# HarvardX:PH125.9x Final Capstone-Predictive Model on Insurance Claims

Andrew Chikunga

August 17, 2020

### Background

Recently, there has been an increase in the number of building collapse in Lagos and major cities in Nigeria. Olusola Insurance Company offers a building insurance policy that protects buildings against damages that could be caused by a fire or vandalism, by a flood or storm.

As a Data Analyst,you are to build a predictive model to determine if a building will have an insurance claim during a certain period or not. You will have to predict the probability of having at least one claim over the insured period of the building.

The model will be based on the building characteristics. The target variable, Claim, is a categorical variable with:

1 if the building has at least a claim over the insured period. 0 if the building doesn't have a claim over the insured period.

### Problem Statement:

To build a logistic model that will predict if a building will have an insurance claim or not during a certain period.

## 1.Importing Essential Libraries

I will begin by importing our dependencies that we require. The following dependencies are popularly used for data wrangling operations and visualizations.

## 2. Loading the required datasets.

### Train_data set

We start by loading the train_data set using the code below:

```
##   Customer.Id YearOfObservation Insured_Period Residential Building_Painted
## 1      H14663              2013              1           0                N
## 2       H2037              2015              1           0                V
## 3       H3802              2014              1           0                N
## 4       H3834              2013              1           0                V
## 5       H5053              2014             1          0                V
## 6       H4977              2012             1          0                V
##   Building_Fenced Garden Settlement Building.Dimension Building_Type
## 1               V      V          U                290             1
## 2               N      O          R                490             1
## 3               V      V          U                595             1
## 4               V      V          U               2840             1
## 5               N      O          R                680             1
## 6               N      O          R                535             1
##   Date_of_Occupancy NumberOfWindows Geo_Code Claim
## 1              1960               .     1053     0
## 2              1850               4     1053     0
## 3              1960               .     1053     0
## 4              1960               .     1053     0
## 5              1800               3     1053     0
## 6              1980               3     1143     0
```

I will start with using the dim function to print out the dimensionality of our dataframe.

```
## [1] 7160   14
```

I will use the summary() to review the contents and shape of the data.As its name implies, the summarize function reduces a data frame to a summary of just one vector or value. this function is valuable because it often provides exactly what is needed in terms of summary statistics.

```
summary(train_data)
```

```
##   Customer.Id   YearOfObservation Insured_Period    Residential
##  H12608 :   1   Min.   :2012      Min.   :0.0000   Min.   :0.0000
##  H12616 :   1   1st Qu.:2012      1st Qu.:0.9973   1st Qu.:0.0000
##  H12617 :   1   Median :2013      Median :1.0000   Median :0.0000
##  H12619 :   1   Mean   :2014      Mean   :0.9098   Mean   :0.3054
##  H12621 :   1   3rd Qu.:2015      3rd Qu.:1.0000   3rd Qu.:1.0000
##  H12622 :   1   Max.   :2016      Max.   :1.0000   Max.   :1.0000
##  (Other):7154
##  Building_Painted Building_Fenced Garden   Settlement Building.Dimension
##  N:1778           N:3608           :   7   R:3610     Min.   :    1
##  V:5382           V:3552          O:3602   U:3550     1st Qu.:  528
##                                   V:3551              Median : 1083
##                                                       Mean   : 1884
##                                                       3rd Qu.: 2290
##                                                       Max.   :20940
##                                                       NA's   :106
##  Building_Type   Date_of_Occupancy NumberOfWindows    Geo_Code
##  Min.   :1.000   Min.   :1545         .   :3551    6088   : 143
##  1st Qu.:2.000   1st Qu.:1960      4      : 939    33063  : 137
##  Median :2.000   Median :1970      3      : 844    6083   : 113
##  Mean   :2.186   Mean   :1964      5      : 639           : 102
##  3rd Qu.:3.000   3rd Qu.:1980      2      : 363    13206  :  98
##  Max.   :4.000   Max.   :2016      6      : 306    31555  :  87
##                  NA's   :508       (Other): 518    (Other):6480
##      Claim
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.2282
##  3rd Qu.:0.0000
##  Max.   :1.0000
##
```

Now that I have a general overview of the nature of the data as shown by the statistics above,I will move on to an in depth analysis.

The str method will allows us to know the data type of each variable and factor levels if any.

```
## 'data.frame':    7160 obs. of  14 variables:
##  $ Customer.Id       : Factor w/ 7160 levels "H12608","H12616",..: 1077 4057 4991 5007 5775 5721 710
##  $ YearOfObservation : int  2013 2015 2014 2013 2014 2012 2012 2015 2014 2015 ...
##  $ Insured_Period    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Residential       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Building_Painted  : Factor w/ 2 levels "N","V": 1 2 1 2 2 2 1 1 2 2 ...
##  $ Building_Fenced   : Factor w/ 2 levels "N","V": 2 1 2 2 1 1 2 2 1 1 ...
##  $ Garden            : Factor w/ 3 levels "","O","V": 3 2 3 3 2 2 3 3 2 2 ...
##  $ Settlement        : Factor w/ 2 levels "R","U": 2 1 2 2 1 1 2 2 1 1 ...
##  $ Building.Dimension: int  290 490 595 2840 680 535 2830 4952 2735 520 ...
##  $ Building_Type     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Date_of_Occupancy : int  1960 1850 1960 1960 1800 1980 1988 1988 2013 2011 ...
##  $ NumberOfWindows   : Factor w/ 11 levels "   .",">=10",..: 1 6 1 1 5 5 1 1 5 4 ...
##  $ Geo_Code          : Factor w/ 1308 levels "","10033","10081",..: 9 9 9 9 9 16 16 18 19 28 ...
##  $ Claim             : int  0 0 0 0 0 0 0 0 1 0 ...
```

The first column represents the Customer ID of each point. From the second to the fourteenth column we have the 13 predictors:

("Customer.Id","YearOfObservation" "Insured_Period","Residential","Building_Painted","Building_Fenced","Garden","Se and finally in the last column we have the binary response variable ("Claim").

With the str() function, we can conclude the dataset has 7160 observations of 14 variables.

## variable definations

Customer Id: Identification number for the Policy holder.

YearOfObservation: year of observation for the insured policy.

Insured_Period: duration of insurance policy in Olusola Insurance. (Ex: Full year insurance, Policy Duration = 1; 6 months = 0.5).

Residential: is the building a residential building or not.

Building_Painted: is the building painted or not (N-Painted, V-Not Painted).

Building_Fenced:is the building fence or not (N-Fenced, V-Not Fenced).

Garden building: has garden or not (V-has garden; O-no garden).

Settlement Area: where the building is located. (R- rural area; U- urban area).

Building Dimension: Size of the insured building in m2.

Building_Type: The type of building (Type 1, 2, 3, 4).

Date_of_Occupancy: date building was first occupied.

NumberOfWindows: number of windows in the building.

Geo Code: Geographical Code of the Insured building.

Claim target variable: (0: no claim, 1: at least one claim over insured period).

## 3.Exploratory Data Analysis

Data Exploration is one of the most significant portions of the machine learning process. Clean data can ensure a notable increase in accuracy of our model. No matter how powerful our model is, it cannot function well unless the data we provide it has been thoroughly processed. This step will briefly take you through this step and assist me to visualize the data, find relation between variables, deal with missing values and outliers and assist in getting some fundamental understanding of each variable I will use. Moreover, this step will also enable me to figure out the most important attibutes to feed my model and discard those that have no relevance.

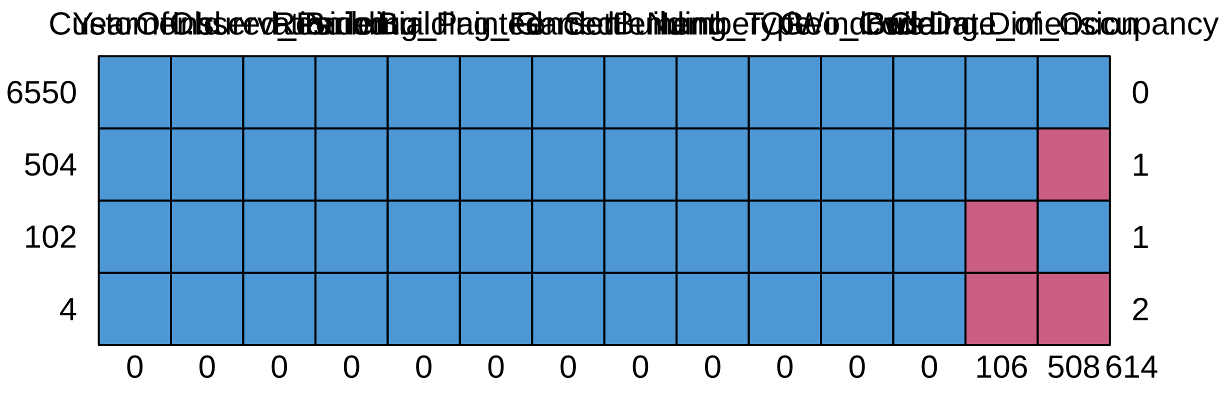
I will begin the data cleaning by checking if the data observations have any missing values:

### Missing values

```
##
## FALSE  TRUE
##   610  6550
```

The above code snippet indicates that 610 sample cases have NA values.In order to fix this, let's look at the summary of all our variables and analyze which variables have the greatest number of null values.The mice package provides a nice function md.pattern() to get a better understanding of the pattern of missing data.

```
md.pattern(train_data)
```



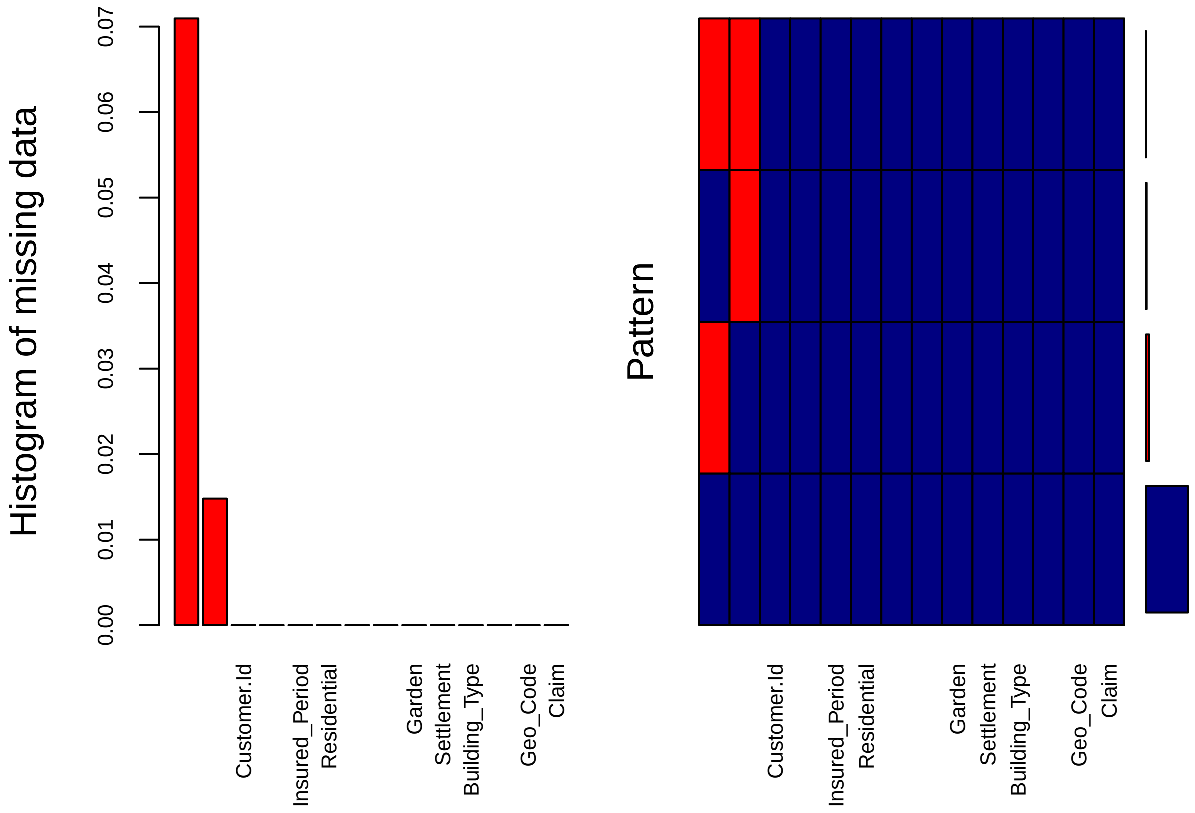
```
##      Customer.Id YearOfObservation Insured_Period Residential Building_Painted
## 6550           1                 1              1           1                1
## 504            1                 1              1           1                1
## 102            1                 1              1           1                1
## 4              1                 1              1           1                1
##                0                 0              0           0                0
##      Building_Fenced Garden Settlement Building_Type NumberOfWindows Geo_Code
## 6550               1      1          1             1               1        1
## 504                1      1         1             1               1        1
## 102                1      1         1             1               1        1
## 4                  1      1         1             1               1        1
##                    0      0         0             0               0        0
##      Claim Building.Dimension Date_of_Occupancy
## 6550     1                  1                 1   0
## 504      1                  1                 0   1
## 102      1                  0                 1   1
## 4        1                  0                 0   2
##          0                106               508 614
```

The output tells us that 6550 samples are complete while it is observed that two variables have a good amount of NA values: Date_of_Occupancy –504,Building.Dimension – 102 & 4 samples missing under Date_of_Occupancy and Building.Dimension are uninterpretable. The reason why we must get rid of missing values is that they lead to wrongful predictions and hence decrease the accuracy of our model.

### summary of missing values

```
aggr_plot <- aggr(train_data, col=c('navyblue','red'), numbers=TRUE, sortVars=TRUE, labels=names(train_
```

```
## Warning in plot.aggr(res, ...): not enough horizontal space to display
## frequencies
```

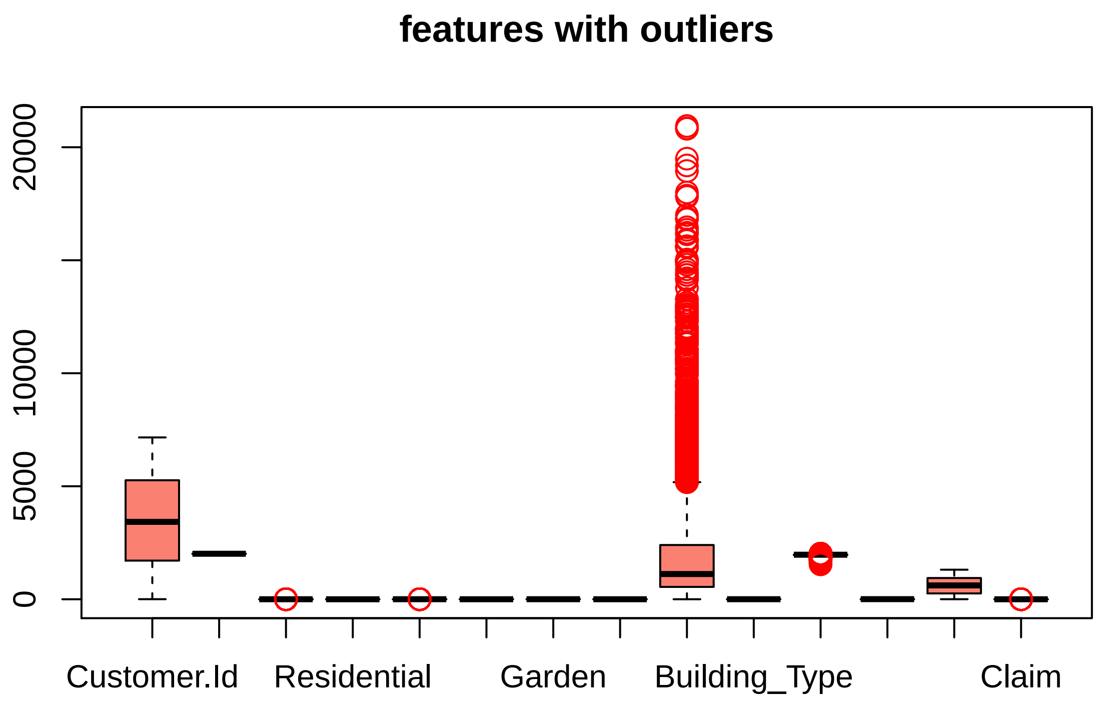


```
##
```

```
##  Variables sorted by number of missings:
##            Variable      Count
##   Date_of_Occupancy 0.07094972
##  Building.Dimension 0.01480447
##         Customer.Id 0.00000000
##   YearOfObservation 0.00000000
##      Insured_Period 0.00000000
##         Residential 0.00000000
##    Building_Painted 0.00000000
##     Building_Fenced 0.00000000
##              Garden 0.00000000
##          Settlement 0.00000000
##       Building_Type 0.00000000
##     NumberOfWindows 0.00000000
##            Geo_Code 0.00000000
##               Claim 0.00000000
```

The plot helps us understanding that almost 91.4% of the samples are not missing any information, 7.1% are missing Date_of_Occupancy value, and 1.5% are missing Building.Dimension. Through this approach the situation looks a bit clearer in my opinion.This also tells us that our dataset is of good quality.

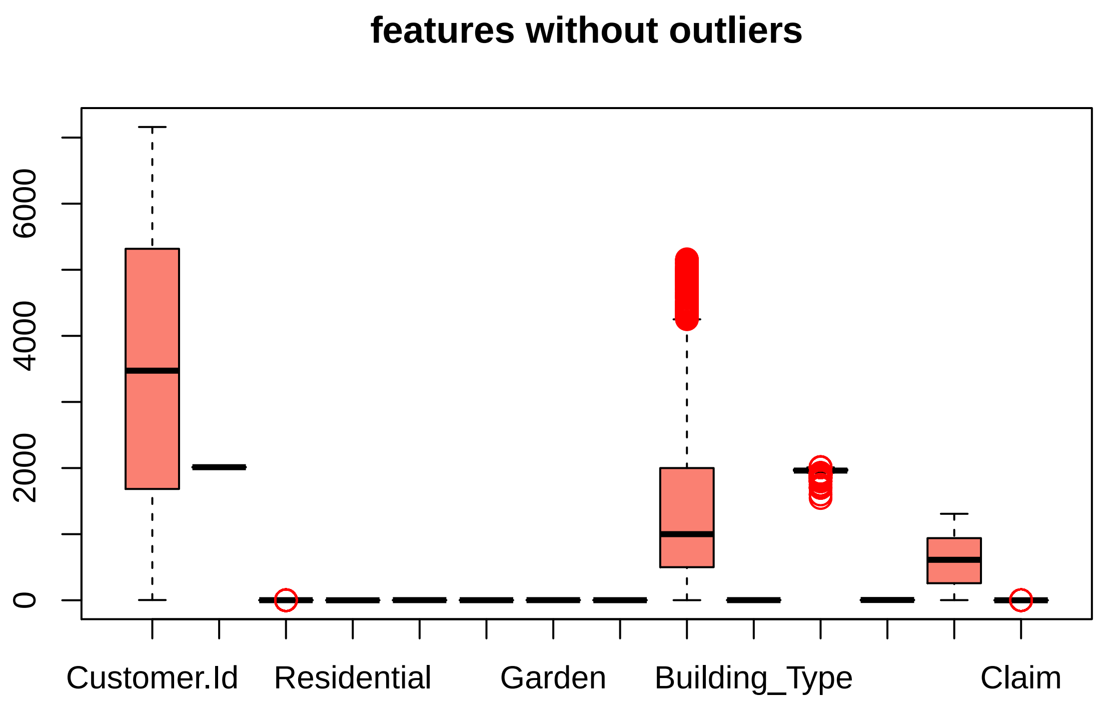
## Visualising distributions:Outlier Analysis or Clustering:



Using the boxplots above,it is straightforward that the [Building Dimension] variable has visible outliers.I am going to remove the ouliers using the code below.

## Removing the outliers on the Building.Dimension data variable

```
Q <- quantile(train_data$Building.Dimension, probs=c(.25, .75), na.rm = FALSE)
iqr <- IQR(train_data$Building.Dimension) # Interquartile range
up <-  Q[2]+1.5*iqr # Upper Range
low<- Q[1]-1.5*iqr # Lower Range
new_train <- subset(train_data, Building.Dimension > (Q[1] - 1.5*iqr) & Building.Dimension < (Q[2]+1.5*i
boxplot(new_train,outcol = "red",main = "features without outliers",col ="salmon", outcex = 1.5) # box p
```



## Investigating variability in all the features of train_data Dataset

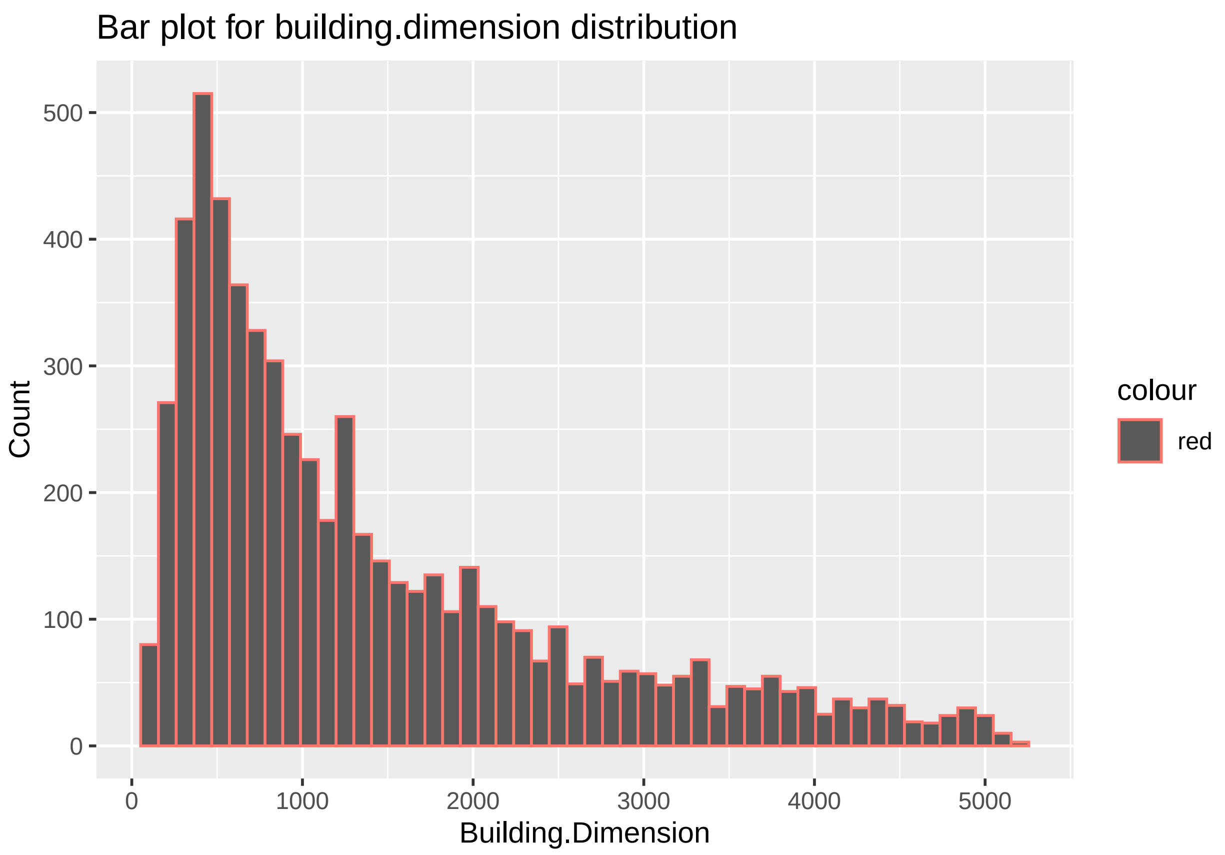
### Continous variables

A variable is continuous if it can take any of an infinite set of ordered values. residential,Building.Dimension,Building type and Date_of_Occupancy examples of continuous variables. To examine the distribution of a continuous variable, I will use a histogram:

### Histogram for Building.Dimension

The data still has a big spike on Dimensions below 50 square meters.I am going to remove all dimension below 50 by using the code below.Please note building regulation in nigeria prescribes that industrial buildings should be atleast 60 square metres while residential is 50 square metres.

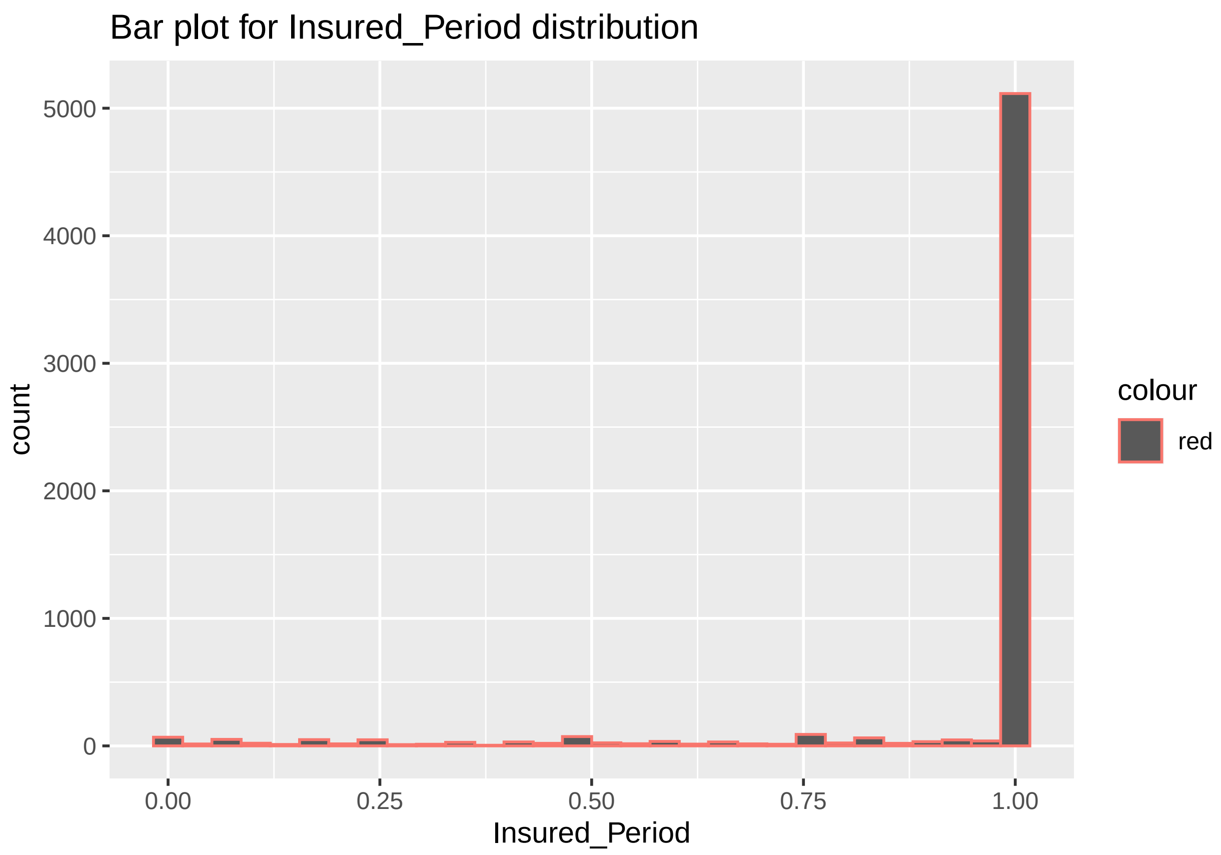
```
new_train<-new_train%>%
filter(Building.Dimension > 50)
qplot(data = new_train, x = Building.Dimension,bins=50,main="Bar plot for building.dimension distributi
```



The data is now left skewed suggesting that the median value is greater than the mean.Furthur cleaning is necessary on this variable.

### Histogram for Insured_period

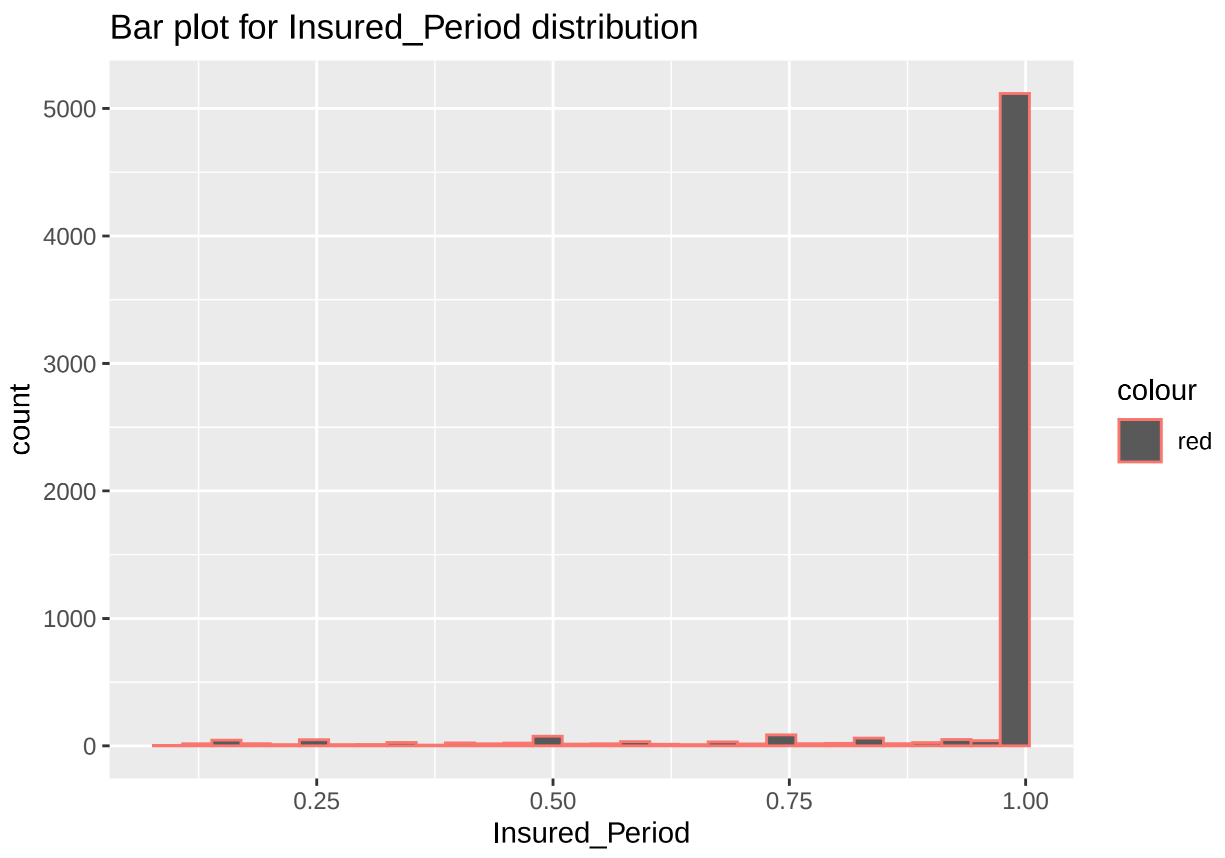
```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



From the distribution above,the number of buildings that were insured for the whole year are significantly higher.This variable indeed shows that it is less significant in affecting the target variable. There is a sizeable number of insured__periods that did not even last a month.I will remove these observations as they are not significant in this study.

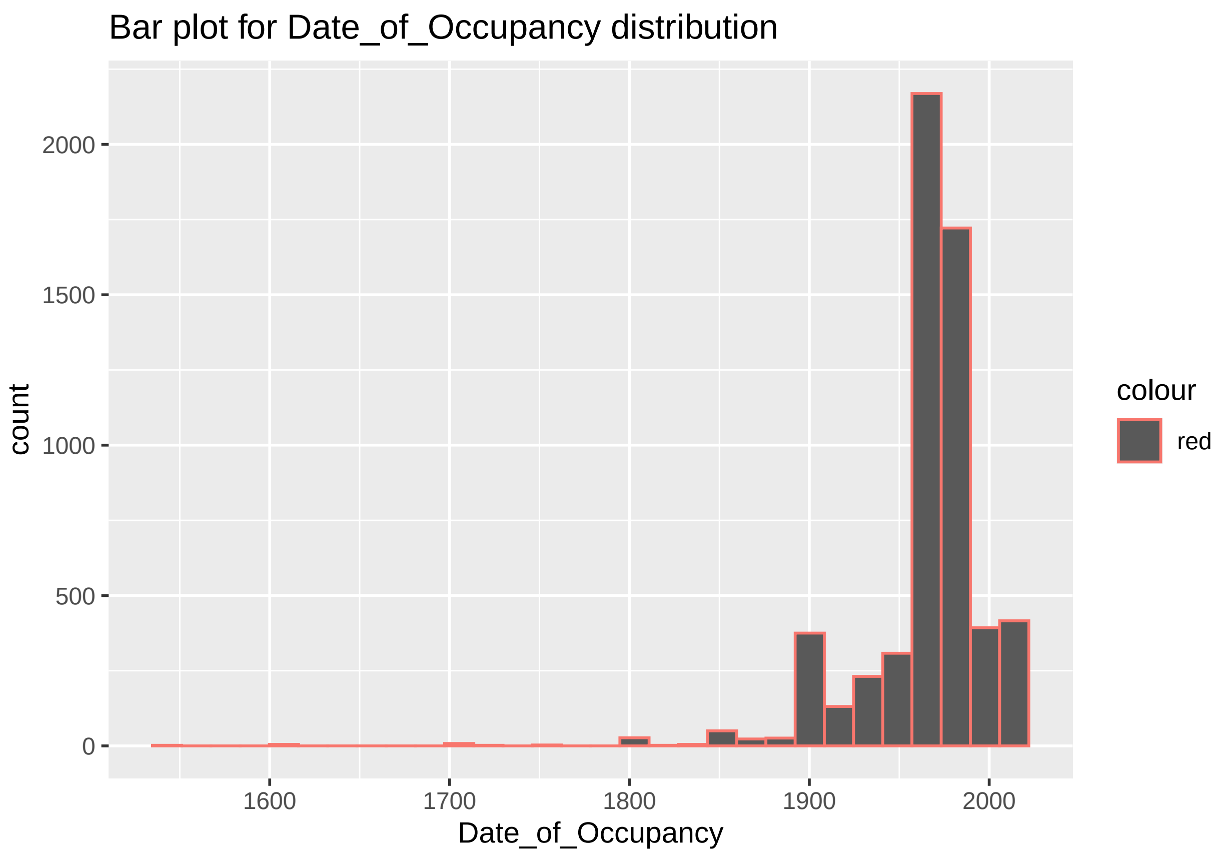
```
new_train<-new_train%>%
filter(Insured_Period > 0.1)
qplot(data = new_train, x = Insured_Period,main="Bar plot for Insured_Period distribution",colour="red"
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



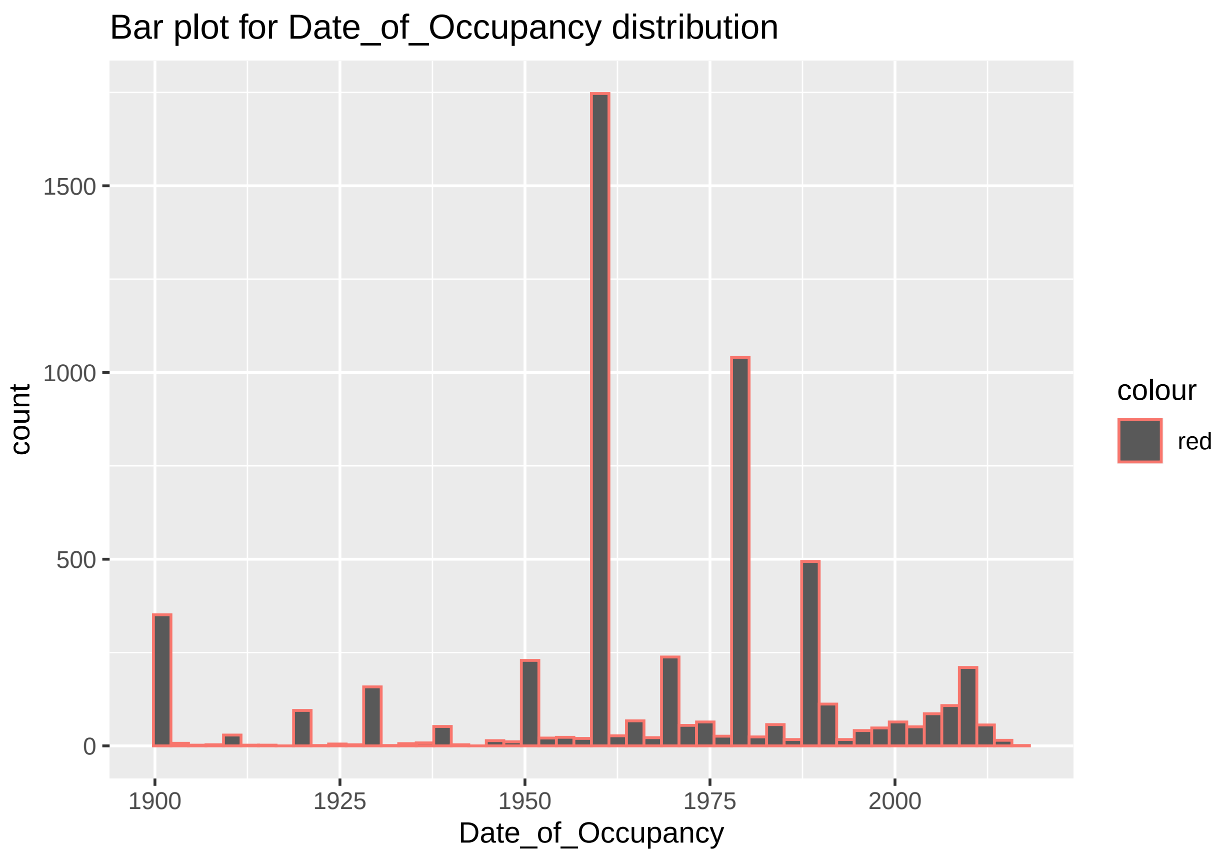
### Histogram for Date_of_Occupancy

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



There are almost zero houses between 1500 and 1900.This could be because houses of this period are either dilpailated or of little value to insure.I am going to remove all houses that are less than the year 1900 with the code below:

```
new_train<-new_train%>%
filter(Date_of_Occupancy >=1900)
qplot(data = new_train, x = Date_of_Occupancy,bins=50,main="Bar plot for Date_of_Occupancy distribution
```

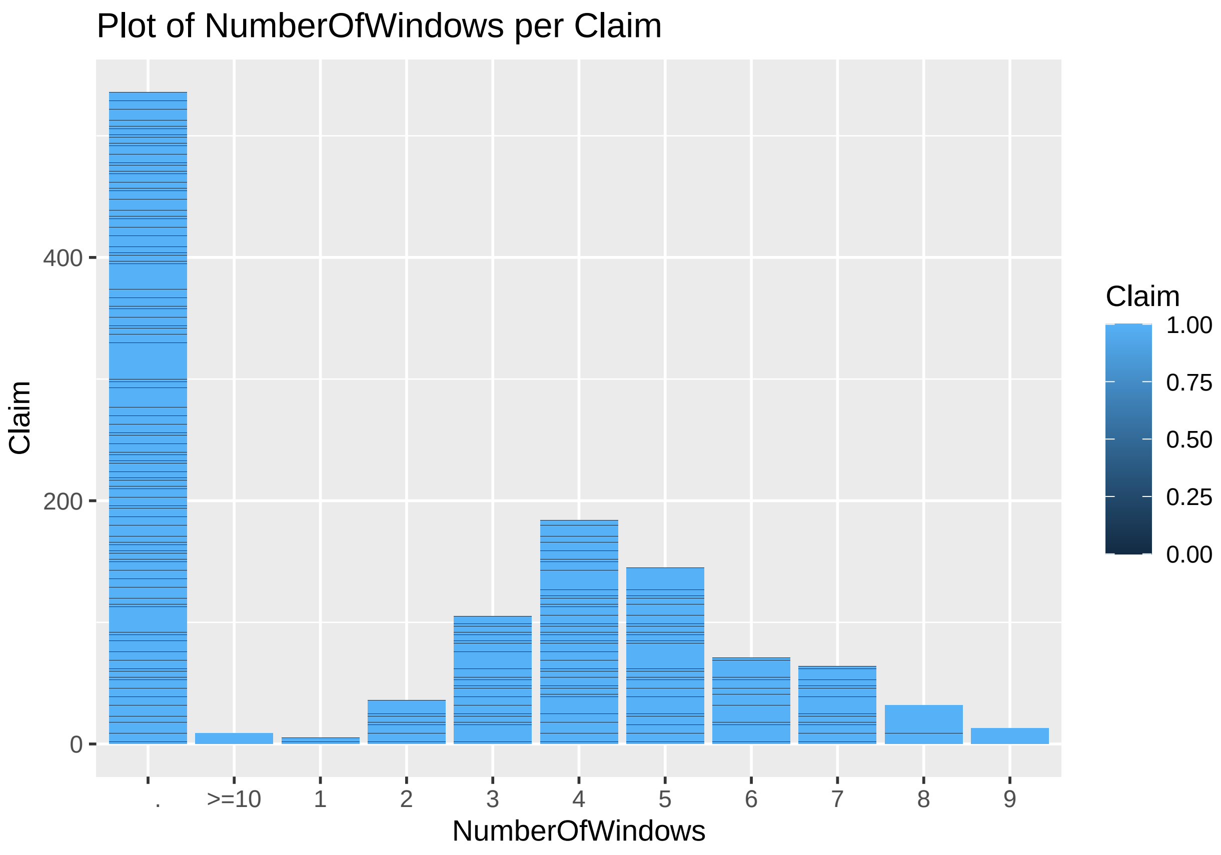


This variable shows little spikes with no definate pattern in occuring.In my own,the variable is significant in building this model.

## Categorical Variables

A variable is categorical if it can only take one of a small set of values. In R, categorical variables are usually saved as factors or character vectors. To examine the distribution of a categorical variable, use a bar chart:The height of the bars displays how many observations occurred with each x value.

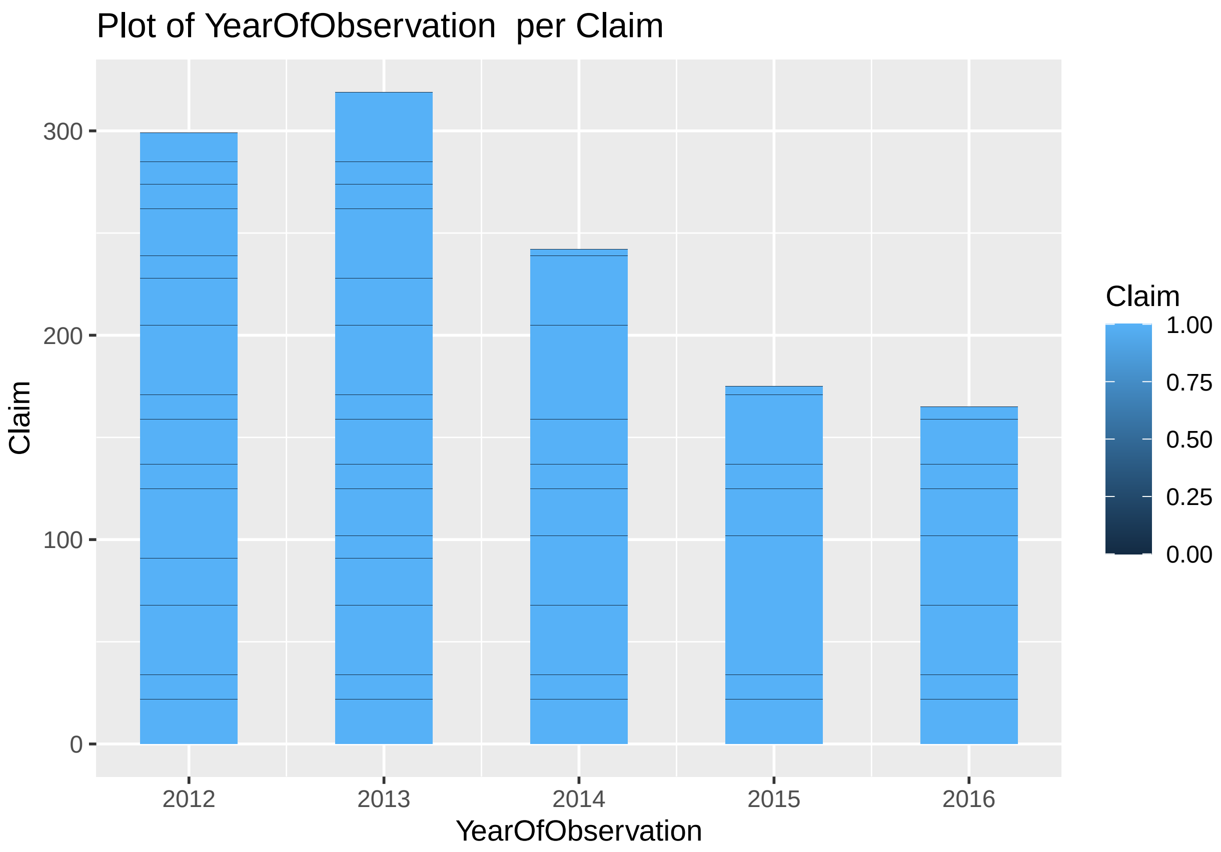
**Bar graph showing the NumberOfWindows.**



The character "." is the most appearing value in the NumberOfWindows feature.statistically it does not make sense to treat them as missing values as this will seriously alter the quality of the data.In my own opinion, I am going to replace this character with zero values suggesting buildings without windows using the following code:

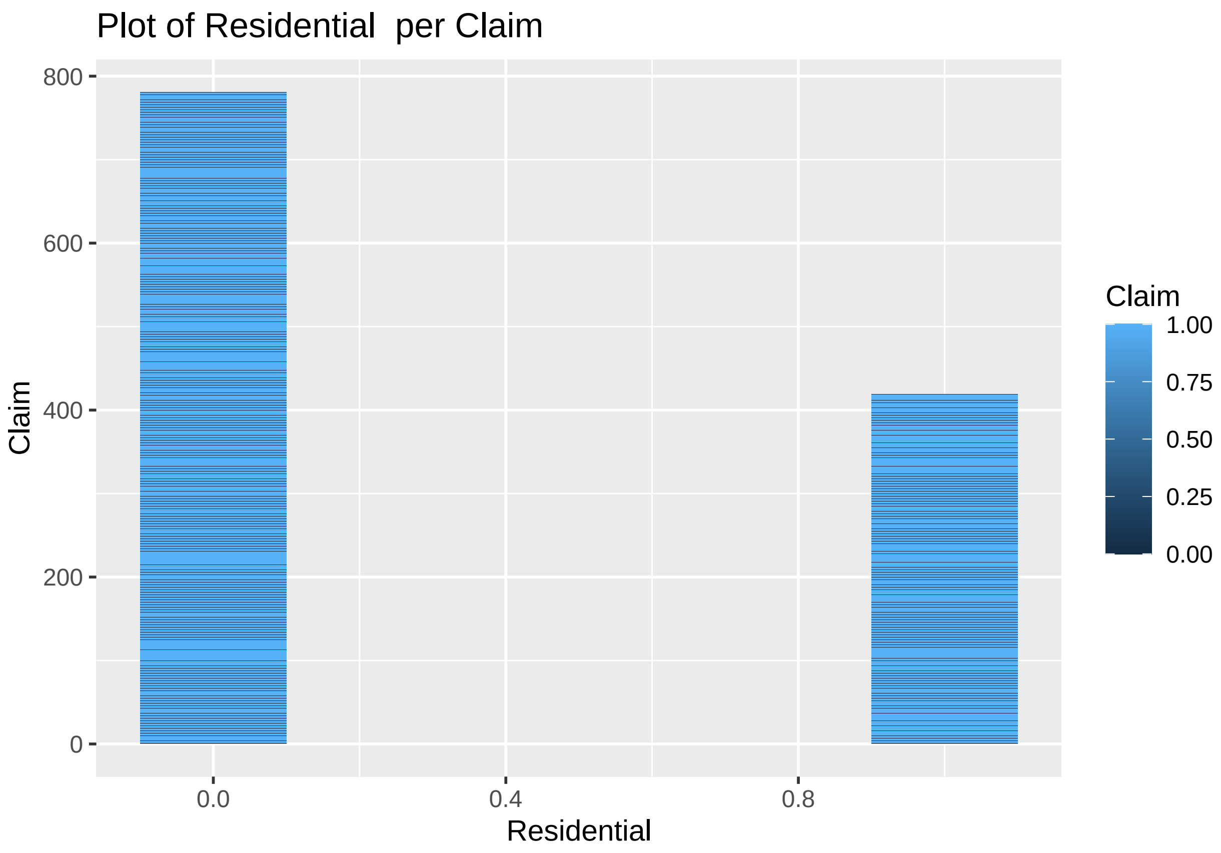
The category zero has twice the number of buildings with the second most appearing.This category tells us this type of buildings constructed in Nigeria therefore it is significant in interpreting the model.

## Bar graph showing year of Observation



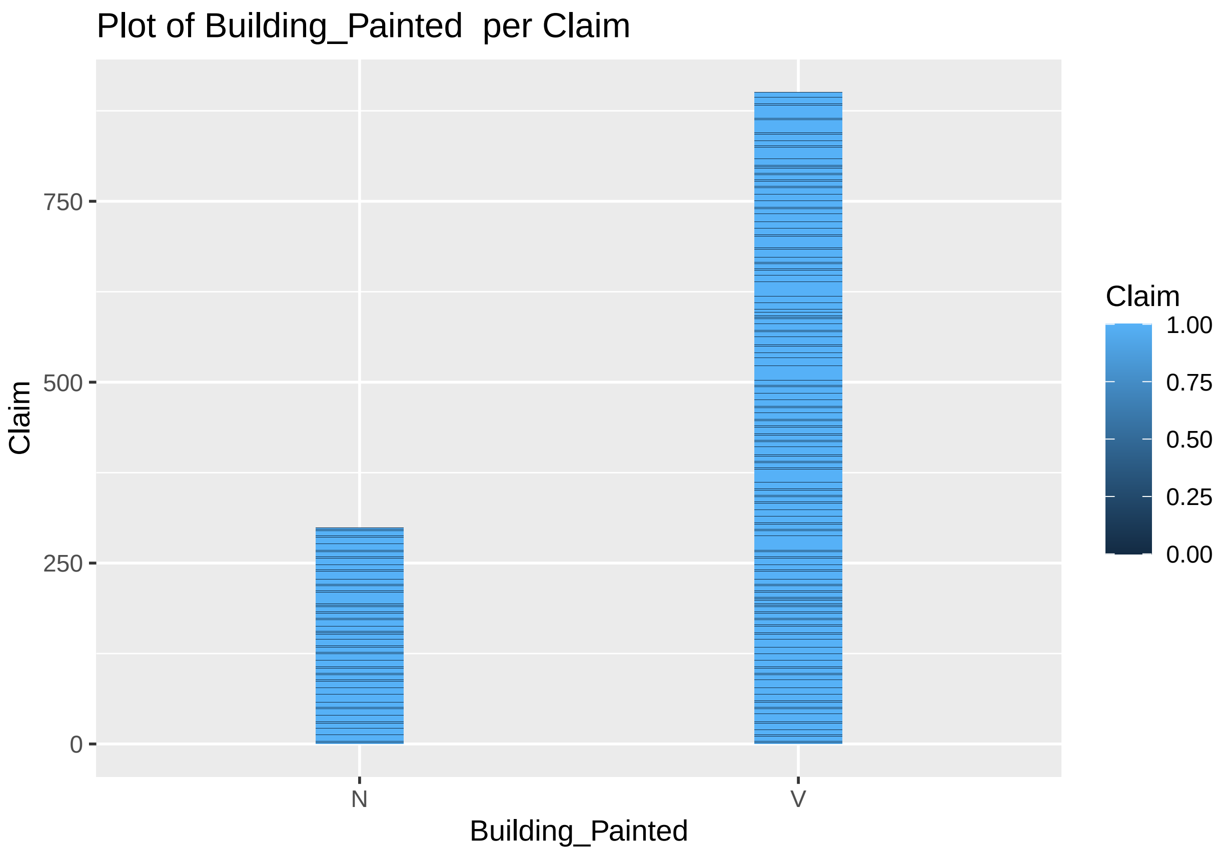
Most observations were done in 2012 and 2013.

**Bar graph showing type of Residential**



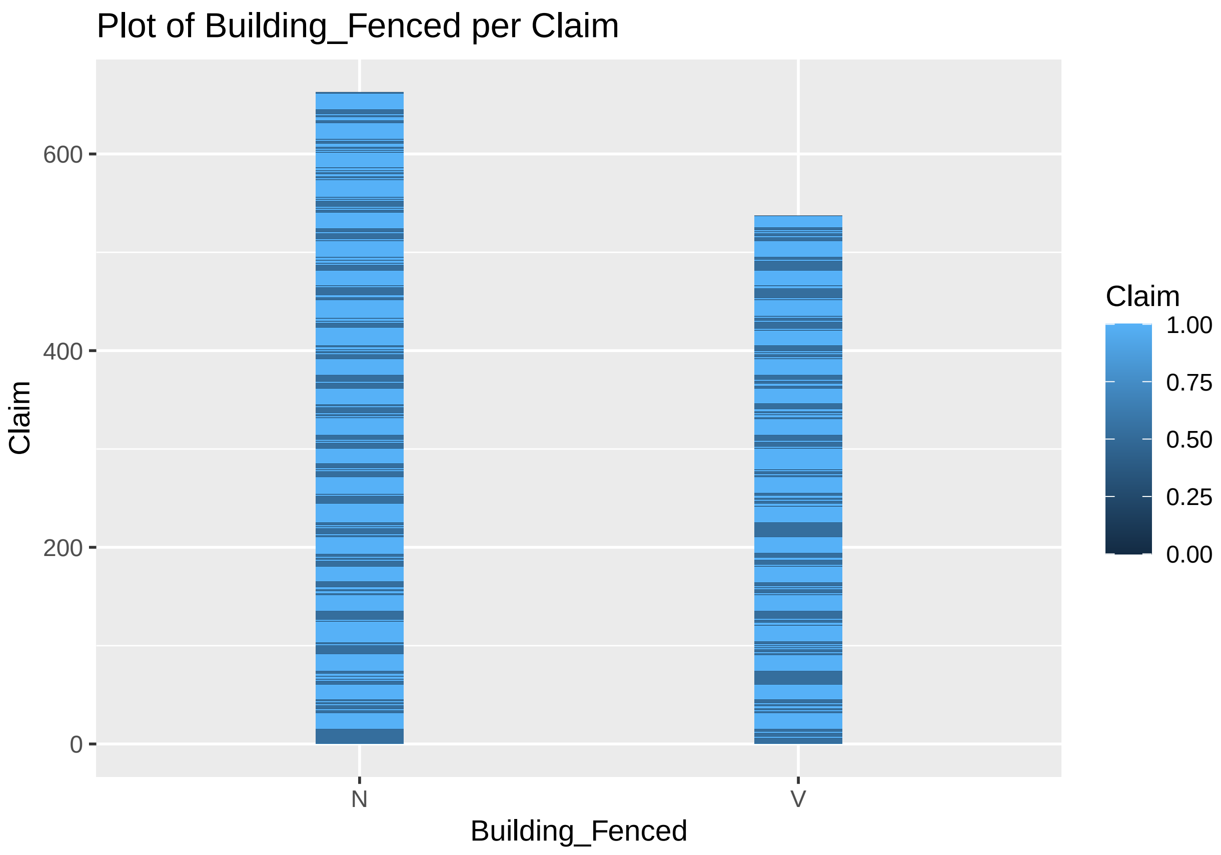
More buildings in this dataset are residential buildings represnted by 0.

**Bar graph showing number of Buildings painted or not**



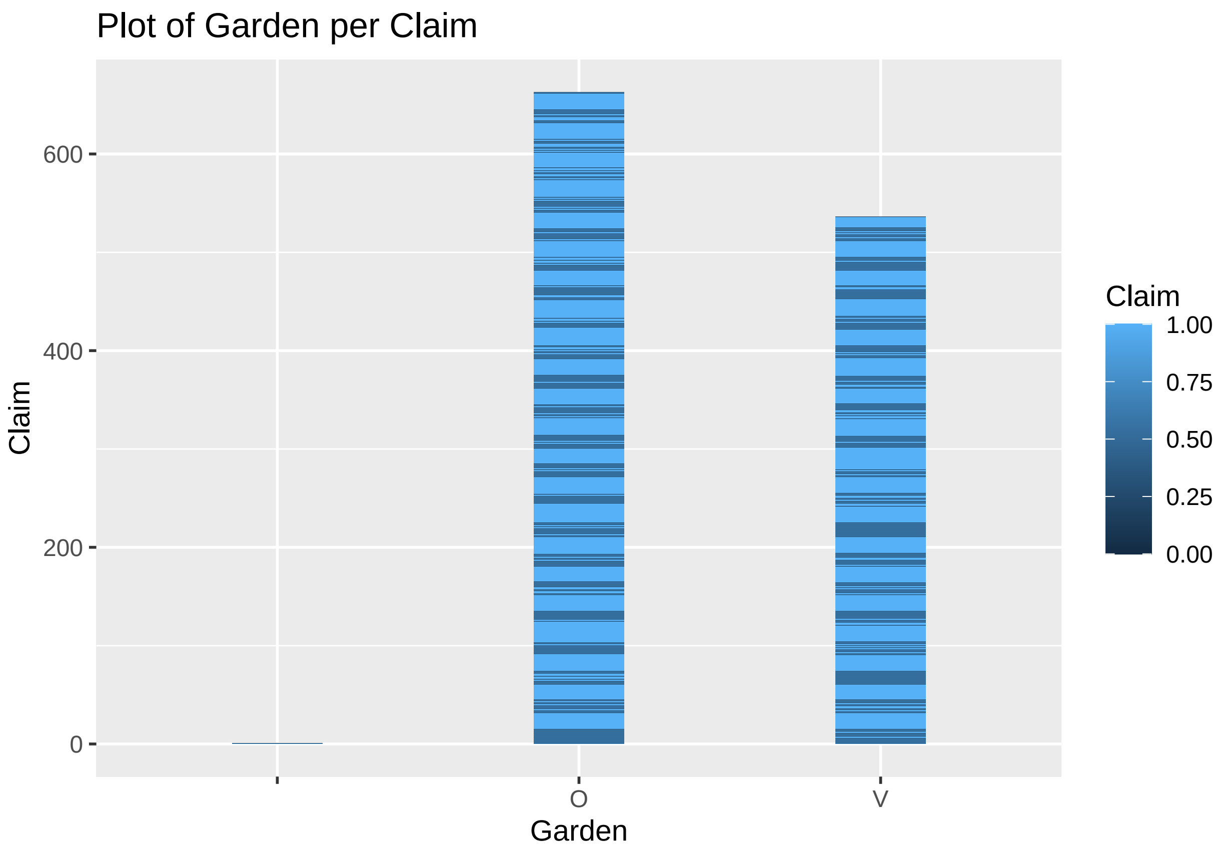
More buildings in this dataset are not painted.

**Bar graph showing number of Buildings fenced or not**



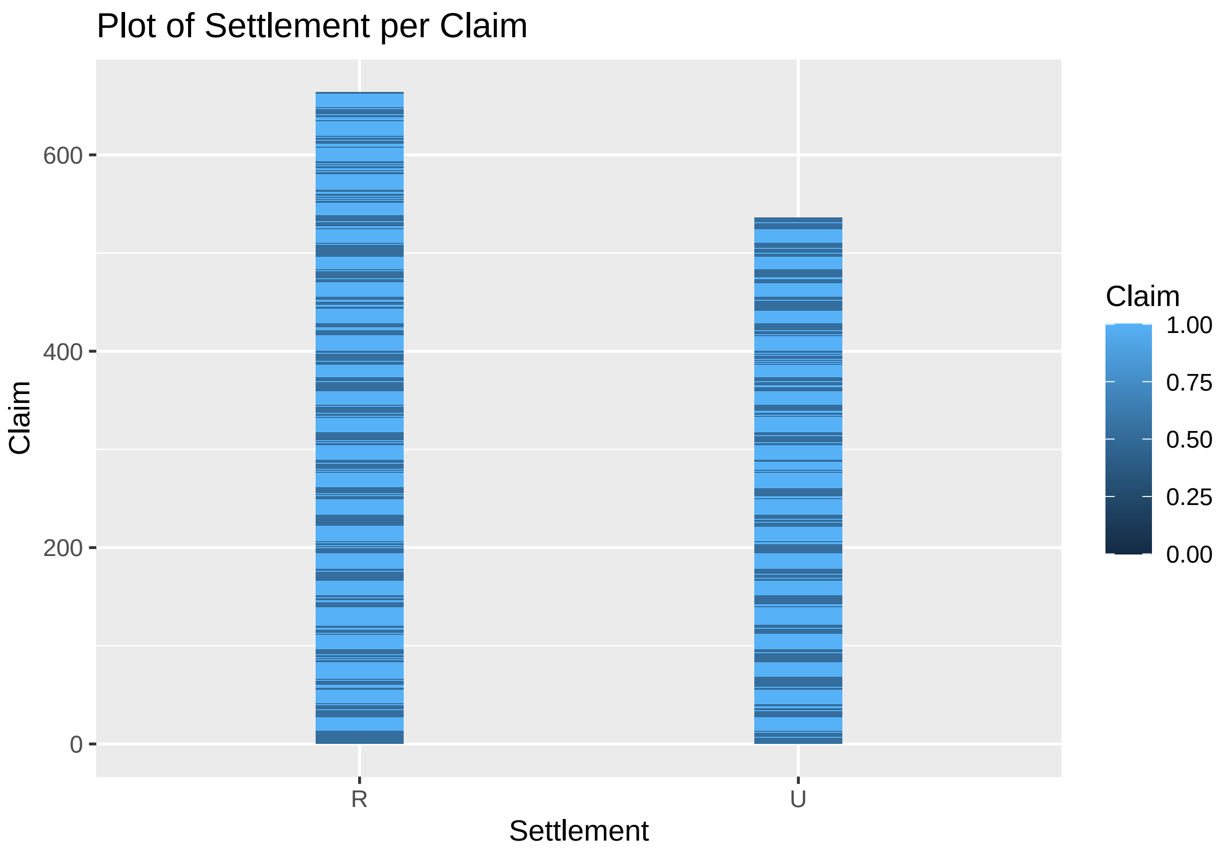
The bar graph shows that there are more Buildings that are fenced than not

**Bar graph showing number of Buildings with gardens or not.**



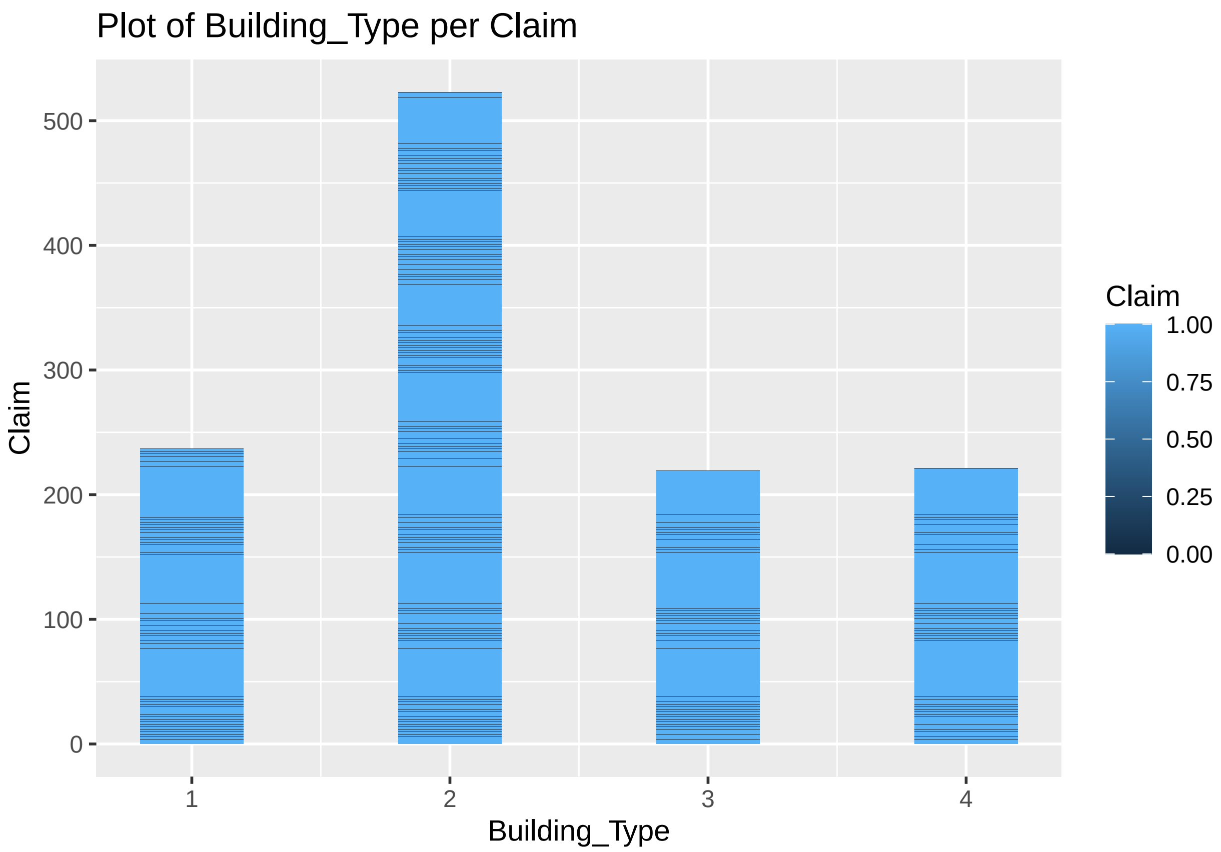
More buildings in this dataset do not have gardens.

**Bar graph showing the settlement type.**



More buildings in this dataset are from rural areas.

### Bar graph showing type of building.



We have 4 types of buldings with the majority of buildings being type 2.

Interpreting all the categorical variables above,it is clear that we more number of no claims than claims.

### Transforming data into numeric

I am going to transform the data it to numeric data type since it'll be more handy to use for the functions ahead.

Another important point to note. When converting a factor to a numeric variable, you should always convert it to character and then to numeric, else, the values can get screwed up,however this dataset is mixed data types,so i will just convert it to numeric.

```r
introduce(new_train)
```

```
##   rows columns discrete_columns continuous_columns all_missing_columns
## 1 5733      14                0                 14                   0
##   total_missing_values complete_rows total_observations memory_usage
## 1                    0          5733              80262       646008
```

As we can observe, there are no missing values left in the dataframe.

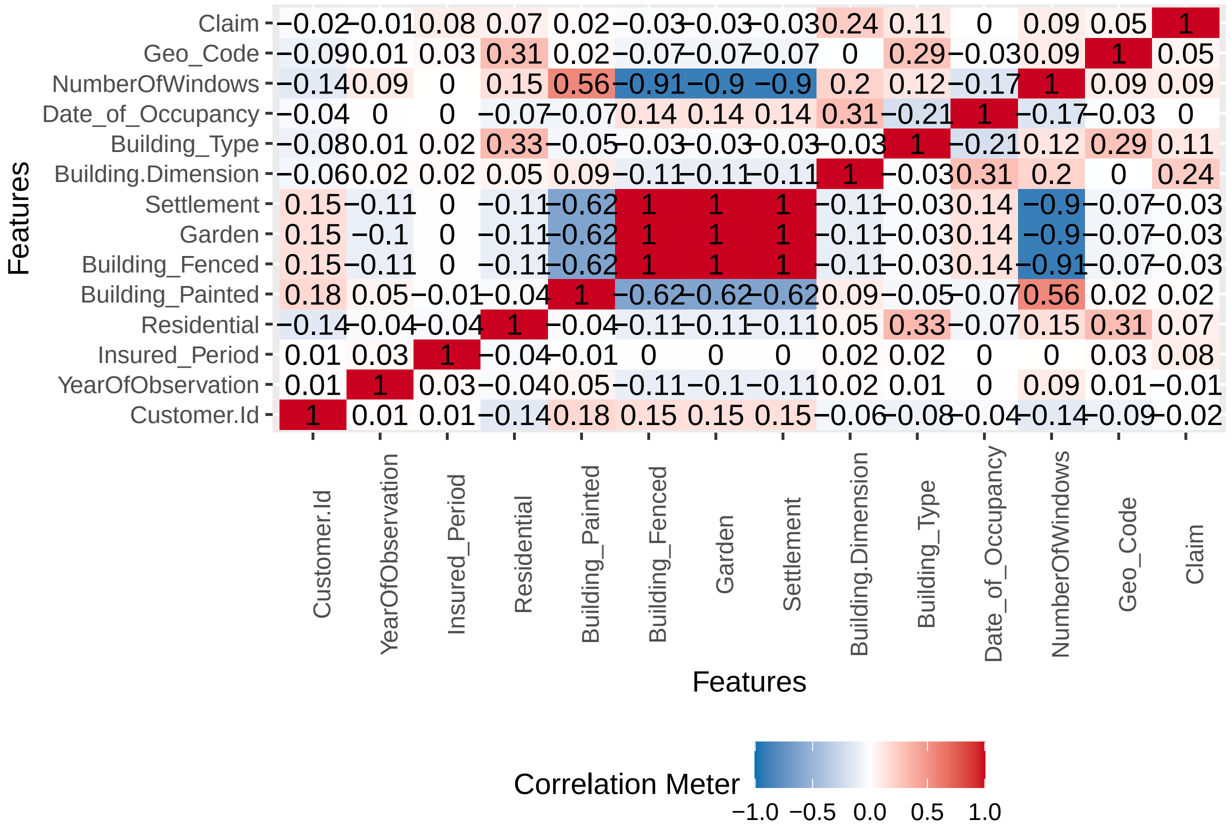
We'll now move on to multi-variate analysis of our variables and draw a correlation heat map from DataExplorer library.

## Heatmap

We'll now move on to multi-variate analysis of our variables and draw a correlation. The heatmap will enable us to find out the correlation between each variable. We are more interested in to find out the correlation between our predictor attributes with the target attribute Claim. The color scheme depicts the strength of correlation between 2 variables.

This will be a simple way to quickly find out how much an impact a variable has on our final outcome. There are other ways as well to figure this out.

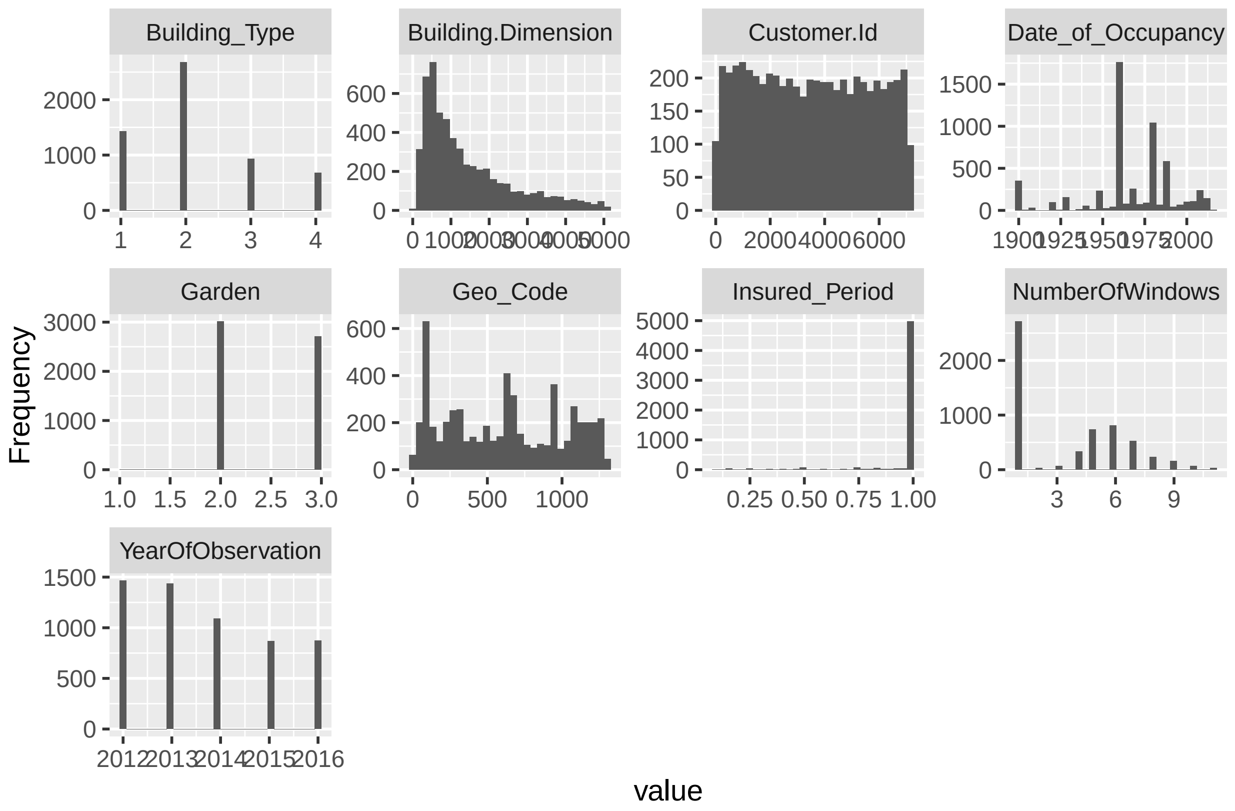
```
plot_correlation(na.omit(new_train), maxcat = 5L)
```



We can observe the week correlation of Customer.Id,YearOfObservation, Building_Panted and Date_of_Occupancy with our target variable.

Now let's have a univariate analysis of our variables. We'll start with the categorical variables and have a quick check on the frequency of distribution of categories. The code below will allow us to observe the required graphs.

```
plot_histogram(new_train)
```



The distribution above shows that most variables are moderately normal.This is a good indication about the predictor power of our variables.

## 4.Feature Engineering

This step can be more important than the actual model used because a machine learning algorithm only learns from the data we give it, and creating features that are relevant to a task is absolutely crucial.

Analyzing our data above, we've been able to note the extremely weak correlation of some variables with the final target variable. The following are the ones which have significantly low correlation values:Customer.Id,YearOfObservation,Building_Painted,Date_of_Occupancy,Building_Fenced,Garden,Settlement,Geo_Code

## 5.Standardization

In statistics, standardization is the process of putting different variables on the same scale. This process allows you to compare scores between different types of variables. Typically, to standardize variables, you calculate the mean and standard deviation for a variable. Then, for each observed value of the variable, you subtract the mean and divide by the standard deviation.

This process produces standard scores that represent the number of standard deviations above or below the mean that a specific observation falls. For instance, a standardized value of 2 indicates that the observation falls 2 standard deviations above the mean. This interpretation is true regardless of the type of variable that you standardize.

The target should not be standardized in this case as it is already in zeros' and ones.

```
data <-scale(data)
head(data)
```

```
##      Insured_Period Residential Building.Dimension Building_Type
## [1,]      0.3284331  -0.6353742         -0.9748406     -1.237809
## [2,]      0.3284331  -0.6353742         -0.7156744     -1.237809
## [3,]      0.3284331  -0.6353742          1.1919588     -1.237809
## [4,]      0.3284331  -0.6353742         -0.7666579     -1.237809
## [5,]      0.3284331  -0.6353742          1.1834616     -1.237809
## [6,]      0.3284331  -0.6353742          2.9865786     -1.237809
##      NumberOfWindows
## [1,]      -0.9543575
## [2,]      -0.9543575
## [3,]      -0.9543575
## [4,]       0.4776471
## [5,]      -0.9543575
## [6,]      -0.9543575
```

```
new_data <-mutate(data,new_train$Claim)
```

```
colnames(new_data)[6] <- "Claim"
new_data$Claim <-as.factor(new_data$Claim)
head(new_data)
```

```
##   Insured_Period Residential Building.Dimension Building_Type NumberOfWindows
## 1      0.3284331  -0.6353742         -0.9748406     -1.237809      -0.9543575
## 2      0.3284331  -0.6353742         -0.7156744     -1.237809      -0.9543575
## 3      0.3284331  -0.6353742          1.1919588     -1.237809      -0.9543575
## 4      0.3284331  -0.6353742         -0.7666579     -1.237809       0.4776471
## 5      0.3284331  -0.6353742          1.1834616     -1.237809      -0.9543575
## 6      0.3284331  -0.6353742          2.9865786     -1.237809      -0.9543575
##   Claim
## 1     0
## 2     0
## 3     0
## 4     0
## 5     0
## 6     0
```

## 6.Create training and testing sets

I will train our model using the fit method with training_data that cointants 70% and testing_data that contain 30% of our dataset. This will be a binary classification model.

## 7. Fitting Logistic Regression Model

Why not linear regression?

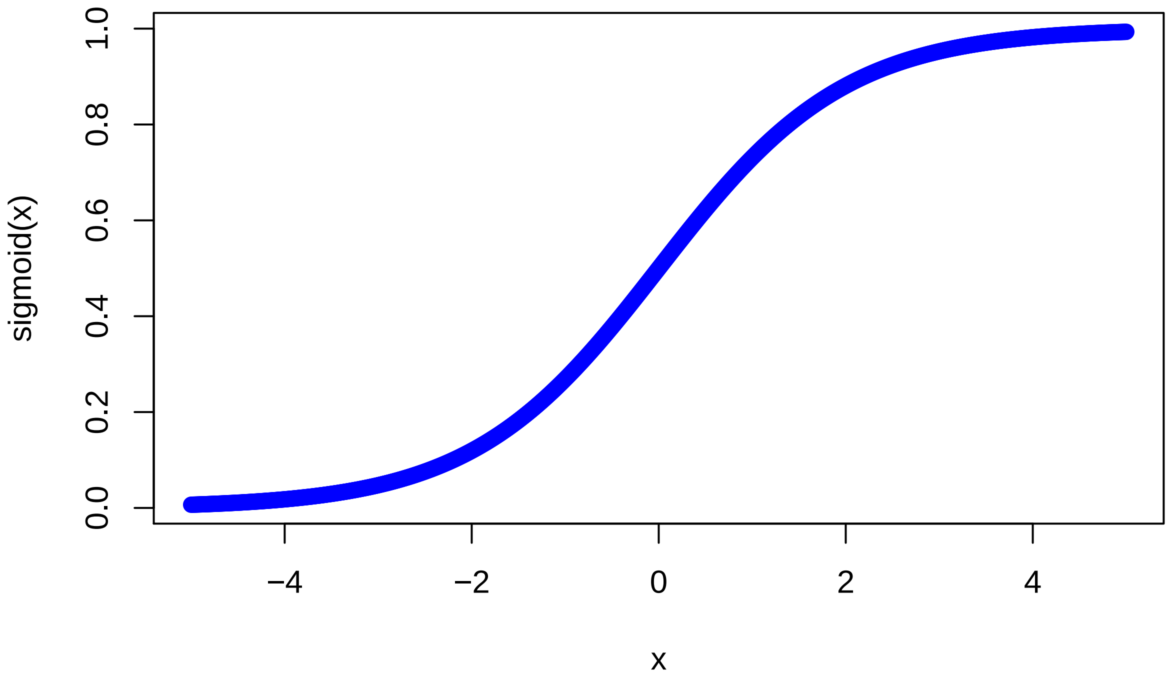
When the response variable has only 2 possible values, it is desirable to have a model that predicts the value either as 0 or 1 or as a probability score that ranges between 0 and 1. Linear regression does not have this capability. Because, If you use linear regression to model a binary response variable, the resulting model may not restrict the predicted Y values within 0 and 1.

In short, Logistic Regression is used when the dependent variable(target) is categorical. For example: 1. To predict whether an email is spam (1) or not spam (0) 2. Whether the tumor is malignant (1) or not (0)

It is named as 'Logistic Regression', because it's underlying technique is quite the same as Linear Regression. There are structural differences in how linear and logistic regression operate. Therefore, linear regression isn't suitable to be used for classification problems. The logistic or logit transformation for a proportion or rate $p$ is defined as: $f(p) = \log\left(\frac{p}{1-p}\right) =$ When $p$ is a proportion or probability, the quantity that is being logged $p/(1-p)$ is called the odds.Its name is derived from one of the core functions behind its implementation called the logistic function or the sigmoid function. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

The Sigmoid function in mathematics is defined as:

```
sigmoid = function(x) {
   1 / (1 + exp(-x))}
# you can use the equal sign as well, the programmer way.
x <- seq(-5, 5, 0.01)
plot(x, sigmoid(x), col='blue')
```



## Checking the accuracy of the model using train data:

So, after evaluating all our predictor variables, it is finally time to perform Predictive analytics. In this stage, we'll build a predictive model that will predict whether an individual Claims or not based on the predictor variables that we evaluated in the previous section.

To build this model I've made use of the glm method since we have to classify an outcome into either of the two classes, i.e: Claim or not:

I am also going to encode the response variable into a factor variable of 1's and 0's. Though, this is only an optional step.So whenever the Class is Claim, it will be 1 else it will be 0. Then, I am converting it into a factor.

```
training_data$Claim <- factor(training_data$Claim,levels=c(0,1))
testing_data$Claim <- factor(testing_data$Claim,levels=c(0,1))
```

The response variable Claim is now a factor variable and all other columns are numeric.

To evaluate the accuracy of the model, we're going to use a confusion matrix:

```
confusionMatrix (training_data$Claim, predict (training_glm, training_data))
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0    1
##          0 3105   68
##          1  768   72
##
##                Accuracy : 0.7917
##                  95% CI : (0.7788, 0.8042)
##     No Information Rate : 0.9651
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0927
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.80170
##             Specificity : 0.51429
##          Pos Pred Value : 0.97857
##          Neg Pred Value : 0.08571
##              Prevalence : 0.96511
##          Detection Rate : 0.77374
##    Detection Prevalence : 0.79068
##       Balanced Accuracy : 0.65799
##
##        'Positive' Class : 0
##
```

The output shows that our model calculates the income level of an individual with an accuracy of approximately 79%, which is a good number to begin with.

## Building the model

In this section of predicting insurance claims , I will fit my model.A logistic regression is used for modeling the outcome probability of a class such as pass/fail, positive/negative and in our case – Claim/no Claim. I will proceed to implement this model on our test data as follows –

```
log.model <- glm(Claim ~., data = training_data, family = binomial)
```

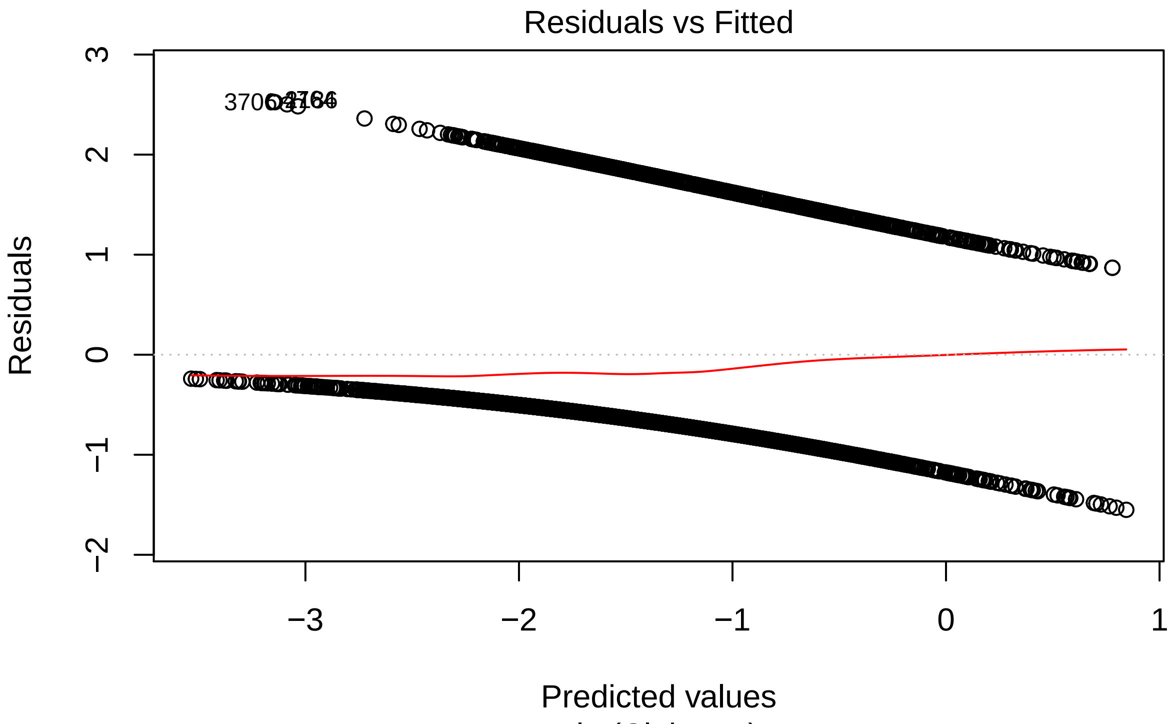
## Summary of the model

```r
summary(log.model)
```

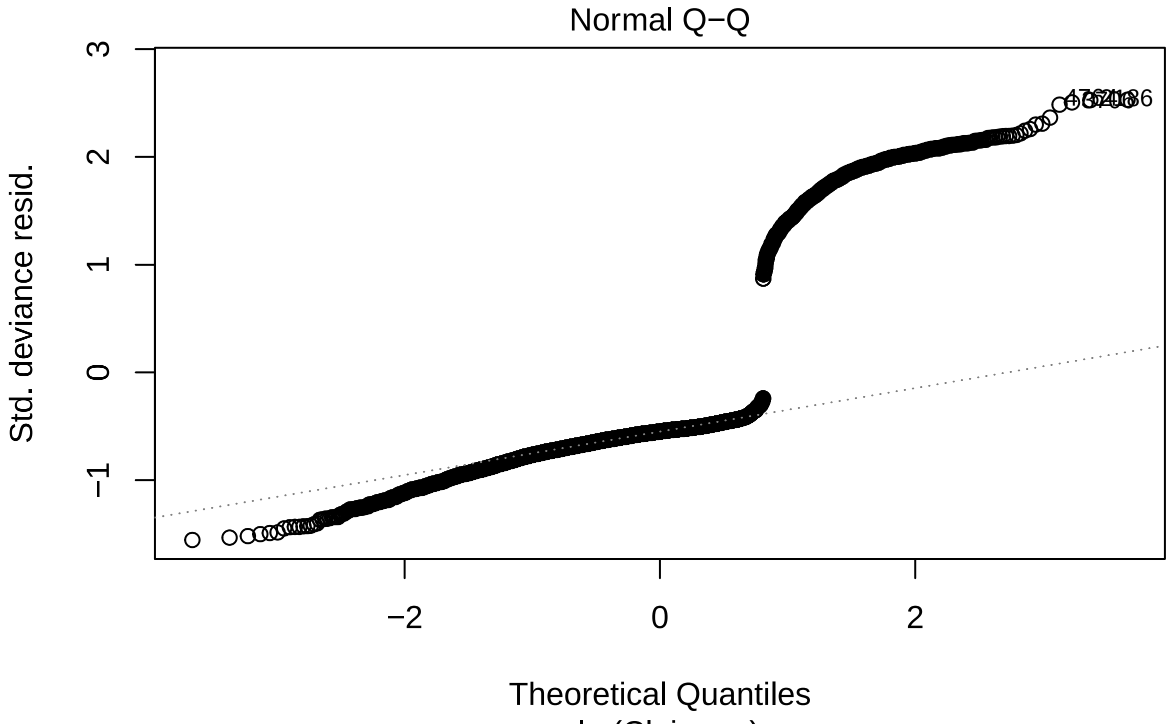
```
##
## Call:
## glm(formula = Claim ~ ., family = binomial, data = training_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5504  -0.6845  -0.5474  -0.4127   2.5271
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.44778    0.04276 -33.862  < 2e-16 ***
## Insured_Period      0.22491    0.05107   4.404 1.06e-05 ***
## Residential         0.05472    0.04174   1.311     0.19
## Building.Dimension  0.53710    0.03817  14.071  < 2e-16 ***
## Building_Type       0.31325    0.04172   7.509 5.98e-14 ***
## NumberOfWindows     0.06055    0.03996   1.515     0.13
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4117.8  on 4012  degrees of freedom
## Residual deviance: 3806.6  on 4007  degrees of freedom
## AIC: 3818.6
##
## Number of Fisher Scoring iterations: 4
```

Four predictor variables have p-values > than 0.05,implying that they are statistically insificant.These values had a significant low correlation but it makes sense for me to keep them as realistically they do have an effect on the target variable and more so we cannot just drop variables due to high p-values without further investigation.

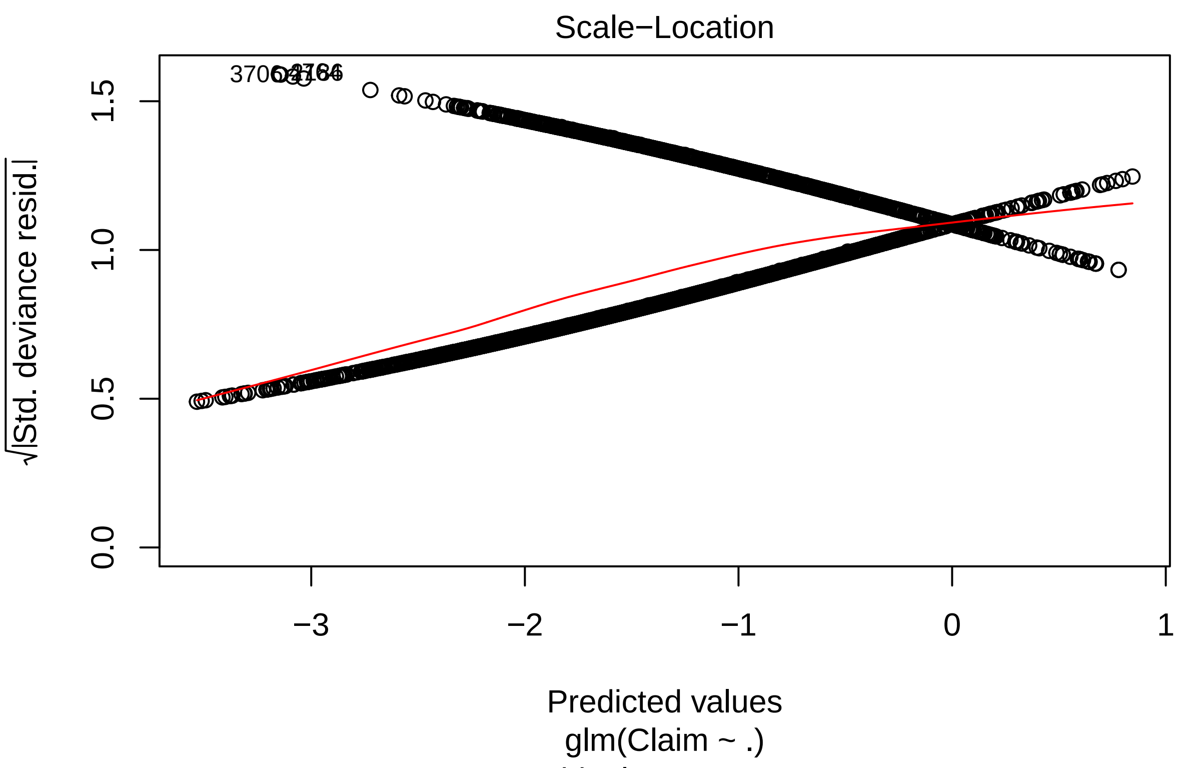
```r
plot(log.model)
```

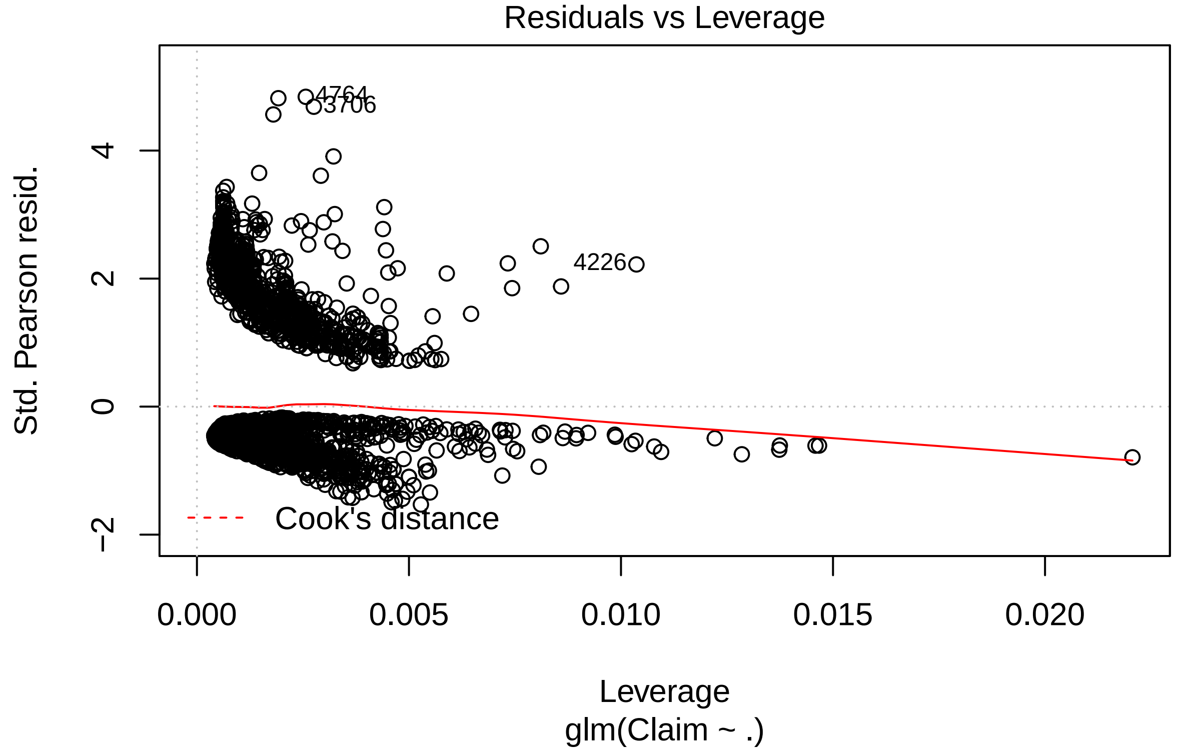


glm(Claim ~ .)



glm(Claim ~ .)





## Validate the model

So far, we used the training data set to build the model, now its time to validate the model by using the testing data set. The test data set is applied to the predictive model to validate the efficiency of the model. The following code snippets shows how this is done:

## 8. How to Predict on Testing Dataset

The log.model is now built.I will use it to predict the response on testing_data.

```
pred <- predict(log.model, newdata = testing_data, type = "response")
```

Now, pred contains the probability that the observation is malignant for each observation. Note that, when you use logistic regression, you need to set type='response' in order to compute the prediction probabilities.

### Recode factors

If the probability of Y is > 0.5, then it can be classified an event. So if pred is greater than 0.5, it is Claim else it is No Claim.

```
y_pred_num <- ifelse(pred > 0.5, 1, 0)
y_pred <- factor(y_pred_num)
y_act <- testing_data$Claim
```

### Accuracy

```
mean(y_pred == y_act) #81%
```

```
## [1] 0.7936047
```

I will try to improve the accuracy improving the inbalance between the two classes:Claim having 1200 values and no Claim having 4533 values as shown below:

```
table(training_data$Claim)
```

```
##
##    0    1
## 3173  840
```

## 9.Improving model accuracy by handling Class Imbalance with Up-sampling.

Since the response variable is a binary categorical variable, you need to make sure the training data has approximately equal proportion of classes.

This concern is normally handled with a couple of techniques called: a. Down Sampling b. Up Sampling c. Hybrid Sampling using

However for this project, I will use up sampling.in UpSampling, rows from the minority class, that is, Claim is repeatedly sampled over and over till it reaches the same size as the majority class (No Claim)

```r
'%ni%' <- Negate('%in%')  # define 'not in' func
set.seed(100)
up_train <- upSample(x = training_data[, colnames(training_data) %ni% "Claim"],
y = (training_data$Claim))
table(up_train$Class)
```

```
##
##    0    1
## 3173 3173
```

As expected, Claim and no Clain values are now in the same ratio.

### The model:

```r
up_log.model <- glm(Class ~., family = "binomial", data=up_train)
summary(up_log.model)
```

```
##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = up_train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.16825  -1.04309   0.02123   1.09922   2.04124
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.12748    0.02705  -4.713 2.44e-06 ***
## Insured_Period      0.24831    0.03240   7.664 1.80e-14 ***
## Residential         0.05394    0.02822   1.912   0.0559 .
## Building.Dimension  0.55601    0.02741  20.286  < 2e-16 ***
## Building_Type       0.31787    0.02847  11.165  < 2e-16 ***
## NumberOfWindows     0.04469    0.02678   1.669   0.0951 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8797.4  on 6345  degrees of freedom
## Residual deviance: 8053.7  on 6340  degrees of freedom
## AIC: 8065.7
##
## Number of Fisher Scoring iterations: 4
```

```r
up_scale_pred <- predict(up_log.model, newdata = testing_data, type = "response")
```

### Up scale Accuracy

```r
up_scale_pred_num <- ifelse(up_scale_pred > 0.5, 1, 0)
up_scale_pred <- factor(up_scale_pred_num, levels=c(0, 1))
up_scale_act <- testing_data$Claim
mean(up_scale_pred == up_scale_act)
```

```
## [1] 0.6534884
```

In order to assess the performance of our model, we will delineate the ROC curve. ROC is also known as Receiver Optimistic Characteristics. For this, we will first import the ROC package and then plot our ROC curve to analyze its performance.

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

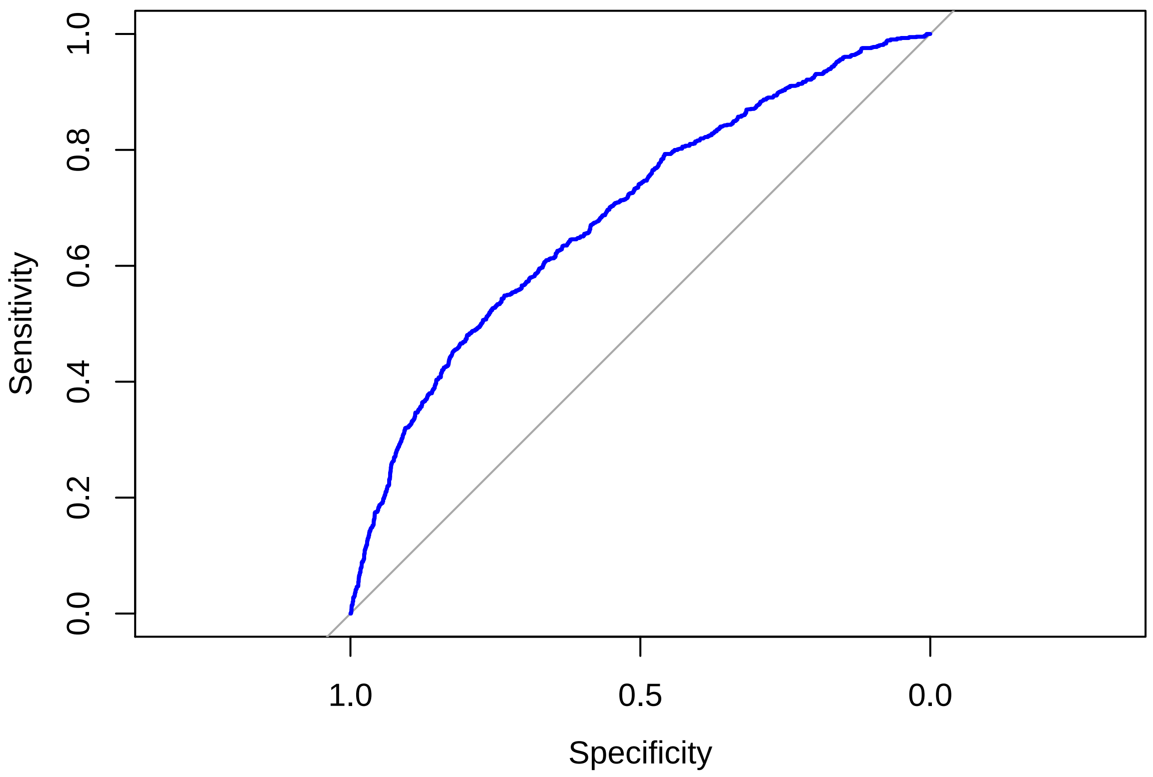
```
## The following object is masked from 'package:colorspace':
##
##     coords
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
lr.predict <- predict(up_log.model,up_train, probability = TRUE)
auc.gbm = roc(up_train$Class, lr.predict, plot = TRUE, col = "blue")
```

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

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



## 9. Conclusion

Logistic regression is a powerful tool, especially in epidemiologic studies,financial modelling and social sciences allowing multiple explanatory variables being analyzed simultaneously, meanwhile reducing the effect of confounding factors. However, researchers must pay attention to model building, avoiding just feeding software with raw data and going forward to results.

Building the model and classifying the Y is only half work done because, the scope of evaluation metrics to judge the efficacy of the model is vast and requires careful judgement to choose the right model. For this project,I only used logistic model as an ML technique,however they are various models that can be used to fit this data such as Gradient Boosting (GBM),Artificial Neural Network,Random forests and KNN techniques.

## 10. References

1. https://data-flair.training/blogs
2. https://rafalab.github.io/dsbook/large-datasets
3. https://zindi.africa/competitions