/203-4-3-roc-and-auc/
5. https://cran.r-project.org/web/packages/ROCR/ROCR.pdf