Data Analysis

Project II

Health Insurance Cross-sell Prediction | Kaggle

Group No. 1

B. Thanushan	S14025
W. S. S. Fernando	S13990
W. M. C. B. Weerakoon	S14028

ABSTRACT

Most of the companies in the world cross-sell their new products to their existing customers before launching them to the open market. This helps the company to reduce time and cost by finding the right type of customers to sell the product and by optimizing their communication plan. This dataset is of policyholders of an insurance company for the past year. It consists of 381109 observations with 12 variables. The response variable is whether the policyholders are interested in the new vehicle insurance offered by the insurance company or not. Our main objective is to find the best accurate model to predict whether the policyholders are interested in the new vehicle insurance offered or not.

Descriptive analysis methods were used to get a clear idea about the dataset. Some of the major findings from the descriptive analysis are vehicle damage and previously insured customers have a relationship, customers who have vehicle damage are more interested in buying the vehicle insurance and majority of customers who have not vehicle insurance are interested in the new vehicle insurance provided by the company.

Logistic regression, decision tree and random classifier algorithms were used in the advanced analysis since the response variable is nominal. The response variable of the dataset is imbalanced with a majority of the responses being “no”, hence the undersampling technique was used to resample the training dataset. The variables “Region_Code” and “Policy*Sales*Channel” had a large number of categories, therefore those two variables were re-categorized.

Three models have been fitted using the logistic regression method, the decision tree method and the random forest method. By comparing the balanced accuracies and f1 scores, the random forest model has been chosen as a suitable model.

A website is also created as the data product and the objective of it is to predict whether the customer is interested in vehicle insurance or not when the user enters data into it.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

01. INTRODUCTION

When introducing a new product, a company always tries to get feedback from their existing customers about the product, and also the company tries to sell the new product to them. Also, when cross-selling a product the company needs to know which customers are viable customers for the new product so that the company can reduce the time and cost by optimizing the communication strategy plan accordingly. This dataset is of health insurance policyholders of an insurance company from the past year. The insurance company has decided to provide vehicle insurance as their new product. Now the company needs a model to predict whether those policyholders will also be interested in vehicle insurance provided by the company.

02. DESCRIPTION OF THE PROBLEM

We are going to build a model to predict whether customers of the insurance company will also be interested in vehicle insurance. This helps the insurance company to decide a communication plan to know which customers to contact so that they can reduce communication costs and time. The problem that we are going to solve using a predictive model will be helpful for the company to maximize its revenue and by reducing its costs.

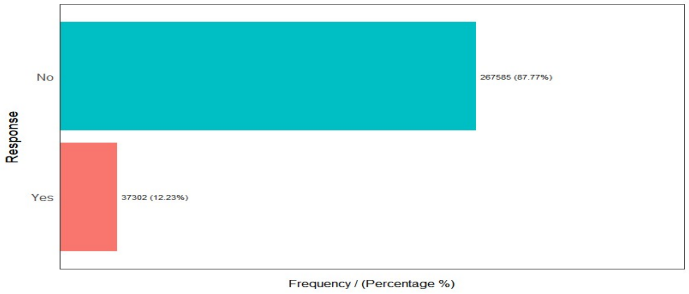
03. DESCRIPTION OF THE DATASET

The data is provided by a user of Kaggle. There are 12 variables available in the dataset both qualitative and quantitative and 381109 customer details. In Kaggle they have given two datasets as train and test but in the test dataset there is no response variable, therefore, we take the training dataset and split it into training and test. There are no null values in the dataset.

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 that customer needs to pay as a 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

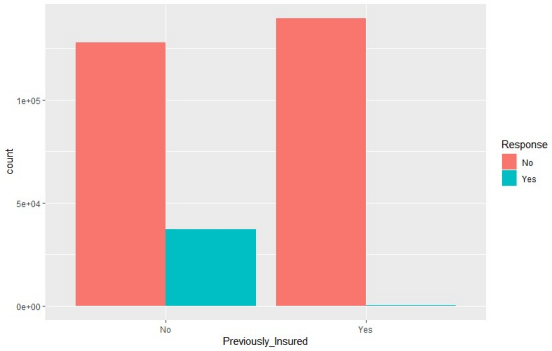
Table 03-1: Description of the dataset

04. IMPORTANT RESULTS OF THE DESCRIPTIVE ANALYSIS



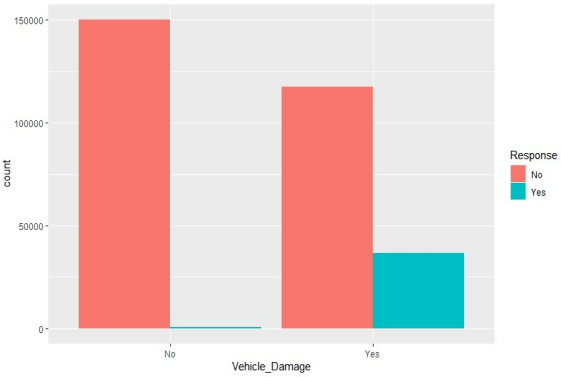
The response of the customers is imbalanced with 87.77% of the customers are not interested in the vehicle insurance and only 12.23% of the customers are interested.

Figure 04-1: Bar plot of response



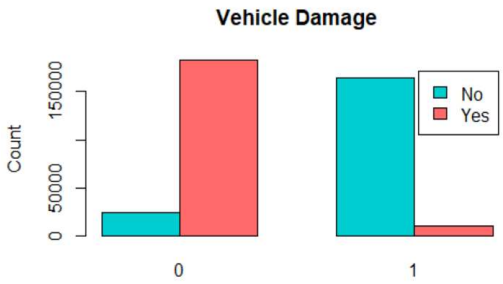
The majority of health insurance policyholders who have not vehicle insurance are interested in new vehicle insurance provided by the company and almost all the policyholders who have vehicle insurance are not interested in vehicle insurance.

Figure 04-2: Group bar plot of previously insured and response



The customers who have vehicle damage are more interested in buying the vehicle insurance provided by the company and almost all the customers who have not had vehicle damage are not interested in the new vehicle insurance.

Figure 04-3: Group bar plot of vehicle damage and response



There is a clear relationship between vehicle damage and previously insured customers. The customers who have not had vehicle insurance are most likely to have vehicle damage, and those who have had a vehicle are insurance less likely to have vehicle damage.

Figure 04-4: Group bar plot of previously insured and vehicle damage

05. IMPORTANT RESULTS OF THE ADVANCED ANALYSIS

Logistic Regression, Decision tree classifier and Random classifier Algorithms were used in the advanced analysis. Our main objective of advanced analysis to find a model with the best accuracy which predicts whether the customer is interested in vehicle insurance or not. Feature Selection was also one of our objectives in the advanced analysis.

It has been found out that the predictors Gender, Age, Driving license, Previously Insured, Vehicle damage, Region code and Policy Sales channel as the important predictors from the logistic regression. The f1 score of 0 and 1 was 0.75 and 0.39 respectively in the logistic regression. The balanced accuracy was 0.78 in the logistic regression.

From the Random forest classifier, we have got Age, Previously insured, Annual Premium, gender, vehicle age, vehicle damage, vintage and Policy sales channel as the important features. The f1 score of 0 and 1 was 0.84 and 0.49 respectively from the random forest technique. The balanced accuracy was 0.84 from the random forest technique.

We have also fitted a model using the decision tree where we have found that the f1 score of 0 and 1 was 0.74 and 0.40 respectively. The balanced accuracy was 0.78 from the Decision tree technique.

By comparison of all three models, the f1 score of both 0 and 1 was higher in the random forest model than the other two models. The balanced accuracy also was high in the random forest technique. Therefore, we select the random forest model as the best model from the above models. Hence, we use this model for prediction in our data product which is a website.

We have two pages in our website. In the homepage, the person has to enter the details of the customer and click predict button to predict whether the customer is interested in a vehicle insurance or not. The predicted value will be shown in a separate web page.

```
[[48539 18335]
 [  335  9013]]
              precision    recall  f1-score   support

           0       0.99      0.73      0.84     66874
           1       0.33      0.96      0.49      9348

    accuracy                           0.76     76222
   macro avg       0.66      0.84      0.66     76222
weighted avg       0.91      0.76      0.80     76222
```

```
balanced_accuracy_score(test_health_y,y_general_predict)
```

```
0.8449955666759561
```

Figure 05-2: Confusion matrix of random forest

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 40799   390
         1 26015  9018

               Accuracy : 0.6536
                 95% CI : (0.6502, 0.657)
    No Information Rate : 0.8766
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2623

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

            Sensitivity : 0.9585
            Specificity : 0.6106
         Pos Pred Value : 0.2574
         Neg Pred Value : 0.9905
             Prevalence : 0.1234
         Detection Rate : 0.1183
   Detection Prevalence : 0.4596
      Balanced Accuracy : 0.7846

       'Positive' Class : 1
```

Figure 05-3: Confusion matrix of logistic regression

```
[[39581 27293]
 [  204  9144]]
              precision    recall  f1-score   support

           0       0.99      0.59      0.74     66874
           1       0.25      0.98      0.40      9348

    accuracy                           0.64     76222
   macro avg       0.62      0.79      0.57     76222
weighted avg       0.90      0.64      0.70     76222
```

```
balanced_accuracy_score(test_health_y,y_general_predict)
```

```
0.7850257106048457
```

Figure 05-1: Confusion matrix of decision tree

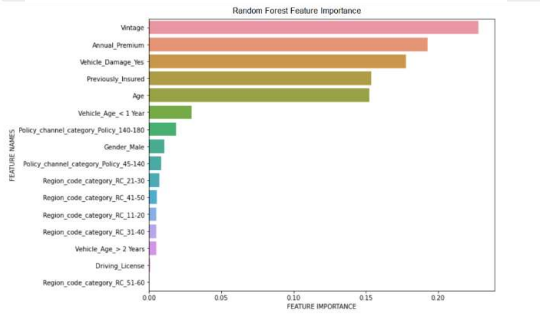


Figure 05-4: Feature importance plot of the final model

06. ISSUES ECOUNTERED AND PROPOSED SOLUTIONS

The variables “Region_Code” had 53 categories and “Policy*Sales*Channel” had 163 categories. Modeling the dataset using both R and python by creating a dummy variable for every category of those variables has taken a lot of time and the both programs ended up getting crashed. To overcome this issue, we re-categorized the variable “Region_Code” into 6 categories and the variable “Policy*Sales*Channel” into 3 categories.

The main issue that we encountered was the imbalanced response of the dataset. When fitting a model ignoring the imbalanced response, we got a low model balanced accuracy values for all three advanced analysis techniques that we used. Furthermore, the fitted model predicted almost all the “yes” responses as “no”. To overcome this problem, we used the undersampling technique, and then we got a better balanced accuracy values for all three models than before.

When we are fitting random forest and decision tree models the R studio took long time to process and got crashed. Therefore, we used python to build these two models.

07. DISCUSSION AND CONCLUSIONS

The main objective of the analysis was to predict the interest of the customer for the new vehicle insurance provided by their existing health insurance company. Another objective of the study was to know which factors affect the response of the customers the most. The dataset consists of 381109 observations with 12 variables.

The response of the dataset was imbalanced with most customers are not interested in buying vehicle insurance and hence the dataset was resampled using the undersampling method to get a higher balanced accuracy. The variable “Region_Code” and the variable “Policy*Sales*Channel” are re-categorized into 6 categories and 3 categories respectively. The random forest model was selected over logistic regression and decision tree methods due to higher balanced accuracy and f1 values, to model the dataset. “Vintage” is the most important variable in the model. Furthermore, the variables “Vintage”, “Annual_Premium”, “Vehicle_Damage_Yes”, “Previously_Insured” and “Age” have stand-out importance from other variables and hence we can conclude that the fact that whether the customer is interested in the new vehicle insurance or not is heavily relied on those five factors.

A website was created using the final model where the user can enter the relevant values for the variables into the website and know whether the customer is interested in vehicle insurance or not.

CODE OF THE BEST MODEL (RANDOM CLASSIFICATION MODEL)

```
import numpy as np ;import pandas as pd;import sklearn;from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split;from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report;from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import average_precision_score;from sklearn.metrics import recall_score;from sklearn.metrics import
accuracy_score;from sklearn.metrics import balanced_accuracy_score
Health_data = pd.read_csv('train.csv')
Health_data.head()
Health_data = Health_data.drop(['id'],axis=1)
Region_code_category = pd.cut(Health_data.Region_Code,bins=[0,10,20,30,40,50,60],labels=['RC_1-10','RC_11-20','RC_21-
30','RC_31-40','RC_41-50','RC_51-60'])
Health_data.insert(4,'Region_code_category',Region_code_category)
Policy_channel_category = pd.cut(Health_data.Policy_Sales_Channel,bins=[0,45,140,180],labels=['Policy_1-45','Policy_45-
140','Policy_140-180'])
Health_data.insert(10,'Policy_channel_category',Policy_channel_category)
Health_data = Health_data.drop(['Region_Code'],axis=1)
Health_data = Health_data.drop(['Policy_Sales_Channel'],axis=1)
Health_data = pd.get_dummies(Health_data,columns=['Gender'],drop_first=True)
Health_data = pd.get_dummies(Health_data,columns=['Vehicle_Age'],drop_first=True)
Health_data = pd.get_dummies(Health_data,columns=['Vehicle_Damage'],drop_first=True)
Health_data = pd.get_dummies(Health_data,columns=['Region_code_category'],drop_first=True)
Health_data = pd.get_dummies(Health_data,columns=['Policy_channel_category'],drop_first=True)
train_health,test_health = train_test_split(Health_data,test_size=0.2,random_state=20)
train_health.head()
train_health['Response'].value_counts()
no_of_yes = len(train_health[train_health['Response'] == 1])
No_indices = train_health[train_health.Response == 0].index
random_No_indices = np.random.choice(No_indices,no_of_yes, replace=False)
Yes_indices = train_health[train_health.Response == 1].index
under_sample_indices = np.concatenate([Yes_indices,random_No_indices])
under_sample_health = train_health.loc[under_sample_indices]
train_health_under,test_health_under = train_test_split(under_sample_health,test_size=0.2,random_state=0)
train_health_under.head()
train_health_under.shape
train_under_y =  train_health_under['Response']
train_under_x = train_health_under.drop(['Response'],axis =1)
test_under_y =  test_health_under['Response']
test_under_x = test_health_under.drop(['Response'],axis =1)
Under_rf = RandomForestClassifier(n_estimators=100, random_state=0, n_jobs=-1)
Under_rf.fit(train_under_x,train_under_y)
feat_labels = train_under_x.columns
for feature in zip(feat_labels, Under_rf.feature_importances_):
   print(feature)
sfm = SelectFromModel(Under_rf, threshold=0.01)
sfm.fit(train_under_x, train_under_y)
Selected_Features = train_under_x.columns[(sfm.get_support())]
X_important_train = train_under_x[Selected_Features]
X_important_test = test_under_x[Selected_Features]
rf_important = RandomForestClassifier(n_estimators=100, random_state=20, n_jobs=-1)
```

```
rf_important.fit(X_important_train, train_under_y)
y_important_pred = rf_important.predict(X_important_test)
print(confusion_matrix(test_under_y,y_important_pred))
print(classification_report(test_under_y,y_important_pred))
balanced_accuracy_score(test_under_y,y_important_pred)
test_health_y =  test_health['Response']
test_health_x = test_health.drop(['Response'],axis =1)
X_general_test = test_health_x[Selected_Features]
y_general_predict = rf_important.predict(X_general_test)
print(confusion_matrix(test_health_y,y_general_predict))
print(classification_report(test_health_y,y_general_predict))
```

SAMPLE CODE OF THE FORM IN HOME PAGE

```
<form name="predict" id="predict" method="post" action="{{url_for('get_data')}}" >
        <div class="row">
            <div class="col-md-6"><label style="text-align: center">Age :</label></div>
            <div class="col-md-6"><input type="number" id="Age" name="Age" class="form form-control"></div>
         </div>
         <div class="clearfix"> </div>
         <div class="row">
            <div class="col-md-6"><label style="text-align: center">Gender :</label></div>
             <div class="col-md-6"><select name="user_gender" id="user_gender" class="form-control" >
                            <option value="">Please select a gender</option>
                            <option value="Male">Male</option>
                            <option value="Female">Female</option>
                        </select>
             </div>
            </div>
            <div class="clearfix"> </div>
        <div class="row">
            <div class="col-md-6"><label style="text-align: center">Have a driving license :</label></div>
               <div class="col-md-6"><select name="d_license" id="d_license" class="form-control" >
                            <option value="">Please select yes/no</option>
                            <option value="Yes">Yes</option>
                            <option value="No">No</option>
                        </select>
            </div>
           </div>
         <div class="clearfix"> </div>
         <div class="row">
            <div class="col-md-6"><label style="text-align: center">Annual Premium :</label></div>
              <div class="col-md-6"><input type="textbox" id="A_premium" name="A_premium" class="form form-
control"></div>
            </div>
```

SAMPLE CODE OF AFTER HTML PAGE

```
<center>
                <h1>Whether the person will be interested in vehicle insurance or not</h1>
                {% if data == 1 %}
                <h1>YES</h1>
                {% else %}
                <h1>NO</h1>
                {% endif %}
                <br><br><br><a href="/">go back to home page</a>
</center>
```

SAMPLE CODE OF BACKEND

```
from flask import Flask,request,render_template,redirect,url_for
import numpy as np
import joblib
model = joblib.load("RandomClassifier_model.joblib")
app = Flask(__name__)
# render default webpage
@app.route('/')
def home():
   return render_template('home_page.html')
@app.route('/', methods=['POST'])
def get_data():
      age = request.form['Age']
      gender = request.form['user_gender']
      insured = request.form['insured']
      V_age = request.form['V_Age']
      PSC = request.form['PSC']
      Vintage = request.form['Vintage']
      A_premium = request.form['A_premium']
      V_age = int(V_age)
      PSC = int(PSC)
      if gender == "Male":
          gender = 1
      else :
          gender =0
      if insured == "Yes":
          insured = 1
      else :
          insured =0
      if V_age < 2:
          V_age = 1
      else :
          V_age = 0
      if PSC > 45 or PSC < 140:
          PSC = 1
      else :
          PSC = 0
      arr = np.array([[age,insured,A_premium,Vintage,gender,V_age,PSC]])
      pred = model.predict(arr)
     return render_template('after.html',data=pred)
if __name__ == '__main__':
   app.run(debug=True,port=5000)
```

Sample R code of logistic regression

```
#Dividing the dataset
set.seed(123)
train_index3 <- sample(1:nrow(data_under), 0.8 * nrow(data_under))
test_index3 <- setdiff(1:nrow(data_under), train_index3)
train.health3 <- data_under[train_index3,]
test.health3 <- data_under[test_index3,]
dim(train.health3)
dim(test.health3)
summary(train.health3)
table(train.health3$Response)
37302*2
under3 <- ovun.sample(Response~., data = train.health1, method = "under", N = 74604)$data
table(under3$Response)
#Dividing the undersampled data
set.seed(123)
train_index_under <- sample(1:nrow(under3), 0.8 * nrow(under3))
test_index_under <- setdiff(1:nrow(under3), train_index_under)
train.health_under <- under3[train_index_under,]
test.health_under <- under3[test_index_under,]
dim(train.health_under)
dim(test.health_under)
summary(train.health_under)
table(train.health_under$Response)
#Fitting the model
log_model_under <- glm(Response ~ ., data = train.health_under, family = 'binomial')
summary(log_model_under)
model_under <- stepAIC(log_model_under)
summary(model_under)
coef(model_under)
# Make predictions
probabilities_under <- predict(model_under, test.health_under, type = "response")
predicted.classes_under <- ifelse(probabilities_under< 0.5, 0, 1)
# Prediction accuracy
observed.classes_under <- test.health_under$Response
mean(predicted.classes_under == observed.classes_under)
confusionMatrix(as.factor(predicted.classes_under), as.factor(observed.classes_under), positive = "1")
```

Sample python code for decision tree classifier

```
# Function to perform training with giniIndex.
def train_using_gini(train_under_x, test_under_x, train_under_y):
    # Creating the classifier object
    clf_gini = DecisionTreeClassifier(criterion = "gini",
            random_state = 100,max_depth=3, min_samples_leaf=5)
    # Performing training
    clf_gini.fit(train_under_x, train_under_y)
    return clf_gini
clf_gini = train_using_gini(train_under_x, test_under_x, train_under_y)
y_pred_gini = clf_gini.predict(test_under_x)
print(confusion_matrix(test_under_y,y_pred_gini))
print(classification_report(test_under_y,y_pred_gini))
```