ALY6020 Predictive Analytics

Northeastern University

Fall 2020

Instructor: Ajit Appari

Group Assignment – Final Deliverables

Dec 8th, 2020

Xinyu Xu, Wenjia Zhang, Zimin Cai, Akash Sharma

Northeastern University

Table of Contents

Executive Summary

There is a company that provides health insurance wants to find a way to predict whether an existing customer will be interested in vehicle insurance. Based on this circumstance, we would like to research what aspects trigger customers' purchase willingness and what factors can be used to target potential customers more accurately. Through working on these factors, we established and optimized a variety of models, the approaches we used include the K-nearest neighbors (KNN) method, XGBoost (decision tree-based method), Regression method, and Naive Bayes method.

Among the modeling prediction results of all the above methods, KNN model has the highest accuracy (when k-value=13, accuracy is 0.8761), and the logistic regression model takes the shortest time (8-9 seconds on average). However, after discussion, our group agreed that it was not reasonable to consider the quality of the model solely from only one perspective. Given that other models all had some deficiencies in specificity, precision, or F1-score, we finally chose the Naive Bayes method with the best comprehensive performance as our prediction model and recommended it to this insurance company.

Project Objective

By building predictive models and optimizations, help the insurance company use the most suitable model to find people among existing health insurance customers who might be interested in vehicle insurance.

Introduction

We cannot prevent accidents from happening in our life, but having insurance can at least provide us with some financial protection in case of accidents. The contemporary insurance industry includes many types of insurance. Our daily needs mainly include life insurance, health insurance, long-term disability, renters, property insurance, commercial insurance, etc. Among property insurances, vehicle insurance plays a great part. With the continuous rise of people's living standards, their risk awareness is gradually strengthened, and the competition in the insurance industry is also more and more incentive. How to stand out from the competition and attract more customers is one of the important goals of many insurance companies.

According to our basic understanding and interest in the insurance industry, we selected our project to help a company that provides health insurance predict which of its existing customers would be interested in buying vehicle insurance through modeling. The more specific research questions are: What factors will impact the will of people to purchase vehicle insurance? What factors can be used to target potential customers more accurately？ Based on the study of these factors, we will establish and optimize a variety of models, and provide the optimal one to help the insurance company identify the most potential customers. By using this approach, the insurance company could put forward a precision marketing strategy at a low operating cost, gain a higher insurance purchase rate and further increase its revenue effectively.

The analytical tool we chose is RStudio and the datasets our group use are obtained from Kaggle's website. There are two files in total, a train set with 381,109 observations and 12 variables, along with a test set with 127,037 observations and 11 variables (it does not have one variable called "Response"). The data covered customers' gender, vehicle age, and vintage, etc. which can be used as independent variables in the prediction model, and the dependent variable is the response of customers. The specific variables are all described as shown in the table below:

Variable	**Definition**
id	Unique ID for the customer
Gender	Gender of the customer
Age	Age of the customer
Driving_License	0 : Customer does not have DL, 1 : Customer already has DL

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 customer needs to pay as premium in the year
Policy*Sales*Channel	Anonymized Code for the channel of outreaching to the customer ie. Different Agents, Over Mail, Over Phone, In Person, etc.
Vintage	Number of Days, Customer has been associated with the company
Response	1 : Customer is interested, 0 : Customer is not interested

Since the dependent variable is in binary format (1 is positive response, 0 is negative response), when using these data sets for prediction modeling, we first determine that prediction models such as KNN, Logistic regression, classification tree are all relatively suitable. Therefore, we start with these methods and try one by one. As we continued learning and the project progressed, we experimented with decision tree methods (XGBoost), Naive Bayes methods, etc, in an attempt to find optimization or alternative solutions. We hope to finally compare the results to determine the prediction model that best suited the current situation of the insurance company.

Implementation

The dataset of our topic has been split into train and test set in the Kaggle and it contains int/num, factor values. Most importantly, since the test dataset has no index column, we can only process the models of supervised learning by splitting the train dataset as as sub_train(80%) and sub_test(20%) to test the model accuracy. The analysis includes two parts, Data processing and Data modeling.

Part 1 Data Processing

Data Description

First, we load all the packages and datasets we needed for processing and modeling. Then, we checked the missing value for the data set. Fortunately, there are no missing value in our datasets.

In data description, by the result charts below.

```
> ### check data
> summary(train)
       id             Gender               Age        Driving_License    Region_Code   
 Min.   :     1   Length:381109      Min.   :20.00   Min.   :0.0000   Min.   : 0.00  
 1st Qu.: 95278   Class :character   1st Qu.:25.00   1st Qu.:1.0000   1st Qu.:15.00  
 Median :190555   Mode  :character   Median :36.00   Median :1.0000   Median :28.00  
 Mean   :190555                      Mean   :38.82   Mean   :0.9979   Mean   :26.39  
 3rd Qu.:285832                      3rd Qu.:49.00   3rd Qu.:1.0000   3rd Qu.:35.00  
 Max.   :381109                      Max.   :85.00   Max.   :1.0000   Max.   :52.00  
 Previously_Insured Vehicle_Age        Vehicle_Damage     Annual_Premium   Policy_Sales_Channel
 Min.   :0.0000     Length:381109      Length:381109      Min.   :  2630   Min.   :  1         
 1st Qu.:0.0000     Class :character   Class :character   1st Qu.: 24405   1st Qu.: 29         
 Median :0.0000     Mode  :character   Mode  :character   Median : 31669   Median :133         
 Mean   :0.4582                                           Mean   : 30564   Mean   :112         
 3rd Qu.:1.0000                                           3rd Qu.: 39400   3rd Qu.:152         
 Max.   :1.0000                                           Max.   :540165   Max.   :163         
    Vintage         Response     
 Min.   : 10.0   Min.   :0.0000  
 1st Qu.: 82.0   1st Qu.:0.0000  
 Median :154.0   Median :0.0000  
 Mean   :154.3   Mean   :0.1226  
 3rd Qu.:227.0   3rd Qu.:0.0000  
 Max.   :299.0   Max.   :1.0000  
```

```
> summary(test)
       id             Gender               Age        Driving_License    Region_Code   
 Min.   :381110   Length:127037      Min.   :20.00   Min.   :0.0000   Min.   : 0.00  
 1st Qu.:412869   Class :character   1st Qu.:25.00   1st Qu.:1.0000   1st Qu.:15.00  
 Median :444628   Mode  :character   Median :36.00   Median :1.0000   Median :28.00  
 Mean   :444628                      Mean   :38.77   Mean   :0.9981   Mean   :26.46  
 3rd Qu.:476387                      3rd Qu.:49.00   3rd Qu.:1.0000   3rd Qu.:35.00  
 Max.   :508146                      Max.   :85.00   Max.   :1.0000   Max.   :52.00  
 Previously_Insured Vehicle_Age        Vehicle_Damage     Annual_Premium   Policy_Sales_Channel
 Min.   :0.00       Length:127037      Length:127037      Min.   :  2630   Min.   :  1.0       
 1st Qu.:0.00       Class :character   Class :character   1st Qu.: 24325   1st Qu.: 26.0       
 Median :0.00       Mode  :character   Mode  :character   Median : 31642   Median :135.0       
 Mean   :0.46                                             Mean   : 30525   Mean   :111.8       
 3rd Qu.:1.00                                             3rd Qu.: 39408   3rd Qu.:152.0       
 Max.   :1.00                                             Max.   :472042   Max.   :163.0       
    Vintage     
 Min.   : 10.0  
 1st Qu.: 82.0  
 Median :154.0  
 Mean   :154.3  
 3rd Qu.:227.0  
 Max.   :299.0  
```

```
> str(train)
'data.frame':	381109 obs. of  12 variables:
 $ id                  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Gender              : chr  "Male" "Male" "Male" "Male" ...
 $ Age                 : int  44 76 47 21 29 24 23 56 24 32 ...
 $ Driving_License     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Region_Code         : num  28 3 28 11 41 33 11 28 3 6 ...
 $ Previously_Insured  : int  0 0 0 1 1 0 0 0 1 1 ...
 $ Vehicle_Age         : chr  "> 2 Years" "1-2 Year" "> 2 Years" "< 1 Year" ...
 $ Vehicle_Damage      : chr  "Yes" "No" "Yes" "No" ...
 $ Annual_Premium      : num  40454 33536 38294 28619 27496 ...
 $ Policy_Sales_Channel: num  26 26 26 152 152 160 152 26 152 152 ...
 $ Vintage             : int  217 183 27 203 39 176 249 72 28 80 ...
 $ Response            : int  1 0 1 0 0 0 0 1 0 0 ...
```

Data formatting

Then order to process the model, we also need to format datasets for training and testing. First, we transferred the factor values, Gender and Vehicle Damage Vehicle Age, into Boolean value (0 and 1) by using “factor” function to change their labels in both train and test sets. Then, we used the “mutate”, “case_when” and “dummy_cols” functions in column, Vehicle_Age, Region_Code and Policy_Sales_Channel, to create dummy values to replace the original columns. Therefore, after formatting, we got the name and format of columns as the chart below. Also, we did the same process with the test set and we got the same result as the train set.

In addition, as we mentioned above, the testing set has no label column so we split the train set as sub_train and sub_test for modeling. To get the same results each time running the models, we set the seed as label (12345) to remember the random sets we used for modeling each time. What’s more, since the trainset was split to train and test, we might get the high accuracy and low Kappa value in a model. For better testing, after selecting the best model for trainset, we will also process it into test set for prediction.

The charts below are the column name and data type after cleaning.

```
> colnames(train)
 [1] "id"                                "Gender"
 [3] "Age"                               "Driving_License"
 [5] "Previously_Insured"                "Vehicle_Damage"
 [7] "Annual_Premium"                    "Vintage"
 [9] "Response"                          "Vehicle_Age_< 1 Year"
[11] "Vehicle_Age_> 2 Years"             "Vehicle_Age_1-2 Year"
[13] "Region_Code_0_10"                  "Region_Code_10_20"
[15] "Region_Code_20_30"                 "Region_Code_30_40"
[17] "Region_Code_40_50"                 "Region_Code_equal&over_50"
[19] "Policy_Sales_Channel_0_15"         "Policy_Sales_Channel_15_30"
[21] "Policy_Sales_Channel_30_45"        "Policy_Sales_Channel_45_60"
[23] "Policy_Sales_Channel_60_75"        "Policy_Sales_Channel_75_90"
[25] "Policy_Sales_Channel_90_105"       "Policy_Sales_Channel_105_120"
[27] "Policy_Sales_Channel_120_135"      "Policy_Sales_Channel_135_150"
[29] "Policy_Sales_Channel_equal_over_150"
```

```
> Clean_test <- write.csv(test,file="/Users/icho/Desktop/MNIST-data/Clean_test.cs
> str(train)
'data.frame':	381109 obs. of  29 variables:
 $ id                              : num  1 2 3 4 5 6 7 8 9 10 ...
 $ Gender                          : num  1 1 1 1 0 0 1 0 0 0 ...
 $ Age                             : num  44 76 47 21 29 24 23 56 24 32 ...
 $ Driving_License                 : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Previously_Insured              : num  0 0 0 1 1 0 0 0 1 1 ...
 $ Vehicle_Damage                  : num  1 0 1 0 0 1 1 1 0 0 ...
 $ Annual_Premium                  : num  40454 33536 38294 28619 27496 ...
 $ Vintage                         : num  217 183 27 203 39 176 249 72 28 80 .
 $ Response                        : num  1 0 1 0 0 0 0 1 0 0 ...
 $ Vehicle_Age_< 1 Year            : num  0 0 0 1 1 1 1 0 1 1 ...
 $ Vehicle_Age_> 2 Years           : num  1 0 1 0 0 0 0 0 0 0 ...
 $ Vehicle_Age_1-2 Year            : num  0 1 0 0 0 0 0 1 0 0 ...
 $ Region_Code_0_10                : num  0 1 0 0 0 0 0 0 1 1 ...
 $ Region_Code_10_20               : num  0 0 0 1 0 0 1 0 0 0 ...
 $ Region_Code_20_30               : num  1 0 1 0 0 0 0 1 0 0 ...
 $ Region_Code_30_40               : num  0 0 0 0 0 1 0 0 0 0 ...
 $ Region_Code_40_50               : num  0 0 0 0 1 0 0 0 0 0 ...
 $ Region_Code_equal&over_50       : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Policy_Sales_Channel_0_15       : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Policy_Sales_Channel_15_30      : num  1 1 1 0 0 0 0 1 0 0 ...
 $ Policy_Sales_Channel_30_45      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Policy_Sales_Channel_45_60      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Policy_Sales_Channel_60_75      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Policy_Sales_Channel_75_90      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Policy_Sales_Channel_90_105     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Policy_Sales_Channel_105_120    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Policy_Sales_Channel_120_135    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Policy_Sales_Channel_135_150    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Policy_Sales_Channel_equal_over_150: num  0 0 0 1 1 1 1 0 1 1 ...
```

Part 2 Data Modeling

We set "Response" as our label and other factors as features. Also, since the ID of clients is unique and will not impact the "Response" thus we excluded it from modeling. The number of modeling techniques we adopted is four, which are K-nearest neighbors (KNN) method, XGBoost (decision tree-based method), Regression and Naive Bayes.

1、KNN model

The principle of KNN algorithm is that a sample belongs to a category if most of the k samples in the feature space that are most similar (that is, the most adjacent samples in the feature space) belong to that category. In other words, it classifies unknown variables into the most majority category in the sample according to majority-voting rules. Also, it is a relatively simple and commonly used classification algorithm. Considering that our data set is not large and there are not many variables, it is not difficult to find the most reasonable k-value after repeated attempts, so we first tried this method. In this model, we used the KNN function to do the training and predicting by selected K= 3,11,21,9,13,15.

2、Decision tree model with XGBoost

The basic algorithm of XGB is the decision tree model. Its rule is to reduce the loss function by adding new trees and fitting false residuals. The fitting process is the second-order Taylor expansion of the loss function, and the regular term is added to the objective function to find the optimal solution, so as to balance the decline of the objective function and the complexity of the model, and avoid overfitting. It uses CART(Classification and Regression Tree) tree instead of a normal decision tree. For classification problems, since the value corresponding to the leaf node of CART tree is an actual fraction rather than a determined category, this will facilitate the implementation of efficient optimization algorithm. So the biggest advantage of this model is high speed and high precision.

Among the many decision tree-based methods, XGBoost integrated learning technique has the advantages of supporting efficient parallel training, high speed, low memory consumption and high accuracy, so we also tried this modeling method. In this model, we first transformed the label of subtrain and subtest into numeric so as to process in the xgb.DMatrix function. After that, we set parameter of xgboost function, such as max depth =6, eta = 0.3 and objective= "binary: logistic".

3、Regression

In this model, we utilized two approaches to find out the related variables with Response, which are logistic regression and stepwise(backward) regression.

Logistic regression used to measure the relationship between the dependent variables (the label we want to predict) and one or more independent variables (features). Logistic regression is the preferred method for a binary task, which outputs a discrete binary between 0 and 1. It is a basic classification algorithm which is easy to utilize and has great performance so we think it is suitable to apply into our dataset, which all of the features are num after transforming. In this model, we applied the "glm" function to process the model.

Stepwise(backward) regression is a method of fitting regression models in which predictive variables are selected by an automated program. The rule of it is to consider adding or subtracting a variable from the set of explanatory variables at each step based on predefined criteria. In this regression, we adopted the backward elimination method. In this method, all variables are put into the model, and then try to remove one independent variable from the model to see if there is a significant change in the variation of the label variable in the whole model. After that, variables that reduce the amount of explanation least are eliminated. The process iterates until no independent variables meet the criteria for elimination. For this model, we processed it by using "step" function.

4、Naive Bayes

Naive Bayes applies Bayes' theorem and naive independence hypothesis. It is based on conditional probability and deduces the probability of an event from the given data. It has the advantage of being able to categorize quickly, and it also is able to deal with any number of predictors, whether they are continuous or classified. But it also has the disadvantage of assuming that the features are independent of each other. Here we use Naive Bayes to classify variables that influence to "Respond". The "naiveBayes" function is used for processing the function and we need to transfer our label "Response" as factor to run the model.

Data Analysis

Since our dataset's label column only consists of 0 and 1, we decided to focus on binary classification instead of multiple classification. Also, because the values in label column are imbalance, we need to introduce more metrics, precision, sensitivity(recall), F1-score and specificity (all related with confusion matrix), to evaluate the performance of the models except of Confusion Matrix and timer.

Confusion Matrix is a table that commonly used to describe the performance of a classification model (or “classifier”) over a set of test data whose true values are known. Take an example for understand how it works. (the picture is attached below). The rule of it is False Positive Rate = FP/actual no； True Negative Rate(sensitivity/recall) = TN/actual no ； Accuracy = (TP+TN)/total. Again, the test set has no label columns, so we split the train set into sub_train and sub_test for modeling. As train sets are segmented for training and testing, higher accuracy and lower Kappa values can be obtained in the model.

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

The sensitivity(recall) is the correct predictive proportion in all the positive results in the predictive model. The function of it is as same as, which is Sensitivity = TP/(FN+TP). The specificity is the ratio of the number of correctly classified negative numbers to the number of actually negative numbers, which is Specificity = TN/ (FP+TN). Precision is the proportion that been correctly classified as positive out of all the positive results. The function of it is Precision = TP/(TP+FP). F1-score is a harmonic mean of accuracy and recall rates and its function is F1 score = 2 * (precision * recall)/ (precision + recall).

1、K-nearest neighbors (KNN)

	1	2	3	4	5	6	
Knn=num	3	11	21	9	13	15	Average
Accuracy	0.8569	0.8756	0.8758	0.8751	0.8761	0.876	0.8761
time(mins)	13.74	17.95	14.91	13.88	13.92	14.18	14.7633333
Sensitivity	0.9601	0.99732	0.999925	0.9947	0.99859	0.999401	0.99167267
Specificity	0.1290901	0.02005	0.001267	0.03272	0.01309	0.006755	0.03382868
Precision	0.1290901	0.02005489	0.00126662	0.03272113	0.01308845	0.00675533	0.03382942
F1- score	0.2275809	0.03931911	0.00253004	0.06335807	0.02583825	0.01341995	0.06200772

（The yellow part is the best parameters we chose in this model and the red are the best value in each metrics）

In this model, we chose 6 values of K to observe the model. The chart above is the summary of the whole model. After comparing 6 metrics in the table, we can find out that when KNN=3, it has the best F1-score, Precision, specificity and timer. The details of the result and code are attached below.

K=3

```
> start_time_1 <- Sys.time()
> KNN_3<-knn(train = sub_train[,-c(1)],test = sub_test[,-c(1)],cl = sub_train$Response,k=3,prob =
> confusionMatrix(table(KNN_3,sub_test$Response)) # 0.8433
Confusion Matrix and Statistics


KNN_3     0     1
    0 64088  8251
    1  2660  1223

               Accuracy : 0.8569
                 95% CI : (0.8543, 0.8593)
    No Information Rate : 0.8757
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1195

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9601
            Specificity : 0.1291
         Pos Pred Value : 0.8859
         Neg Pred Value : 0.3150
             Prevalence : 0.8757
         Detection Rate : 0.8408
   Detection Prevalence : 0.9491
      Balanced Accuracy : 0.5446

       'Positive' Class : 0

> end_time_1 <- Sys.time()
> print(end_time_1 - start_time_1)
Time difference of 13.74344 mins
>
```

```
> precision_knn_3 <- 1223/(8251+1223)
> precision_knn_3
[1] 0.1290901
> F1_knn_3  <-2*(precision_knn_3*0.9601)/(precision_knn_3+0.9601)
> F1_knn_3
[1] 0.2275809
```

2、Extreme Gradient Boosting (XGBoost)

	1	2	3	4	5	6	
xgb=num	(0.3,6)	(0.1,6)	(0.05,6)	(0.1,8)	(0.1,4)	(0.5,6)	Average
Accuracy	0.8748	0.8757	0.8757	0.8754	0.8757	0.8738	0.8757
time(mins)	1.461	1.584	1.48	1.913	1.016	1.37	1.016
Sensitivity	0.99643	0.9997	0.99985	0.998816	0.9999251	0.99269	0.99269
Specificity	0.01805	0.001689	0.0004222	0.006122	0.0006333	0.3589	0.0004222
Precision	0.0180494	0.00168883	0.00042221	0.00612202	0.00063331	0.03588769	0.00042221
F1- score	0.03545653	0.00337197	0.00084406	0.01216945	0.00126582	0.0692711	0.00084406

（The group 6 in yellow is the best parameters we chose in this model and the red are the best value in each metrics）

By using Xgb function, we tried to adjust parameters, which are eta and max_depth, to compare the performance of the model under different setting. The chart above is the six metrics that we used to compare the model performance. Though when eta=4 and max_depth =4 the model has the best accuracy, sensitivity and timer, its other three metrics are not that promising compared with other parameters setting. Therefore, we finally chose when eta= 0.5 and max_depth = 6 is the best model in XGBoosting since it has the best specificity, precision and F1-score. The details of the results of group 6 are attached below.

Group 6 (0.5,6)

```
> start_time_2_5 <- Sys.time()
> para_6<-list(eta=0.5,max_depth=6,objective="binary:logistic",eval_metric="error")
> xgb_6<-xgboost(data = subtrain_matrix,nrounds = 120,params = para_6)
[1]	train-error:0.122130
[2]	train-error:0.122130
[3]	train-error:0.122130
[4]	train-error:0.122130
[5]	train-error:0.122130
[6]	train-error:0.122130
[7]	train-error:0.122212
```

```
> xgbpred_6<-predict(xgb_6,subtest_matrix,type="response")
> summary(xgbpred_6)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.0000017 0.0002521 0.0327916 0.1215679 0.2473932 0.8731293
> xgbpred_6<-as.integer(round(xgbpred_6))
> confusionMatrix(table(xgbpred_6,factor(sub_test$Response))) #0.8731
Confusion Matrix and Statistics


xgbpred_6     0     1
        0 66260  9134
        1   488   340

               Accuracy : 0.8738
                 95% CI : (0.8714, 0.8761)
    No Information Rate : 0.8757
    P-Value [Acc > NIR] : 0.9482

                  Kappa : 0.047

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.99269
            Specificity : 0.03589
         Pos Pred Value : 0.87885
         Neg Pred Value : 0.41063
             Prevalence : 0.87571
         Detection Rate : 0.86930
   Detection Prevalence : 0.98914
      Balanced Accuracy : 0.51429

       'Positive' Class : 0

> end_time_2_5 <- Sys.time()
> print(end_time_2_5 - start_time_2_5)
Time difference of 1.37024 mins
```

```
> precision_xgb_6 <- 340/(9134+340)
> precision_xgb_6
[1] 0.03588769
> F1_xgb_6  <-2*(precision_xgb_6*0.99269)/(precision_xgb_6+0.99269)
> F1_xgb_6
[1] 0.0692711
```

3、Regression

Regression	Logistic	backward
Accuracy	0.8756	0.8756
time	8.48sec	9.75 mins
Sensitivity	0.9998502	0.9998502
Specificity	0.0001056	0.0001056
Precision	0.00010555	0.00010555
F1- score	0.00021108	0.00021108

Through the chart above, we can know that the logistic and backward regression has the same values in every metrics in this table expect the backward took much longer time than logistic regression since it algorithm is more complicate than the Logistic one.

Logistic Regression

```
> options(warn=-1)
> start_time_3_0 <- Sys.time()
> lr_1<-glm(sub_train$Response~., family = binomial(link = logit),data=sub_train[,-c(1)])
> lrpred_1<-predict(lr_1,newdata = sub_test,type = "response")
> lrpred_1<-as.integer(round(lrpred_1))
> confusionMatrix(table(lrpred_1,factor(sub_test$Response))) #0.8757
Confusion Matrix and Statistics


lrpred_1     0     1
       0 66738  9473
       1    10     1

               Accuracy : 0.8756
                 95% CI : (0.8732, 0.8779)
    No Information Rate : 0.8757
    P-Value [Acc > NIR] : 0.5421

                  Kappa : -1e-04

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9998502
            Specificity : 0.0001056
         Pos Pred Value : 0.8757004
         Neg Pred Value : 0.0909091
             Prevalence : 0.8757052
         Detection Rate : 0.8755740
   Detection Prevalence : 0.9998557
      Balanced Accuracy : 0.4999779

       'Positive' Class : 0

> end_time_3_0 <- Sys.time()
> print(end_time_3_0 - start_time_3_0)
Time difference of 8.480634 secs
```

```
> precision_lr_1 <- 1/(9473+1)
> precision_lr_1
[1] 0.000105552
> F1_lr_1  <-2*(precision_lr_1*0.9998502)/(precision_lr_1+0.9998502)
> F1_lr_1
[1] 0.0002110818
```

Stepwise(backward) regression

```
> start_time_3_1 <- Sys.time()
> lr_2<-step(lr_1,direction = 'backward')
Start:  AIC=166714.7
sub_train$Response ~ Gender + Age + Driving_License + Previously_Insured +
    Vehicle_Damage + Annual_Premium + Vintage + `Vehicle_Age_< 1 Year` +
    `Vehicle_Age_> 2 Years` + `Vehicle_Age_1-2 Year` + Region_Code_0_10 +
    Region_Code_10_20 + Region_Code_20_30 + Region_Code_30_40 +
    Region_Code_40_50 + `Region_Code_equal&over_50` + Policy_Sales_Channel_0_15 +
    Policy_Sales_Channel_15_30 + Policy_Sales_Channel_30_45 +
    Policy_Sales_Channel_45_60 + Policy_Sales_Channel_60_75 +
    Policy_Sales_Channel_75_90 + Policy_Sales_Channel_90_105 +
    Policy_Sales_Channel_105_120 + Policy_Sales_Channel_120_135 +
    Policy_Sales_Channel_135_150 + Policy_Sales_Channel_equal_over_150
```

```
                                      Df Deviance    AIC
- Vintage                              1   166663 166713
<none>                                     166663 166715
- Policy_Sales_Channel_75_90           1   166667 166717
- Policy_Sales_Channel_15_30           1   166667 166717
- Policy_Sales_Channel_90_105          1   166668 166718
- Policy_Sales_Channel_30_45           1   166672 166722
- Annual_Premium                       1   166673 166723
- Policy_Sales_Channel_120_135         1   166675 166725
- Policy_Sales_Channel_135_150         1   166676 166726
- Policy_Sales_Channel_105_120         1   166679 166729
- Policy_Sales_Channel_0_15            1   166684 166734
- Policy_Sales_Channel_45_60           1   166692 166742
- Policy_Sales_Channel_equal_over_150  1   166696 166746
- Policy_Sales_Channel_60_75           1   166697 166747
- Region_Code_0_10                     1   166713 166763
- Gender                               1   166727 166777
- Driving_License                      1   166732 166782
- Region_Code_40_50                    1   166738 166788
- Region_Code_30_40                    1   166741 166791
- `Vehicle_Age_> 2 Years`              1   166753 166803
- Region_Code_10_20                    1   166755 166805
- Region_Code_20_30                    1   166805 166855
- Age                                  1   168541 168591
- `Vehicle_Age_< 1 Year`               1   168773 168823
- Vehicle_Damage                       1   171358 171408
- Previously_Insured                   1   172380 172430
```

```
Coefficients:
                                      Estimate Std. Error z value Pr(>|z|)
(Intercept)                         -3.082e+00  2.498e-01 -12.337  < 2e-16 ***
Gender                               1.002e-01  1.252e-02   8.000 1.24e-15 ***
Age                                 -2.606e-02  6.162e-04 -42.298  < 2e-16 ***
Driving_License                      1.300e+00  1.873e-01   6.942 3.87e-12 ***
Previously_Insured                  -3.995e+00  9.361e-02 -42.680  < 2e-16 ***
Vehicle_Damage                       1.991e+00  3.816e-02  52.171  < 2e-16 ***
Annual_Premium                       1.114e-06  3.489e-07   3.194 0.001404 **
`Vehicle_Age_< 1 Year`              -1.034e+00  2.275e-02 -45.443  < 2e-16 ***
`Vehicle_Age_> 2 Years`              2.059e-01  2.141e-02   9.616  < 2e-16 ***
Region_Code_0_10                     3.459e-01  4.996e-02   6.924 4.40e-12 ***
Region_Code_10_20                    4.740e-01  5.115e-02   9.267  < 2e-16 ***
Region_Code_20_30                    5.486e-01  4.840e-02  11.336  < 2e-16 ***
Region_Code_30_40                    4.356e-01  5.076e-02   8.582  < 2e-16 ***
Region_Code_40_50                    4.185e-01  5.003e-02   8.364  < 2e-16 ***
Policy_Sales_Channel_0_15           -6.295e-01  1.456e-01  -4.322 1.54e-05 ***
Policy_Sales_Channel_15_30          -3.168e-01  1.497e-01  -2.117 0.034279 *
Policy_Sales_Channel_30_45          -4.853e-01  1.629e-01  -2.980 0.002885 **
Policy_Sales_Channel_45_60          -8.368e-01  1.633e-01  -5.125 2.98e-07 ***
Policy_Sales_Channel_60_75          -1.027e+00  1.856e-01  -5.533 3.15e-08 ***
Policy_Sales_Channel_75_90          -6.391e-01  3.036e-01  -2.105 0.035259 *
Policy_Sales_Channel_90_105         -5.033e-01  2.169e-01  -2.320 0.020356 *
Policy_Sales_Channel_105_120        -7.954e-01  2.061e-01  -3.858 0.000114 ***
Policy_Sales_Channel_120_135        -4.978e-01  1.504e-01  -3.309 0.000935 ***
Policy_Sales_Channel_135_150        -6.484e-01  1.835e-01  -3.534 0.000409 ***
Policy_Sales_Channel_equal_over_150 -8.139e-01  1.512e-01  -5.383 7.31e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 226317  on 304886  degrees of freedom
Residual deviance: 166663  on 304862  degrees of freedom
AIC: 166713

Number of Fisher Scoring iterations: 9
```

```
> lrpred_2<-predict(lr_2,newdata = sub_test,type = "response")
> lrpred_2<-as.integer(round(lrpred_2))
> confusionMatrix(table(lrpred_2,factor(sub_test$Response))) #0.8757
Confusion Matrix and Statistics


lrpred_2     0     1
       0 66738  9473
       1    10     1

               Accuracy : 0.8756
                 95% CI : (0.8732, 0.8779)
    No Information Rate : 0.8757
    P-Value [Acc > NIR] : 0.5421

                  Kappa : -1e-04

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9998502
            Specificity : 0.0001056
         Pos Pred Value : 0.8757004
         Neg Pred Value : 0.0909091
             Prevalence : 0.8757052
         Detection Rate : 0.8755740
   Detection Prevalence : 0.9998557
      Balanced Accuracy : 0.4999779

       'Positive' Class : 0

> end_time_3_1 <- Sys.time()
> print(end_time_3_1 - start_time_3_1)
Time difference of 9.757932 mins
```

```
precision_lr_2 <- 1/(9437+1)
precision_lr_2
l] 0.0001059547
F1_lr_2  <-2*(precision_lr_2*0.9998502)/(precision_lr_2+0.9998502)
F1_lr_2
l] 0.0002118868
```

4、Naive Bayes

Bayes	
Accuracy	0.7229
time(sec)	33.1
Sensitivity	0.7048
Specificity	0.8506
Precision	0.8506439
F1- score	0.7708845

The Naive Bayes algorithm is simple to used and its model runs quickly. However, its every metrics look pretty good in the table. Comparing with other model, the Naive Bayes have very good performance in the binary question.

```
> # Naive Bayes
> start_time_4 <- Sys.time()
> nb1 <- naiveBayes(as.factor(Response)~.,data=sub_train[,-c(1)])
> pred_nb1 <- predict(nb1,newdata = sub_test)
> summary(pred_nb1)
    0     1
48460 27762
> confusionMatrix(pred_nb1,factor(sub_test$Response))
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 47045  1415
         1 19703  8059

               Accuracy : 0.7229
                 95% CI : (0.7197, 0.7261)
    No Information Rate : 0.8757
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3038

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7048
            Specificity : 0.8506
         Pos Pred Value : 0.9708
         Neg Pred Value : 0.2903
             Prevalence : 0.8757
         Detection Rate : 0.6172
   Detection Prevalence : 0.6358
      Balanced Accuracy : 0.7777

       'Positive' Class : 0

> end_time_4 <- Sys.time()
> print(end_time_4 - start_time_4)
Time difference of 33.13378 secs
```

Model Summary

	1	2	3	4	5	6	
Knn=num	3	11	21	9	13	15	Average
Accuracy	0.8569	0.8756	0.8758	0.8751	0.8761	0.876	0.8761
time(mins)	13.74	17.95	14.91	13.88	13.92	14.18	14.7633333
Sensitivity	0.9601	0.99732	0.999925	0.9947	0.99859	0.999401	0.99167267
Specificity	0.1290901	0.02005	0.001267	0.03272	0.01309	0.006755	0.03382868
Precision	0.1290901	0.02005489	0.00126662	0.03272113	0.01308845	0.00675533	0.03382942
F1- score	0.2275809	0.03931911	0.00253004	0.06335807	0.02583825	0.01341995	0.06200772

	1	2	3	4	5	6	
xgb=num	(0.3,6)	(0.1,6)	(0.05,6)	(0.1,8)	(0.1,4)	(0.5,6)	Average
Accuracy	0.8748	0.8757	0.8757	0.8754	0.8757	0.8738	0.8757
time(mins)	1.461	1.584	1.48	1.913	1.016	1.37	1.016
Sensitivity	0.99643	0.9997	0.99985	0.998816	0.9999251	0.99269	0.99269
Specificity	0.01805	0.001689	0.0004222	0.006122	0.0006333	0.3589	0.0004222
Precision	0.0180494	0.00168883	0.00042221	0.00612202	0.00063331	0.03588769	0.00042221
F1- score	0.03545653	0.00337197	0.00084406	0.01216945	0.00126582	0.0692711	0.00084406

Regression	Logistic	backward
Accuracy	0.8756	0.8756
time	8.48sec	9.75 mins
Sensitivity	0.9998502	0.9998502
Specificity	0.0001056	0.0001056
Precision	0.00010555	0.00010555
F1- score	0.00021108	0.00021108

Bayes	
Accuracy	0.7229
time(sec)	33.1
Sensitivity	0.7048
Specificity	0.8506
Precision	0.8506439
F1- score	0.7708845

In conclude, though other models may have better accuracy and timer than the Naive Bayes, they all have low performance on specificity, precision and F1-score. Also, since the dataset is imbalance, the accuracy of it might mislead the result and that is why we need other metrics involved and compared. To sum up, the Naive Bayes method has a very good value on every metrics, so we pick this model for the final test and the code and result as below.

```
> start_time_5 <- Sys.time()
> final_pred <- predict(nb1,newdata = test)
> end_time_5 <- Sys.time()
> print(end_time_5 - start_time_5)
Time difference of 1.015118 mins
> summary.factor(final_pred)
    0     1
80754 46283
> dim(test)
[1] 127037     28
```

Based on our prediction with the Naive Bayes model, we finally created a label column on the test data set and we met our goal that help the insurance company use the most suitable model to find people among existing health insurance customers who might be interested in vehicle insurance.

Discussion

In the whole project, although we have made a variety of modeling attempts, we finally realized a total of four modeling methods, they are the K-nearest neighbors (KNN) method, XGBoost (decision tree-based method), Regression method, and Naive Bayes method. Among the modeling prediction results of all methods, KNN model has the highest accuracy and the logistic regression model takes the shortest time. However, given that other models all had some deficiencies in specificity, precision, or F1-score, we finally chose the Naive Bayes method with the best comprehensive performance as our prediction model and recommended it to this insurance company.

Due to the defects of the data set, our project was also limited, which may eventually lead to some accuracy problems in actual use. In the original training data set provided by Kaggle, the dependent variable “Response” has a large deviation of 1 and 0, which 1 exists in about 12% of the total. The direct result of such an unbalanced ratio is that the machine deep learning process will be affected, and the learning effect will also be compromised. We also considered that the higher accuracy of the model may be due to the fact that most of the predictions are 0 (Customer is not interested). But at least we have no plan to obtain a better training set to further optimize our model so far, which is also our regret. If the insurance company can provide an updated and more ideal data set as a training set in the future, we would like to provide a more optimized modeling prediction solution accordingly.

Our project aims to help an insurance company that provides health insurance predict which of its existing customers will be interested in buying vehicle insurance, and they will optimize its business model and marketing plan based on the predicted results to increase sales and ultimately achieve profit growth. For students like us who lack practical experience, this is an excellent attempt to apply what we have learned to real-life problems. The final realization of the project goal also gave us a sense of accomplishment. And we believe what we learn and experience in the project will lay the foundation for our future career in data analysis.

Reference

[1] Health Insurance Cross Sell Prediction. Retrieved from:

https://www.kaggle.com/anmolkumar/health-insurance-cross-sell-prediction?select=test.csv

[2] Vishal Morde (Apr 2019). XGBoost Algorithm: Long May She Reign!. Retrieved from

https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[3] data school (March 2014). Simple guide to confusion matrix terminology. Retrieved from

https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/#:~:text=A%20confusion%20matrix%20is%20a,related%20terminology%20can%20be%20confusing.

[4] Wiki. Stepwise regression. Retrieved from

https://en.wikipedia.org/wiki/Stepwise_regression#:~:text=In%20statistics%2C%20stepwise%20regression%20is,based%20on%20some%20prespecified%20criterion.

[5] Wiki. Naive Bayes classifier. Retrieved from

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[6] Guru Prasad (Nov 2019). Notes on Sensitivity, Specificity, Precision,Recall and F1 score. Retrieved from https://medium.com/analytics-vidhya/notes-on-sensitivity-specificity-precision-recall-and-f1-score-e34204d0bb9b