UNIVERSITY OF SURREY

THE FACULTY OF ARTS AND SOCIAL SCIENCES

Surrey Business School

MSc Business Analytics

Academic Year: 2018-2019

# DATA ANALYTICS

MANM301

COURSEWORK

Submitted On:
Monday 28th January 2019

| Student URN | MSc Programme | Surname | First Name |
|---|---|---|---|
| 6562752 | MSc Business Analytics | DIXIT | AMOL |

## Table of Contents

## List of Figures

## List of Tables

## Project Specification

This project is pursued to enhance the profitability of the Caravan Insurance Company by selling their insurance policies to prospective customers. The project tries to model through a Machine Learning approach a prediction whether a customer will purchase the Caravan Insurance based on their socio-demographic and personal features. Further, this project creates the profile of a customer who is likely to buy the Caravan Insurance based on their location and personal details so that the marketing efforts may be directed to capture maximum positive responses. Having a marketing campaign that wisely targets the eligible customers would not only bring higher income to the company, but also improve the company and customers' relationship through personalised services.

## Resourcing

The data set supplied by the Dutch Data Mining Company (Sentinal Machine Research) contains 9822 customer records with 86 features having details of customers over a widespread geographical market with respect to their insurance, interests, education, income levels and other social aspects. The dataset is divided into two parts for the project, training data (5822 records) and test data (4000 records). From first look at the data we realise that only a very small percentage of persons buy the insurance policy which is evident from the training data set where out of the 5822 persons only 348 (~ 6 %) are positive responses.

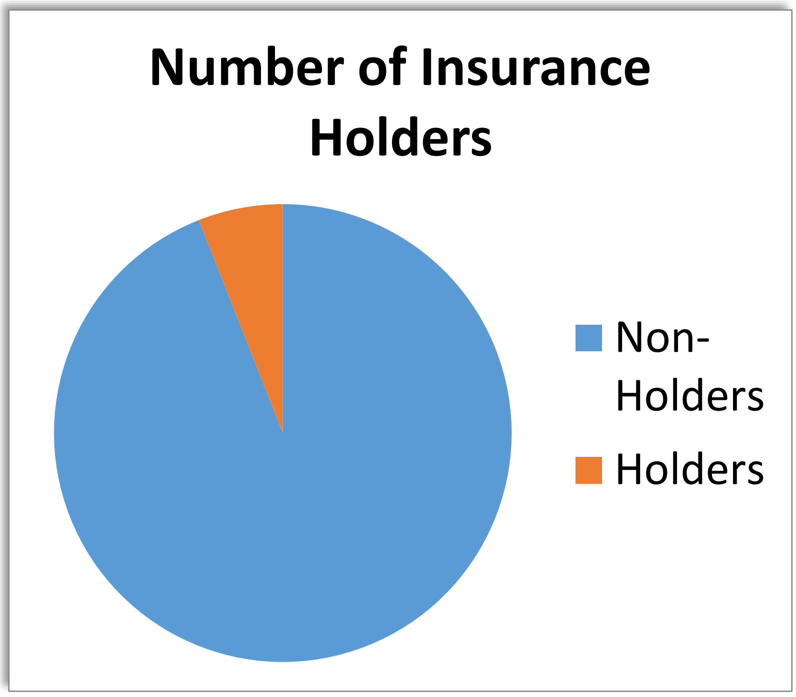


*Fig 1. Pie chart showing the ratio of Caravan Insurance policy holders to non-holders.*

Several techniques were used to model this data such as decision tree, logical regression, and random forest, the results and recommendations from which are in this report. For creating the profile of the customer C5.0 algorithm is employed to create a decision tree and generate rule sets accordingly.

Tools used in this project are R Studio, MS Excel, E-views, Tableau, SPSS Modeler, MS PowerPoint and MS Word.

## Data Cleansing & Data Exploration

Of the provided dataset 43 features represent socio-demographic fields and the remaining personal details of the prospective buyers, the remaining features referred to policy data. The fields are either numerical or categorical in nature.

### Missing Data Handling

There is no inconsistency in the data and the all fields are complete in all accords, thus no missing data handling is required.

### Correlation among Features

The data was reviewed thoroughly to capture if there were any fields which relayed similar information. Using these fields in conjunction would lead to the issue of multi-collinearity as they would not be independent in nature. For this the Spearman's Collinearity Coefficient was calculated (shown in Appendix A). Fields with high correlation amongst themselves are combined and employed in the data mining, some examples of this are:

| **Feature 1 (F1)** | **Feature 2 (F2)** | **Correlation Coefficient (F1, F2)** |
|---|---|---|
| Rented House (MHHUUR) | Home Owners (MHKOOP) | -1.0 |
| Number of Car Policies (APERSAUT) | Contribution of Car Policies (PPERSAUT) | +0.9 |
| Number of Boat Policies (APLEZIER) | Contribution of Boat Policies (PPLEZIER) | +1.0 |
| Number of Fire Policies (ABRAND) | Contribution of Fire Policies (PBRAND) | +0.9 |

*Table 1: High Correlation between fields*

Also, to better understand and explore the data we chart (on histograms) the policy holders and non-holders against the percentage of customers for the relevant features which were found during the Data Mining phase of the project (Appendix B).

## Pre-Processing

Next we move on to the stage of Pre-processing to gain a deeper insight about the data before we apply Data Mining algorithms to it and find relevant models and infer results from them.

### Feature Shortlisting

To refrain from creating a highly complex model with a chance of overfitting only a limited number of features would be employed in this project. To find the relevant fields their correlation with the target field was calculated (Appendix C). We find that no field impacts the final target field strongly, however fields related to the policy information (car policy, boat policy, and fire policy) have a much greater influence on the target than the demographical information. From this we have basic information that the Caravan Insurance holders are more likely to hold other policies as well which will be explored in the further stages of the project.

### Creation of Variable Subsets and Dummy Variables

From the previous stages we get an idea as to the information available in our dataset, now we also create Information Gain and Chi-Squared Metrics to find out which features are most relevant to this data analytics project. The metric results are published in Appendix D.

Also, during our data mining activity we found that a specific income class has an effect on the final decision to purchase a policy or not. This class has hence being created into a dummy variable called as (MINK3045.4) which is binary in nature.

### Balancing

The dataset in its raw format is highly biased towards non-buyers. So in order to have better prediction accuracy balancing is done on the database. We balance the data by both randomly deleting non-purchaser records (under sampling) and also by randomly duplicating the purchaser records (over sampling) until such a time that there is significant balance of both in the data. The levels are chosen empirically based on algorithm performance for over sampling as 1:6 and for under sampling as 6.2:1

## Data Mining

In this stage we build models using we which we can predict whether a person will purchase the Caravan Insurance policy or not. These models are judged on their efficiency primarily to maximise the True Positives and minimise the False Negatives. The Decision Tree Model is used to find the rules against which the customer profile is built upon. These rules are assessed on their accuracy and coverage potential.

### Logistic Regression Model (Code: Appendix I)

The LR approach was first applied to train the algorithm with all features (Performance Diagnostics and ROC curve in Appendix E). This model was then balanced and its performance diagnostics were captured. As this would be a very complex model there are chances of overfitting and multi-collinearity when it is applied to the test data, so the features were dropped to enhance the efficiencies while also keeping minimising of complexity and overfitting in mind.

The performance (Table 2) of the model with features (listed below) provides similar results as the overtly complex model with all features, so we can rightfully say that the subset of features is a good selection. Feature interactions that maximise the predictive accuracy of the model have been identified by training the model with different feature combinations and applying the results obtained from the methods described in the data cleansing and pre-processing stages. The following subset was selected as the most predictive to the logistic regression model:

1. Number of Car policies
2. Number of fire policies
3. Number of boat policies
4. Purchasing power
5. Married
6. Lower Level Education

With these features normal model has the highest accuracy and precision (as can be seen in below table), however the sensitivity (0.84%) for this case is very low. To maximise model efficiency, sensitivity requires major enhancement, which provides the reason for the creation of two new models with over and under sampled data. Both these models have much higher sensitivity.

| | **Normal** | **Under Sampled (Level 6.2 : 1)** | **Over Sampled (Level 1 : 6)** |
|---|---|---|---|
| **Accuracy** | 94.03% | 89.05% | 89.40% |
| **Precision** | 40.00% | 20.06% | 20.19% |
| **Sensitivity** | 0.84% | 28.15% | 26.47% |
| **Specificity** | 99.92% | 92.90% | 93.38% |
| **1-Specificity** | 0.08% | 7.10% | 6.62% |

*Table 2: Performance Diagnostics of LR Model*

Over and under sampled models have better results albeit at the expense of 5% accuracy, however this simpler model with 6 features has better characteristics than the unbalanced model with all fields and thus are chosen due to their simplicity and better performance.

Appendix F shows the ROC curve and the logistic regression coefficients for the all three models. The coefficients that are more significant (small p-values) and cause a higher impact on the target field are as follows:

1. Number of boat policies
2. Number of car policies
3. Number of fire policies

Upon analysing the coefficients in Appendix F we find them to in coherence with the histograms in Appendix B, thus validating the LR model in this project.

## Decision Tree Model (Code: Appendix I)

The decision tree model is primarily employed to classify and find whether a person would purchase the policy or not. This method is also used to find the rule sets based upon which the personal profile is built of the prospective customer.

In this approach as well, a model was built using several features (Appendix G) and it was suspected that the complex model had a high chance of overfitting as the rule sets did not make much business sense. To simplify the model a smaller set of features was selected and the tree in Fig.2 below was generated.

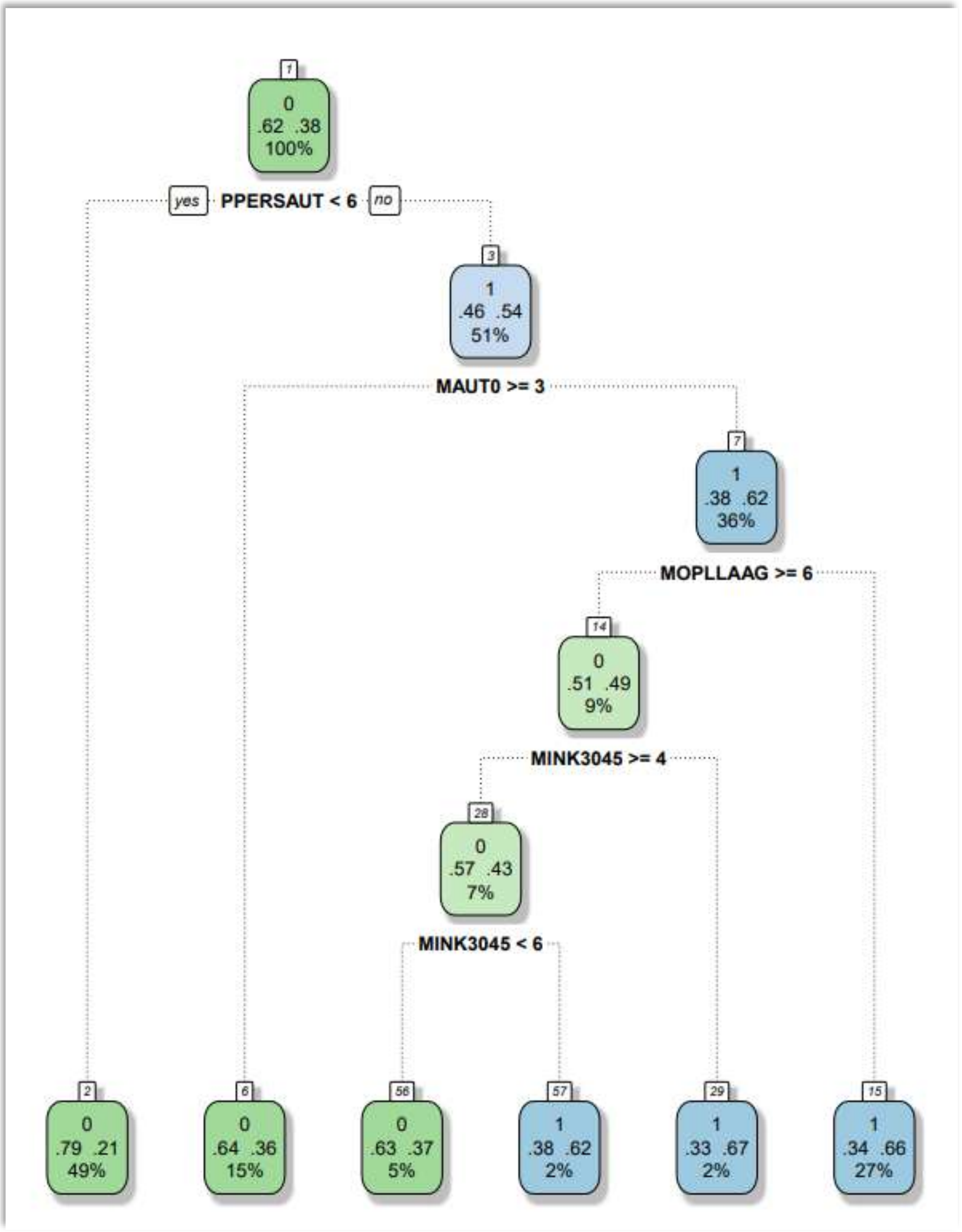


*Fig.2: Decision Tree of the Caravan Insurance purchasers and non-purchasers.*

The efficiency characteristics and the related confusion matrix of the model is shown below.

| | |
|---|---|
| **Accuracy** | 66.44% |
| **Precision** | 61.71% |
| **Sensitivity** | 53.72% |
| **Specificity** | 75.71% |
| **1-Specificity** | 1.29% |

*Table 3: Performance Diagnostics*

| Actual \ Predicted | Yes | No |
|---|---|---|
| **Yes** | TP= 166 | FN= 143 |
| **No** | FP= 103 | TN= 321 |

*Table 4: Confusion Matrix*

Rule Sets obtained from this Tree and their corresponding coverage are listed in the below table:

| **Rule Sets** | | **Coverage** |
|---|---|---|
| if PPERSAUT < 6 | Does Not Buy | 49% |
| if PPERSAUT >=6 and MAUTO >=3 | Does Not Buy | 15% |
| if PPERSAUT >=6 and MAUTO < 3 and MOPLLAAG >=6 and MINK3045 is 4 or 5 | Does Not Buy | 5% |
| if PPERSAUT >=6 and MAUTO < 3 and MOPLLAAG < 6 | Buys | 27% |
| if PPERSAUT >=6 and MAUTO < 3 and MOPLLAAG >=6 and MINK3045 < 4 | Buys | 2% |
| if PPERSAUT >=6 and MAUTO < 3 and MOPLLAAG >=6 and MINK3045 is 6 and above | Buys | 2% |
| | **Total Coverage** | 100% |

*Table 5: Rule sets and their Coverage*

## Bagging and Random Forest (Code: Appendix I)

Another model of Bagging and Random Forest is employed to further explore the data and see if more insights could be extracted from it. We first do the bagging of the data to improve the stability and accuracy of the random forest algorithm used for regression. It also reduces variance and helps in avoiding overfitting. After this the Random Forest algorithm is applied (the number of trees used was 90 and the node size , 9) and a classification is obtained (Appendix H shows one of the generated trees).

The Random Forest algorithm employs the features which were used in the previous methods as well. This also substantiates the previous models. The features used are:

| | | |
|---|---|---|
| 1. | PPERSAUT | Contribution of Car Policies |
| 2. | MINK3045 | Income between 30,000 and 45,000 |
| 3. | MAUTO0 | No car |
| 4. | MOPLLAAG | Lower Education Level |

The confusion matrix for this model is as below:

| **Actual** \ **Predicted** | **Yes** | **No** |
|---|---|---|
| **Yes** | TP= 91 | FN= 84 |
| **No** | FP= 57 | TN= 206 |

| | |
|---|---|
| **Accuracy** | 67.09% |
| **Precision** | 61.49% |
| **Sensitivity** | 52.03% |
| **Specificity** | 78.33% |
| **1-Specificity** | 21.67% |

*Table 6: Confusion Matrix and Performance Diagnostics of the Random Forest Model*

The performance results are better than decision tree but the complexity of the model makes it very difficult to extract some useful patterns so as to gain benefit in business.

## Exploration and Customer Profile:

From the above explored data mining algorithms and analysis, we can make a fair prediction about the profile of a typical customer who buys the Caravan Insurance policy. However, it must be kept in mind that these are not necessary predictors of someone to purchase the policy, but are sufficient enough to assume that a person would buy the policy. So someone with or without a boat policy is also expected to purchase the insurance.

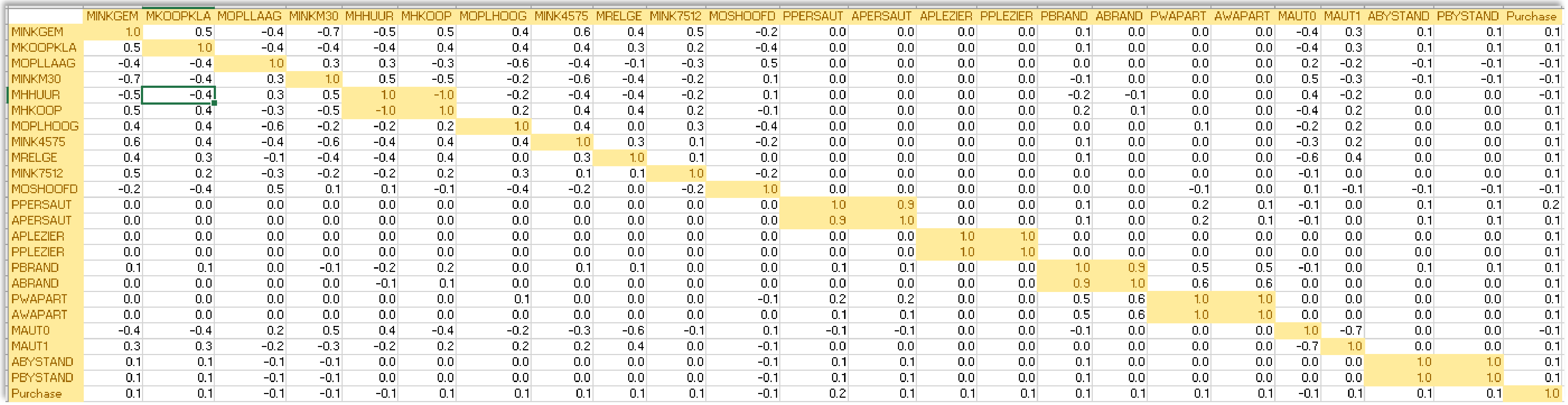
Persons who are more likely to buy the Caravan Insurance policy are generally:

1. Persons of high income groups who possess high purchasing power.
2. Inhabitants of high-education neighbourhoods.
3. Those who have more than one car and often own a boat as well.
4. Those who have third party insurance.
5. Those having a car policy, boat policy and/or fire policy for protection. Thus we can assume that persons who are risk averse by nature would buy the Caravan policy.
6. Married with more than one child.
7. Middle-aged persons. Who can sustain a lavish lifestyle.
8. Traditional families living in small cities.
9. High status retirees: Senior citizens who and are maintaining a high standard of living.

*Fig.3: A mix of persons who are likely to buy Caravan Insurance.*

# Appendix

## Appendix A: Spearman's Collinearity Coefficient



| | MINKGEM | MKOOPKLA | MOPLLAAG | MINKM30 | MHHUUR | MHKOOP | MOPLHOOG | MINK4575 | MRELGE | MINK7512 | MOSHOOFD | PPERSAUT | APERSAUT | APLEZIER | PPLEZIER | PBRAND | ABRAND | PWAPART | AWAPART | MAUT0 | MAUT1 | ABYSTAND | PBYSTAND | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MINKGEM | 1.0 | 0.5 | -0.4 | -0.7 | -0.5 | 0.5 | 0.4 | 0.6 | 0.4 | 0.5 | -0.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 | -0.4 | 0.3 | 0.1 | 0.1 | 0.1 |
| MKOOPKLA | 0.5 | 1.0 | -0.4 | -0.4 | -0.4 | 0.4 | 0.4 | 0.4 | 0.3 | 0.2 | -0.4 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 | -0.4 | 0.3 | 0.1 | 0.1 | 0.1 |
| MOPLLAAG | -0.4 | -0.4 | 1.0 | 0.3 | 0.3 | -0.3 | -0.6 | -0.4 | -0.1 | -0.3 | 0.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.2 | -0.2 | -0.1 | -0.1 | -0.1 |
| MINKM30 | -0.7 | -0.4 | 0.3 | 1.0 | 0.5 | -0.5 | -0.2 | -0.6 | -0.4 | -0.2 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | -0.1 | 0.0 | 0.0 | 0.0 | 0.5 | -0.3 | -0.1 | -0.1 | -0.1 |
| MHHUUR | -0.5 | -0.4 | 0.3 | 0.5 | 1.0 | -1.0 | -0.2 | -0.4 | -0.4 | -0.2 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | -0.2 | -0.1 | 0.0 | 0.0 | 0.4 | -0.2 | 0.0 | 0.0 | -0.1 |
| MHKOOP | 0.5 | 0.4 | -0.3 | -0.5 | -1.0 | 1.0 | 0.2 | 0.4 | 0.4 | 0.2 | -0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.2 | 0.1 | 0.0 | 0.0 | -0.4 | 0.2 | 0.0 | 0.0 | 0.1 |
| MOPLHOOG | 0.4 | 0.4 | -0.6 | -0.2 | -0.2 | 0.2 | 1.0 | 0.4 | 0.0 | 0.3 | -0.4 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.0 | -0.2 | 0.2 | 0.0 | 0.0 | 0.1 |
| MINK4575 | 0.6 | 0.4 | -0.4 | -0.6 | -0.4 | 0.4 | 0.4 | 1.0 | 0.3 | 0.1 | -0.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 | -0.3 | 0.2 | 0.0 | 0.0 | 0.1 |
| MRELGE | 0.4 | 0.3 | -0.1 | -0.4 | -0.4 | 0.4 | 0.0 | 0.3 | 1.0 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 | -0.6 | 0.4 | 0.0 | 0.0 | 0.1 |
| MINK7512 | 0.5 | 0.2 | -0.3 | -0.2 | -0.2 | 0.2 | 0.3 | 0.1 | 0.1 | 1.0 | -0.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -0.1 | 0.0 | 0.0 | 0.0 | 0.1 |
| MOSHOOFD | -0.2 | -0.4 | 0.5 | 0.1 | 0.1 | -0.1 | -0.4 | -0.2 | 0.0 | -0.2 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -0.1 | 0.0 | 0.1 | -0.1 | -0.1 | -0.1 | -0.1 |
| PPERSAUT | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.9 | 0.0 | 0.0 | 0.1 | 0.0 | 0.2 | 0.1 | -0.1 | 0.0 | 0.1 | 0.1 | 0.2 |
| APERSAUT | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.9 | 1.0 | 0.0 | 0.0 | 0.1 | 0.0 | 0.2 | 0.1 | -0.1 | 0.0 | 0.1 | 0.1 | 0.1 |
| APLEZIER | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 |
| PPLEZIER | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 |
| PBRAND | 0.1 | 0.1 | 0.0 | -0.1 | -0.2 | 0.2 | 0.0 | 0.1 | 0.1 | 0.0 | 0.0 | 0.1 | 0.1 | 0.0 | 0.0 | 1.0 | 0.9 | 0.5 | 0.5 | -0.1 | 0.0 | 0.1 | 0.1 | 0.1 |
| ABRAND | 0.0 | 0.0 | 0.0 | 0.0 | -0.1 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.9 | 1.0 | 0.6 | 0.6 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 |
| PWAPART | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 | -0.1 | 0.2 | 0.2 | 0.0 | 0.0 | 0.5 | 0.6 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 |
| AWAPART | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 | 0.1 | 0.0 | 0.0 | 0.5 | 0.6 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.1 |
| MAUT0 | -0.4 | -0.4 | 0.2 | 0.5 | 0.4 | -0.4 | -0.2 | -0.3 | -0.6 | -0.1 | 0.1 | -0.1 | -0.1 | 0.0 | 0.0 | -0.1 | 0.0 | 0.0 | 0.0 | 1.0 | -0.7 | 0.0 | 0.0 | -0.1 |
| MAUT1 | 0.3 | 0.3 | -0.2 | -0.3 | -0.2 | 0.2 | 0.2 | 0.2 | 0.4 | 0.0 | -0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -0.7 | 1.0 | 0.0 | 0.0 | 0.1 |
| ABYSTAND | 0.1 | 0.1 | -0.1 | -0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -0.1 | 0.1 | 0.1 | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.1 |
| PBYSTAND | 0.1 | 0.1 | -0.1 | -0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -0.1 | 0.1 | 0.1 | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.1 |
| Purchase | 0.1 | 0.1 | -0.1 | -0.1 | -0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | -0.1 | 0.2 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | -0.1 | 0.1 | 0.1 | 0.1 | 1.0 |

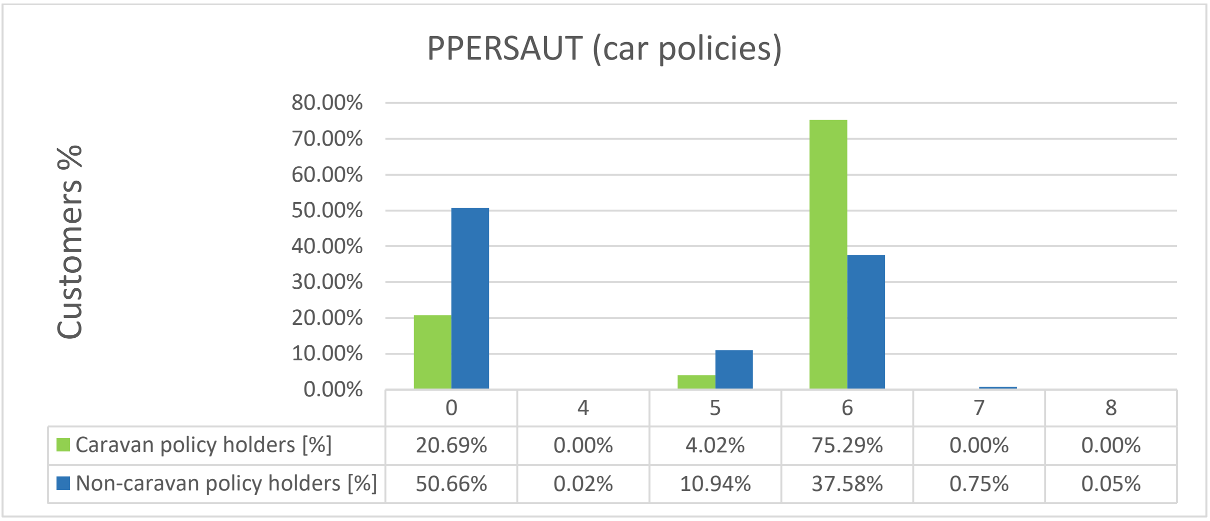
*Table 7: Spearman's Correlation Coefficient*

[Back]

## Appendix B: Histograms

The histograms represent the feature level percentage for Caravan and Non-Caravan Insurance holders

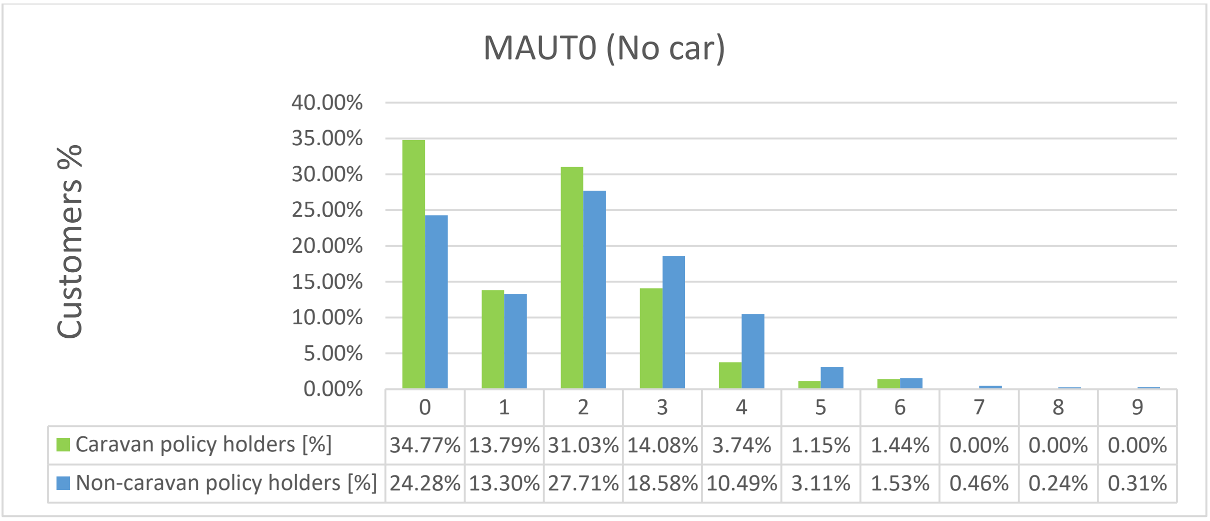
i. Contribution of car policies



| | 0 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|
| 🟩 Caravan policy holders [%] | 20.69% | 0.00% | 4.02% | 75.29% | 0.00% | 0.00% |
| 🟦 Non-caravan policy holders [%] | 50.66% | 0.02% | 10.94% | 37.58% | 0.75% | 0.05% |

*Fig.4: Effect of Contribution of Car Policies on policy purchase.*

ii. No Car

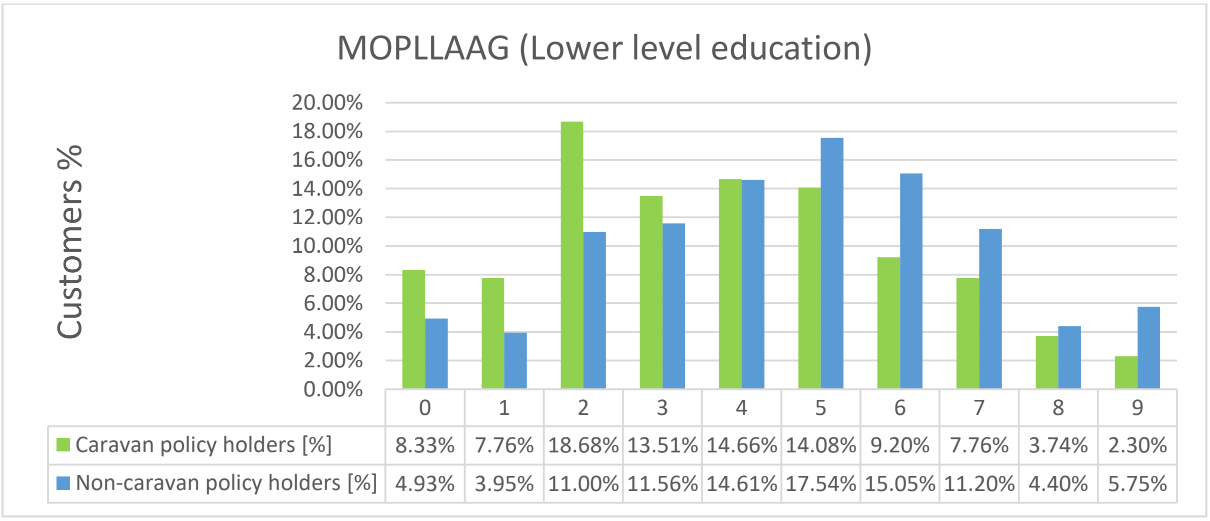


| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 🟩 Caravan policy holders [%] | 34.77% | 13.79% | 31.03% | 14.08% | 3.74% | 1.15% | 1.44% | 0.00% | 0.00% | 0.00% |
| 🟦 Non-caravan policy holders [%] | 24.28% | 13.30% | 27.71% | 18.58% | 10.49% | 3.11% | 1.53% | 0.46% | 0.24% | 0.31% |

*Fig.5: Effect of No Car on policy purchase*

[Back]

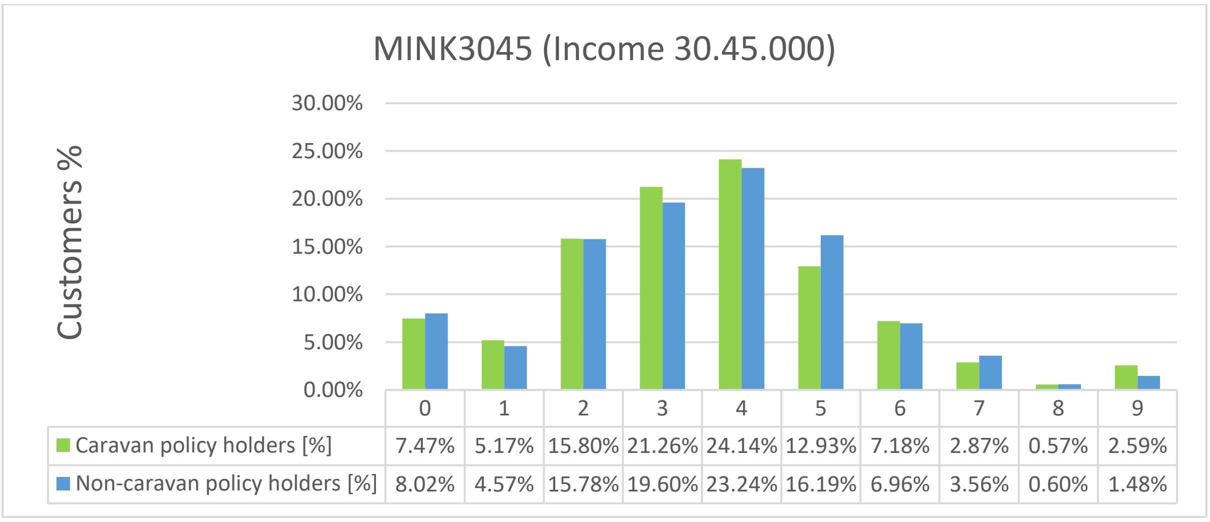
iii. Lower Level Education



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 🟩 Caravan policy holders [%] | 8.33% | 7.76% | 18.68% | 13.51% | 14.66% | 14.08% | 9.20% | 7.76% | 3.74% | 2.30% |
| 🟦 Non-caravan policy holders [%] | 4.93% | 3.95% | 11.00% | 11.56% | 14.61% | 17.54% | 15.05% | 11.20% | 4.40% | 5.75% |

*Fig.6: Effect of Lower Level Education on policy purchase.*

iv. Income between 30,000 and 45,000

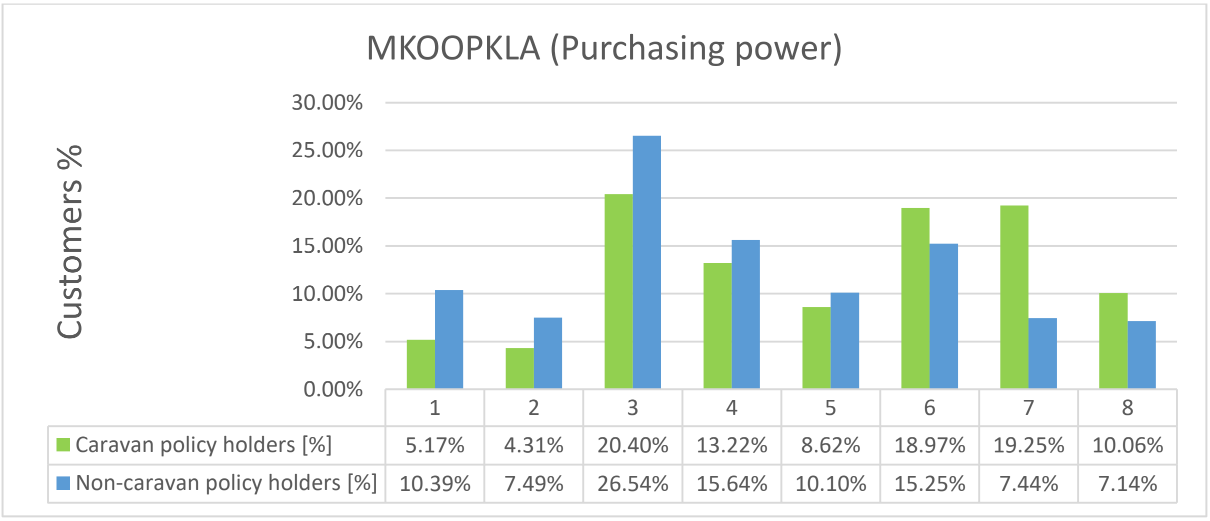


| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 🟩 Caravan policy holders [%] | 7.47% | 5.17% | 15.80% | 21.26% | 24.14% | 12.93% | 7.18% | 2.87% | 0.57% | 2.59% |
| 🟦 Non-caravan policy holders [%] | 8.02% | 4.57% | 15.78% | 19.60% | 23.24% | 16.19% | 6.96% | 3.56% | 0.60% | 1.48% |

*Fig.7: Effect of Income between 30 and 45000 on policy purchase.*

[Back]

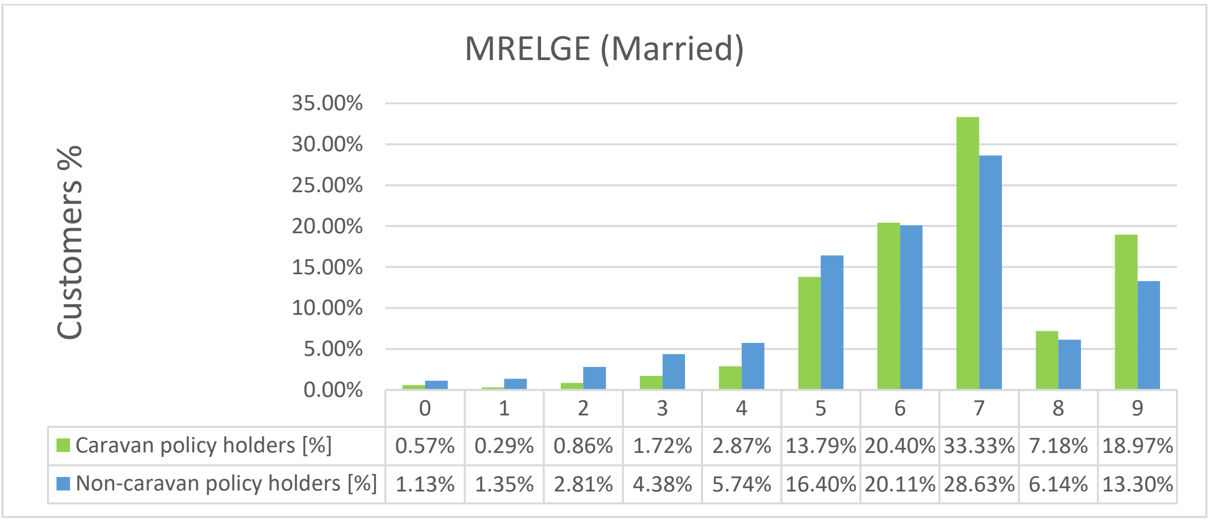
v. Purchasing Power Class



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 🟩 Caravan policy holders [%] | 5.17% | 4.31% | 20.40% | 13.22% | 8.62% | 18.97% | 19.25% | 10.06% |
| 🟦 Non-caravan policy holders [%] | 10.39% | 7.49% | 26.54% | 15.64% | 10.10% | 15.25% | 7.44% | 7.14% |

*Fig.8: Effect of Purchasing Power Class on policy purchase.*

vi. Married

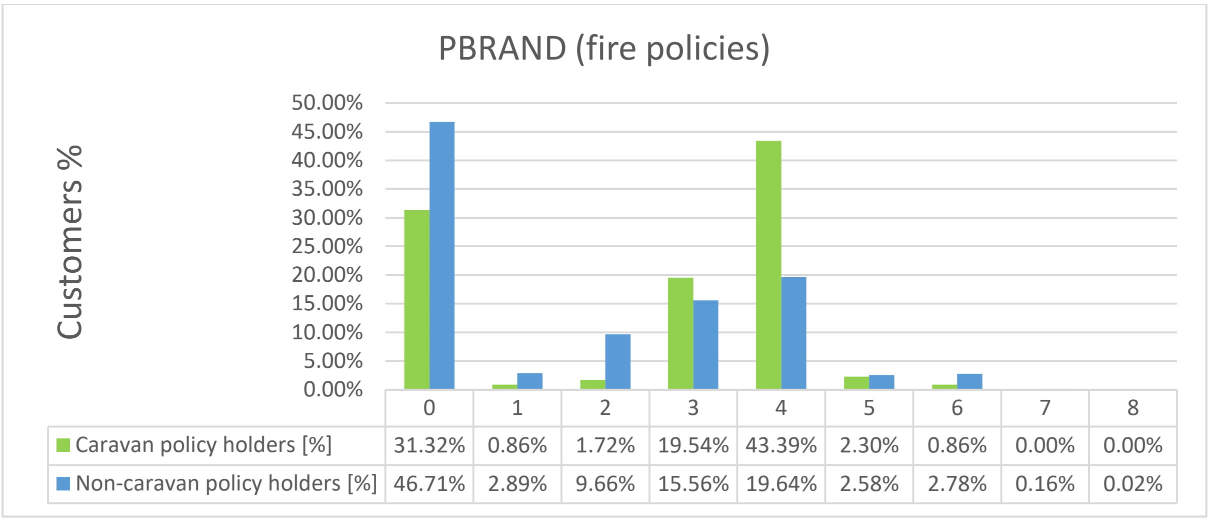


| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 🟩 Caravan policy holders [%] | 0.57% | 0.29% | 0.86% | 1.72% | 2.87% | 13.79% | 20.40% | 33.33% | 7.18% | 18.97% |
| 🟦 Non-caravan policy holders [%] | 1.13% | 1.35% | 2.81% | 4.38% | 5.74% | 16.40% | 20.11% | 28.63% | 6.14% | 13.30% |

*Fig.9: Effect of Being Married on policy purchase.*

[Back]

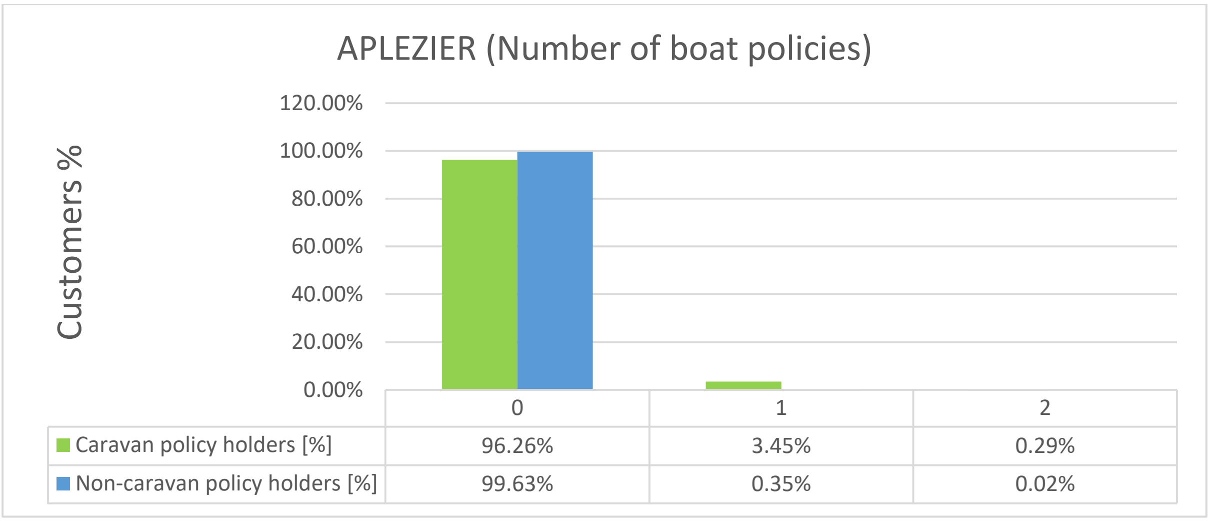
vii. Contribution of Fire Policies



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 🟩 Caravan policy holders [%] | 31.32% | 0.86% | 1.72% | 19.54% | 43.39% | 2.30% | 0.86% | 0.00% | 0.00% |
| 🟦 Non-caravan policy holders [%] | 46.71% | 2.89% | 9.66% | 15.56% | 19.64% | 2.58% | 2.78% | 0.16% | 0.02% |

*Fig.10: Effect of Fire Policies on policy purchase.*

viii. Number of Boat Policies

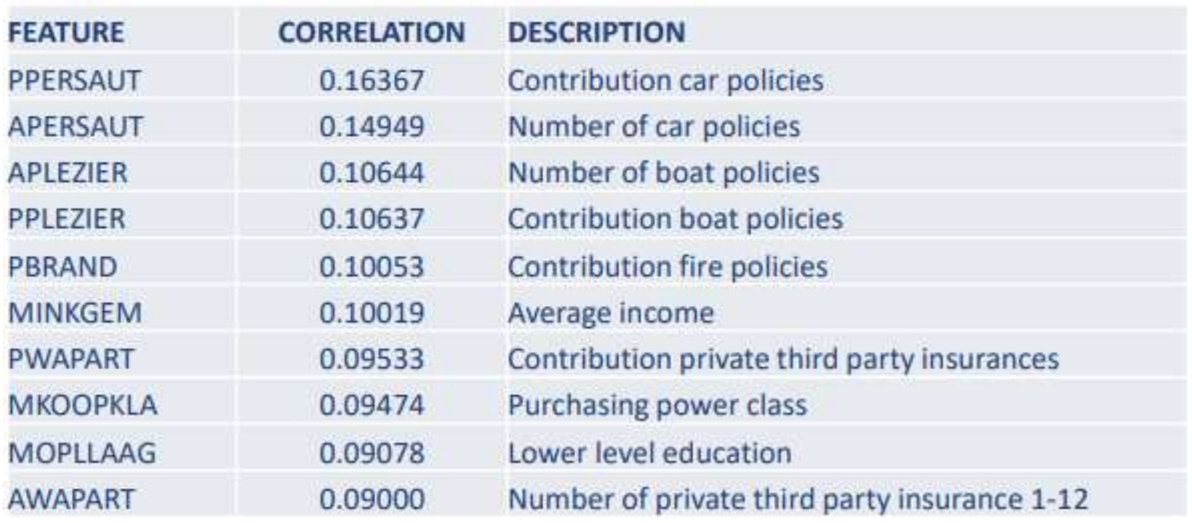


| | 0 | 1 | 2 |
|---|---|---|---|
| 🟩 Caravan policy holders [%] | 96.26% | 3.45% | 0.29% |
| 🟦 Non-caravan policy holders [%] | 99.63% | 0.35% | 0.02% |

*Fig.11: Effect of Boat Policies on policy purchase.*

[Back]

## Appendix C: Feature Shortlisting



| FEATURE | CORRELATION | DESCRIPTION |
|---|---|---|
| PPERSAUT | 0.16367 | Contribution car policies |
| APERSAUT | 0.14949 | Number of car policies |
| APLEZIER | 0.10644 | Number of boat policies |
| PPLEZIER | 0.10637 | Contribution boat policies |
| PBRAND | 0.10053 | Contribution fire policies |
| MINKGEM | 0.10019 | Average income |
| PWAPART | 0.09533 | Contribution private third party insurances |
| MKOOPKLA | 0.09474 | Purchasing power class |
| MOPLLAAG | 0.09078 | Lower level education |
| AWAPART | 0.09000 | Number of private third party insurance 1-12 |

*Fig. 12: Figure showing features with high-correlation with target fiel*

[Back]

## Appendix D: Information Gain and Chi-Squared Metrics

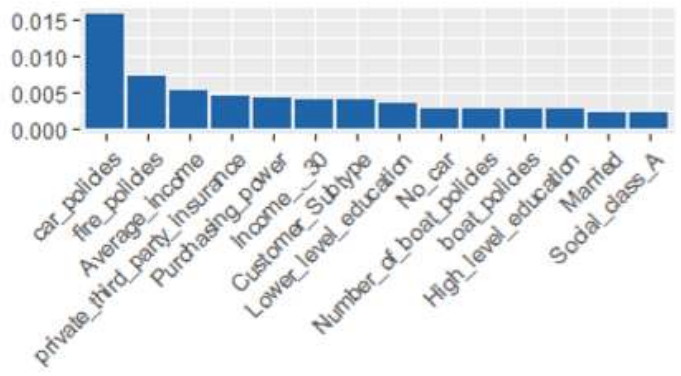


*Fig. 13: Feature classification according to Information Gain metrics*

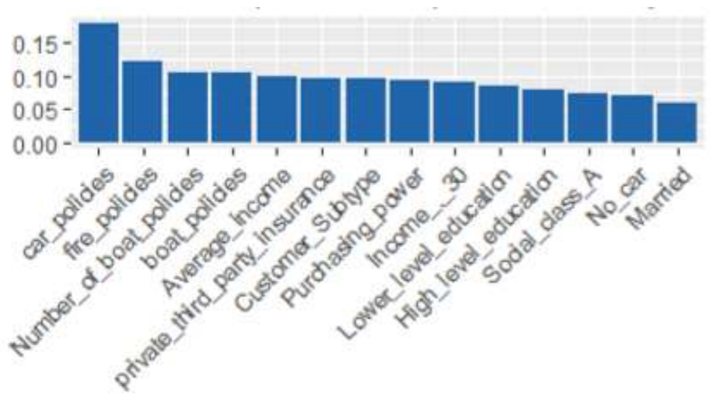


*Fig. 14: Feature classification according to Chi-Squared metrics*

[Back]

## Appendix E: LR Results with all features

Results of Logistic Regression using all features

| | Normal | Under Sampled | Over Sampled |
|---|---|---|---|
| Accuracy | 94.00% | 84.35% | 89.93% |
| Precision | 37.50% | 15.60% | 20.64% |
| Sensitivity | 1.26% | 36.97% | 24.37% |
| Specificity | 99.87% | 87.35% | 94.07% |
| 1- Specificity | 0.13% | 12.65% | 5.93% |

*Table 8: Logistic Regression results when using all features*

ROC Curve

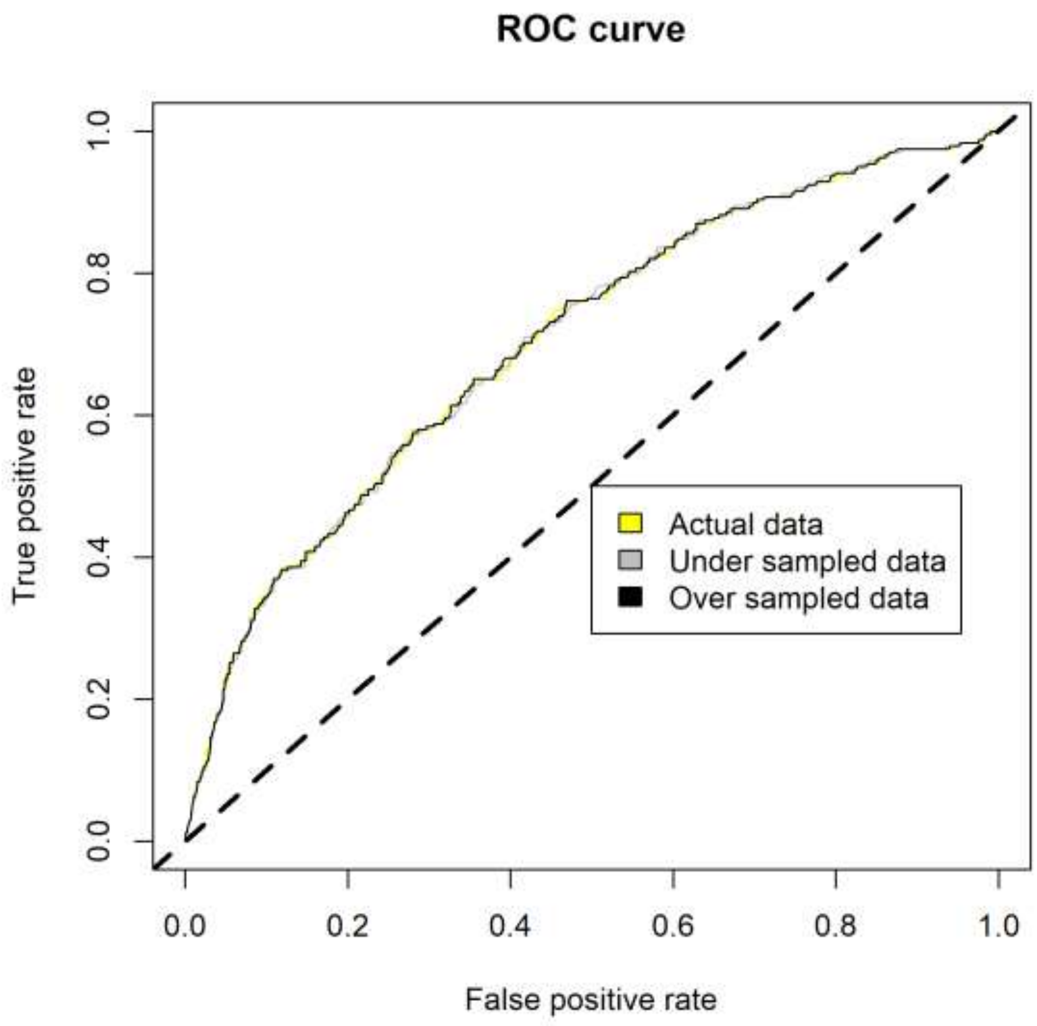


*Fig.15: ROC Curve obtained by using all features.*

[Back]

## Appendix F: LR Results using chosen features

ROC Curve

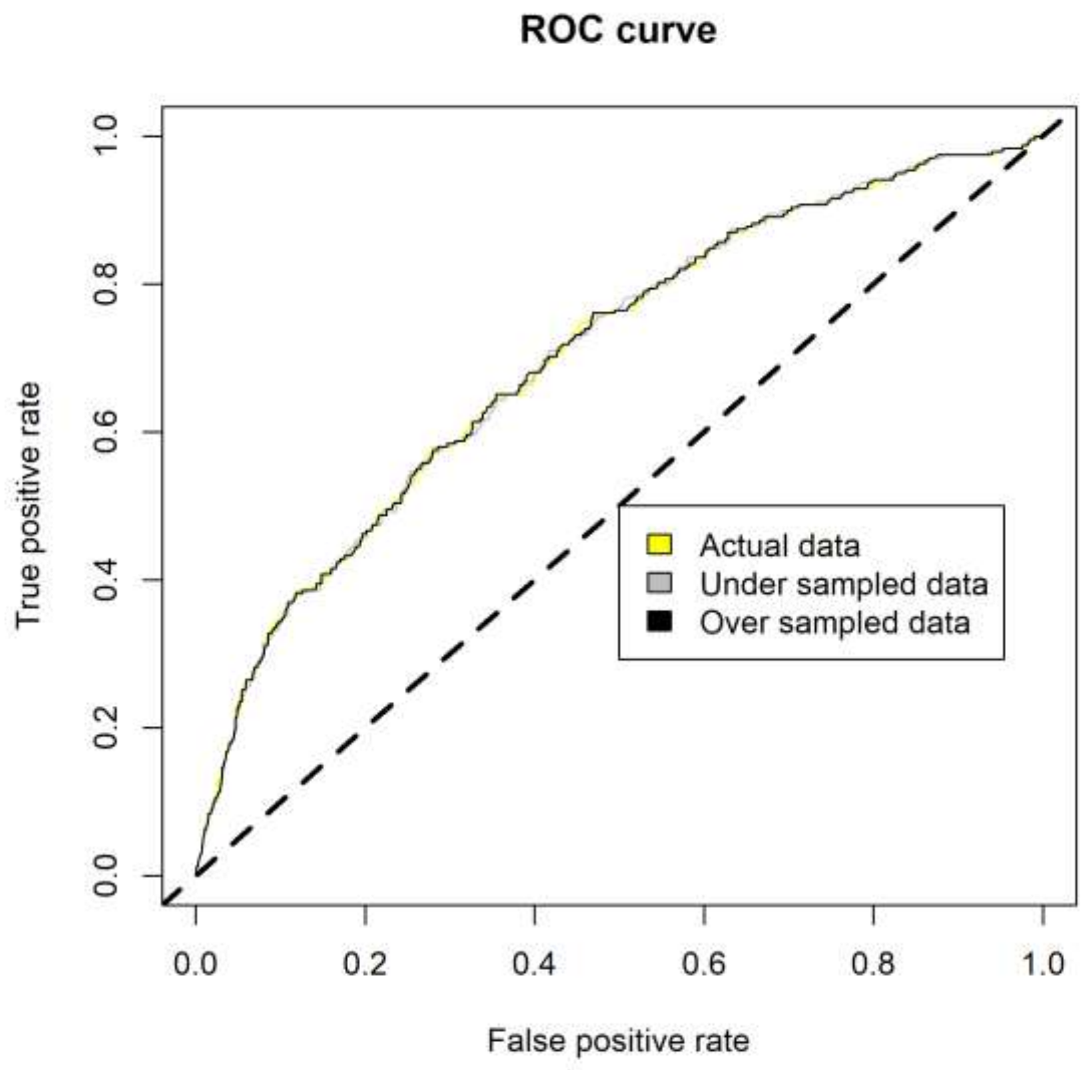
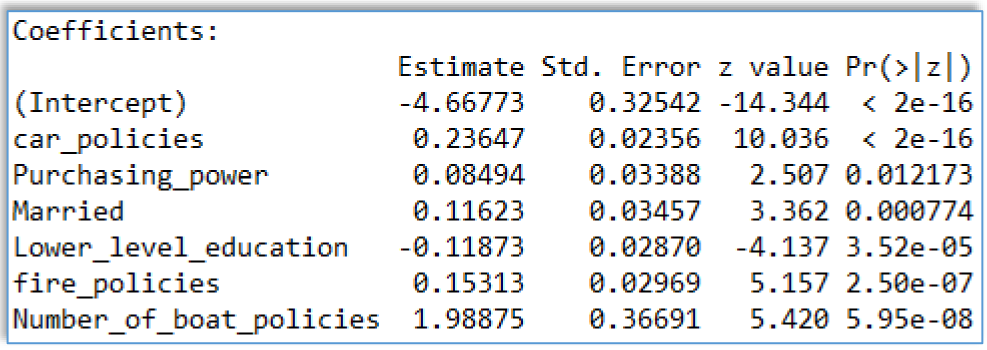


*Fig.16: ROC Curve obtained by using chosen features.*

[Back]

Coefficient Diagnostics of the three Logistic Regression Models

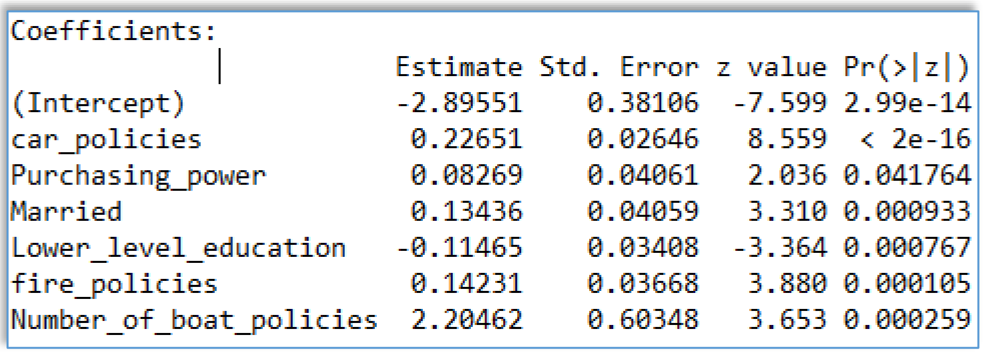
i. Normal



```
Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)             -4.66773    0.32542 -14.344  < 2e-16
car_policies             0.23647    0.02356  10.036  < 2e-16
Purchasing_power         0.08494    0.03388   2.507 0.012173
Married                  0.11623    0.03457   3.362 0.000774
Lower_level_education   -0.11873    0.02870  -4.137 3.52e-05
fire_policies            0.15313    0.02969   5.157 2.50e-07
Number_of_boat_policies  1.98875    0.36691   5.420 5.95e-08
```

*Table 9: Coefficient of Unbalanced LR Model*

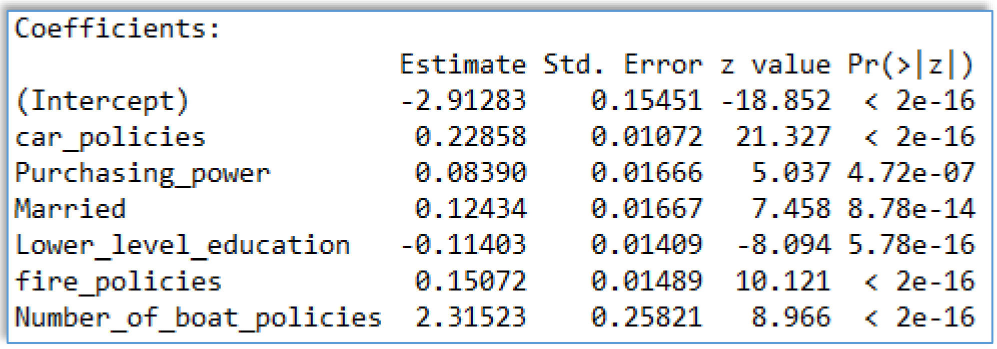
ii. Under sampled



```
Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)             -2.89551    0.38106  -7.599 2.99e-14
car_policies             0.22651    0.02646   8.559  < 2e-16
Purchasing_power         0.08269    0.04061   2.036 0.041764
Married                  0.13436    0.04059   3.310 0.000933
Lower_level_education   -0.11465    0.03408  -3.364 0.000767
fire_policies            0.14231    0.03668   3.880 0.000105
Number_of_boat_policies  2.20462    0.60348   3.653 0.000259
```

*Table 10: Coefficient of Balanced (Under Sampled) LR Model*

iii. Over sampled



```
Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)             -2.91283    0.15451 -18.852  < 2e-16
car_policies             0.22858    0.01072  21.327  < 2e-16
Purchasing_power         0.08390    0.01666   5.037 4.72e-07
Married                  0.12434    0.01667   7.458 8.78e-14
Lower_level_education   -0.11403    0.01409  -8.094 5.78e-16
fire_policies            0.15072    0.01489  10.121  < 2e-16
Number_of_boat_policies  2.31523    0.25821   8.966  < 2e-16
```

*Table 11: Coefficient of Balanced (Over Sampled) LR Model*

[Back]

## Appendix G: Decision Tree with several features

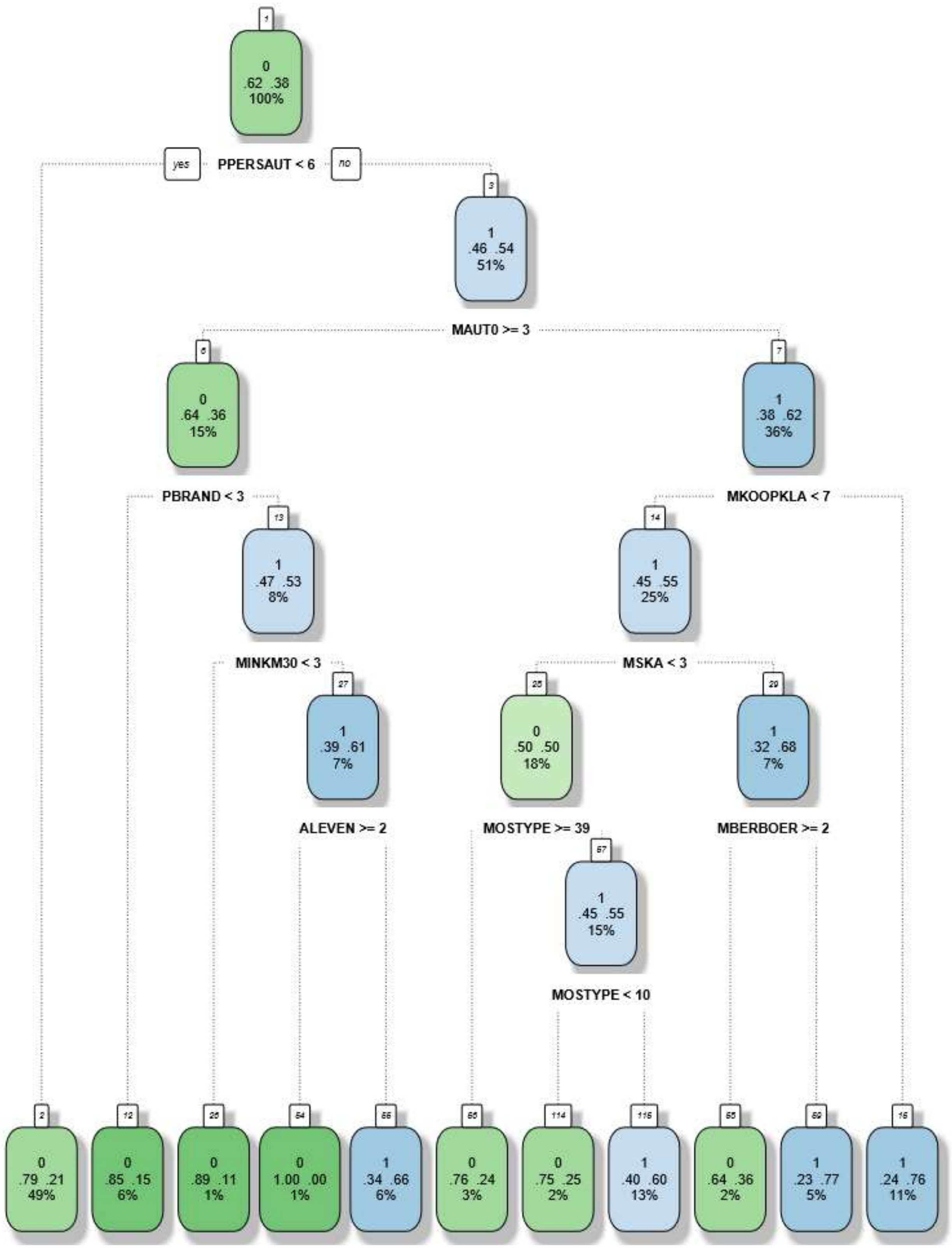


*Fig.17: Decision Tree obtained by using greater number of features.*

[Back]

## Appendix H: Random Forest

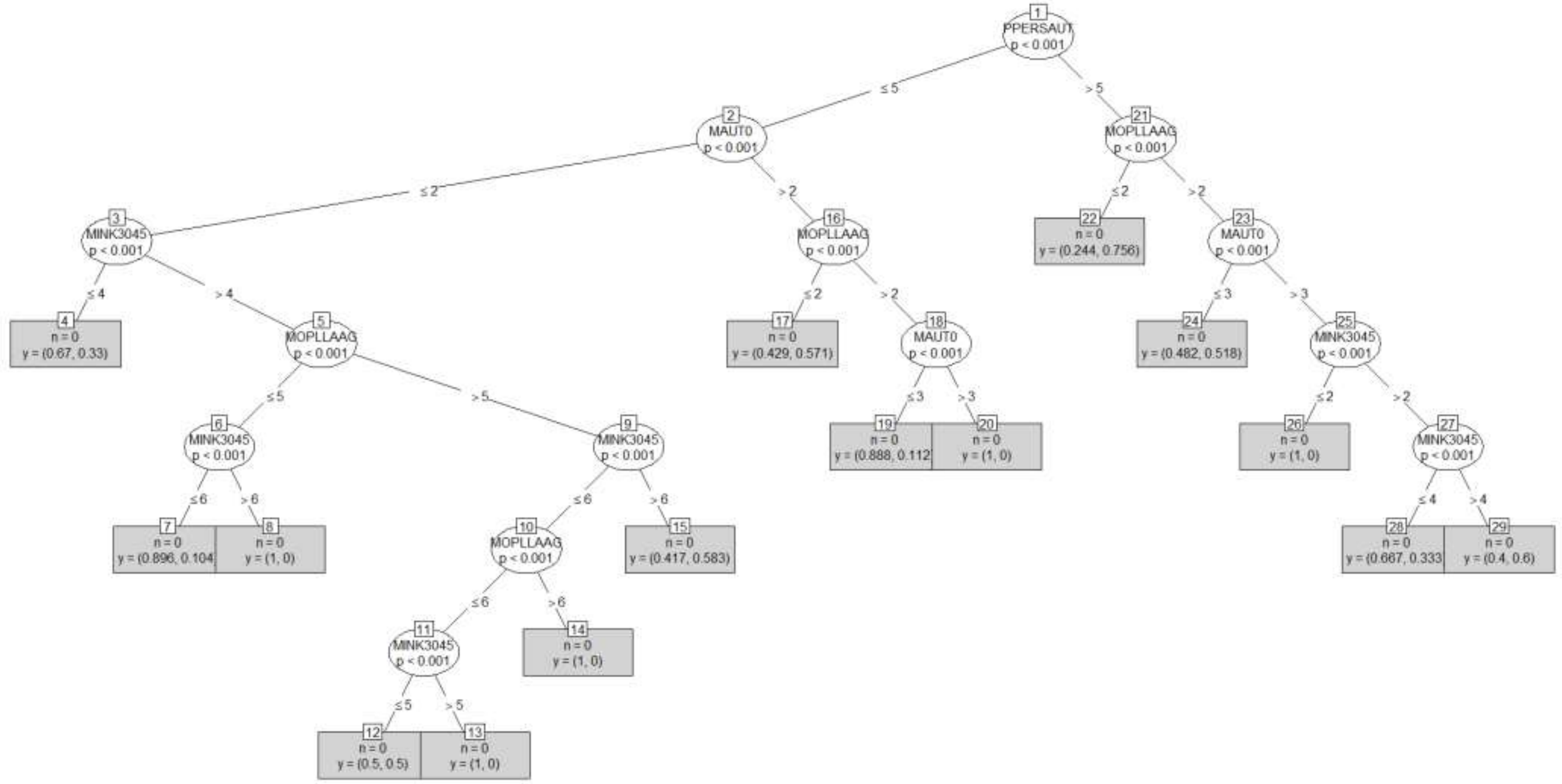


*Fig.18: Decision Tree obtained by using greater number of features.*

[Back]

## Appendix I: Codes

### Reading Data from Database

```
library(RPostgreSQL)

library(knitr)

library(rmarkdown)

library(ggplot2)

library(ggplot2)

library(plyr)

library(dummies)

library(ROCR)

library(crossval)

# ######################################################################

#

# ############################ functions ###########################

#

# ######################################################################


####### Rename_DF_caravan_toUpper #######


Rename_DF_caravan_toUpper <- function(df) {

  df <- rename(df,c(mostype="MOSTYPE"))

  df <- rename(df,c(maanthui="MAANTHUI"))

  df <- rename(df,c(mgemomv="MGEMOMV"))

  df <- rename(df,c(mgemleef="MGEMLEEF"))

  df <- rename(df,c(moshoofd="MOSHOOFD"))

  df <- rename(df,c(mgodrk="MGODRK"))

  df <- rename(df,c(mgodpr="MGODPR"))

  df <- rename(df,c(mgodov="MGODOV"))

  df <- rename(df,c(mgodge="MGODGE"))

  df <- rename(df,c(mrelge="MRELGE"))

  df <- rename(df,c(mrelsa="MRELSA"))
df <- rename(df,c(mrelov="MRELOV"))
df <- rename(df,c(mfalleen="MFALLEEN"))
df <- rename(df,c(mfgekind="MFGEKIND"))
df <- rename(df,c(mfwekind="MFWEKIND"))
df <- rename(df,c(moplhoog="MOPLHOOG"))
df <- rename(df,c(moplmidd="MOPLMIDD"))
df <- rename(df,c(mopllaag="MOPLLAAG"))
df <- rename(df,c(mberhoog="MBERHOOG"))
df <- rename(df,c(mberzelf="MBERZELF"))
df <- rename(df,c(mberboer="MBERBOER"))
df <- rename(df,c(mbermidd="MBERMIDD"))
df <- rename(df,c(mberarbg="MBERARBG"))
df <- rename(df,c(mberarbo="MBERARBO"))
df <- rename(df,c(mska="MSKA"))
df <- rename(df,c(mskb1="MSKB1"))
df <- rename(df,c(mskb2="MSKB2"))
df <- rename(df,c(mskc="MSKC"))
df <- rename(df,c(mskd="MSKD"))
df <- rename(df,c(mhhuur="MHHUUR"))
df <- rename(df,c(mhkoop="MHKOOP"))
df <- rename(df,c(maut1="MAUT1"))
df <- rename(df,c(maut2="MAUT2"))
df <- rename(df,c(maut0="MAUT0"))
df <- rename(df,c(mzfonds="MZFONDS"))
df <- rename(df,c(mzpart="MZPART"))
df <- rename(df,c(minkm30="MINKM30"))
df <- rename(df,c(mink3045="MINK3045"))
df <- rename(df,c(mink4575="MINK4575"))
df <- rename(df,c(mink7512="MINK7512"))
df <- rename(df,c(mink123m="MINK123M"))
df <- rename(df,c(minkgem="MINKGEM"))
df <- rename(df,c(mkoopkla="MKOOPKLA"))
df <- rename(df,c(pwapart="PWAPART"))
df <- rename(df,c(pwabedr="PWABEDR"))
df <- rename(df,c(pwaland="PWALAND"))
df <- rename(df,c(ppersaut="PPERSAUT"))
df <- rename(df,c(pbesaut="PBESAUT"))
df <- rename(df,c(pmotsco="PMOTSCO"))
df <- rename(df,c(pvraaut="PVRAAUT"))
df <- rename(df,c(paanhang="PAANHANG"))
df <- rename(df,c(ptractor="PTRACTOR"))
df <- rename(df,c(pwerkt="PWERKT"))
df <- rename(df,c(pbrom="PBROM"))
df <- rename(df,c(pleven="PLEVEN"))
df <- rename(df,c(ppersong="PPERSONG"))
df <- rename(df,c(pgezong="PGEZONG"))
df <- rename(df,c(pwaoreg="PWAOREG"))
df <- rename(df,c(pbrand="PBRAND"))
df <- rename(df,c(pzeilpl="PZEILPL"))
df <- rename(df,c(pplezier="PPLEZIER"))
df <- rename(df,c(pfiets="PFIETS"))
df <- rename(df,c(pinboed="PINBOED"))
df <- rename(df,c(pbystand="PBYSTAND"))
df <- rename(df,c(awapart="AWAPART"))
df <- rename(df,c(awabedr="AWABEDR"))
df <- rename(df,c(awaland="AWALAND"))
df <- rename(df,c(apersaut="APERSAUT"))
df <- rename(df,c(abesaut="ABESAUT"))
df <- rename(df,c(amotsco="AMOTSCO"))
df <- rename(df,c(avraaut="AVRAAUT"))
df <- rename(df,c(aaanhang="AAANHANG"))
df <- rename(df,c(atractor="ATRACTOR"))
```

```
  df <- rename(df,c(awerkt="AWERKT"))
  df <- rename(df,c(abrom="ABROM"))
  df <- rename(df,c(aleven="ALEVEN"))
  df <- rename(df,c(apersong="APERSONG"))
  df <- rename(df,c(agezong="AGEZONG"))
  df <- rename(df,c(awaoreg="AWAOREG"))
  df <- rename(df,c(abrand="ABRAND"))
  df <- rename(df,c(azeilpl="AZEILPL"))
  df <- rename(df,c(aplezier="APLEZIER"))
  df <- rename(df,c(afiets="AFIETS"))
  df <- rename(df,c(ainboed="AINBOED"))
  df <- rename(df,c(abystand="ABYSTAND"))
  df <- rename(df,c(caravan="CARAVAN"))
}

####### Rename_DF_caravan #######

Rename_DF_caravan <- function(df) {
  df <- rename(df,c(mostype="Customer_Subtype"))
  df <- rename(df,c(maanthui="Number_of_houses"))
  df <- rename(df,c(mgemomv="Avg_size_household"))
  df <- rename(df,c(mgemleef="Avg_age"))
  df <- rename(df,c(moshoofd="Customer_main_type"))
  df <- rename(df,c(mgodrk="Roman_catholic"))
  df <- rename(df,c(mgodpr="Protestant"))
  df <- rename(df,c(mgodov="Other_religion"))
  df <- rename(df,c(mgodge="No_religion"))
  df <- rename(df,c(mrelge="Married"))
  df <- rename(df,c(mrelsa="Living_together"))
  df <- rename(df,c(mrelov="Other_relation"))
  df <- rename(df,c(mfalleen="Singles"))
  df <- rename(df,c(mfgekind="Household_without_children"))

  df <- rename(df,c(mfwekind="Household_with_children"))

  df <- rename(df,c(moplhoog="High_level_education"))

  df <- rename(df,c(moplmidd="Medium_level_education"))

  df <- rename(df,c(mopllaag="Lower_level_education"))

  df <- rename(df,c(mberhoog="High_status"))

  df <- rename(df,c(mberzelf="Entrepreneur"))

  df <- rename(df,c(mberboer="Farmer"))

  df <- rename(df,c(mbermidd="Middle_management"))

  df <- rename(df,c(mberarbg="Skilled_labourers"))

  df <- rename(df,c(mberarbo="Unskilled_labourers"))

  df <- rename(df,c(mska="Social_class_A"))

  df <- rename(df,c(mskb1="Social_class_B1"))

  df <- rename(df,c(mskb2="Social_class_B2"))

  df <- rename(df,c(mskc="Social_class_C"))

  df <- rename(df,c(mskd="Social_class_D"))

  df <- rename(df,c(mhhuur="Rented_house"))

  df <- rename(df,c(mhkoop="Home_owners"))

  df <- rename(df,c(maut1="X1_car"))

  df <- rename(df,c(maut2="X2_cars"))

  df <- rename(df,c(maut0="No_car"))

  df <- rename(df,c(mzfonds="National_Health_Service"))

  df <- rename(df,c(mzpart="Private_health_insurance"))

  df <- rename(df,c(minkm30="Income_._30"))

  df <- rename(df,c(mink3045="Income_30.45.000"))

  df <- rename(df,c(mink4575="Income_45.75.000"))

  df <- rename(df,c(mink7512="Income_75.122.000"))

  df <- rename(df,c(mink123m="Income_.123.000"))

  df <- rename(df,c(minkgem="Average_income"))

  df <- rename(df,c(mkoopkla="Purchasing_power"))

  df <- rename(df,c(pwapart="private_third_party_insurance"))
  df <- rename(df,c(pwabedr="third_party_insurance_firms"))
  df <- rename(df,c(pwaland="third_party_insurane_agriculture"))
  df <- rename(df,c(ppersaut="car_policies"))
  df <- rename(df,c(pbesaut="delivery_van_policies"))
  df <- rename(df,c(pmotsco="Contribution_motorcycle.scooter"))
  df <- rename(df,c(pvraaut="lorry_policies"))
  df <- rename(df,c(paanhang="trailer_policies"))
  df <- rename(df,c(ptractor="tractor_policies"))
  df <- rename(df,c(pwerkt="agricultural_machines_policies"))
  df <- rename(df,c(pbrom="moped_policies"))
  df <- rename(df,c(pleven="life_insurances"))
  df <- rename(df,c(ppersong="private_accident_insurance_policies"))
  df <- rename(df,c(pgezong="family_accidents_insurance_policies"))
  df <- rename(df,c(pwaoreg="disability_insurance_policies"))
  df <- rename(df,c(pbrand="fire_policies"))
  df <- rename(df,c(pzeilpl="surfboard_policies"))
  df <- rename(df,c(pplezier="boat_policies"))
  df <- rename(df,c(pfiets="bicycle_policies"))
  df <- rename(df,c(pinboed="property_insurance_policies"))
  df <- rename(df,c(pbystand="social_security_insurance_policies"))
  df <- rename(df,c(awapart="Number_of_private_third_party_insurance"))
  df <- rename(df,c(awabedr="Number_of_third_party_insurance_firms"))
  df <- rename(df,c(awaland="Number_of_third_party_insurane_agriculture"))
  df <- rename(df,c(apersaut="Number_of_car_policies"))
  df <- rename(df,c(abesaut="Number_of_delivery_van_policies"))
  df <- rename(df,c(amotsco="Number_of_motorcycle.scooter_policies"))
  df <- rename(df,c(avraaut="Number_of_lorry_policies"))
  df <- rename(df,c(aaanhang="Number_of_trailer_policies"))
  df <- rename(df,c(atractor="Number_of_tractor_policies"))
  df <- rename(df,c(awerkt="Number_of_agricultural_machines_policies"))
  df <- rename(df,c(abrom="Number_of_moped_policies"))
```

```
  df <- rename(df,c(aleven="Number_of_life_insurances"))
  df <- rename(df,c(apersong="Number_of_private_accident_insurance_policies"))
  df <- rename(df,c(agezong="Number_of_family_accidents_insurance_policies"))
  df <- rename(df,c(awaoreg="Number_of_disability_insurance_policies"))
  df <- rename(df,c(abrand="Number_of_fire_policies"))
  df <- rename(df,c(azeilpl="Number_of_surfboard_policies"))
  df <- rename(df,c(aplezier="Number_of_boat_policies"))
  df <- rename(df,c(afiets="Number_of_bicycle_policies"))
  df <- rename(df,c(ainboed="Number_of_property_insurance_policies"))
  df <- rename(df,c(abystand="Number_of_social_security_insurance_policies"))
  df <- rename(df,c(caravan="Number_of_mobile_home_policies"))
}

# ###################################################################
#
# ########################### CODE ##########################
#
# ###################################################################

con <- dbConnect(PostgreSQL(), user= "postgres", password="postgres", dbname="caravan")



# OJO: CAMBIAR LA QUERY
# df_DB <- fetch( dbSendQuery(con,
#                              "SELECT * FROM tt_features_1_selection"), n=Inf)
# df_DB <- rename(df_DB,c(caravan="Number_of_mobile_home_policies"))
# df_DB <- Rename_DF_caravan(df_DB)
# df_DB <- Rename_DF_caravan_toUppeDr(df_DB)
# lapply(dbListConnections(drv = dbDriver("PostgreSQL")),
#        function(x) {dbDisconnect(con = x)})
cat("\014")
```

## Logistic Regression [Back]

```
#                               -- Loading libraries --

#


# clear variables

closeAllConnections()

rm(list=ls())

cat("\014")


library(RPostgreSQL)

library(knitr)

library(rmarkdown)

library(ggplot2)

library(ggplot2)

library(plyr)

library(dummies)

library(ROCR)

library(crossval)

library(xlsx)


# load data

read_data_DB = 0

if (read_data_DB == 1) { # read data from DB

  # First execute file ReadDataFromDB.R

  source("C:\\Users\\Laura\\Desktop\\Caravan Insurance - R\\CODE\\ReadDataFromDB.R")

  df_DBtrain <- fetch( dbSendQuery(con,

                                  "SELECT * FROM tt_caravan_v1"), n=Inf)


  df_DBtrain <- Rename_DF_caravan_toUpper ( df_DBtrain)

  df_DBtrain <- rename(df_DBtrain,c(CARAVAN="Purchase"))
  caravan_df_maindataset <- df_DBtrain

  caravan_df_train <- df_DBtrain

  df_DBtest <- fetch( dbSendQuery(con,

                        "SELECT * FROM tt_test_caravan"), n=Inf)

  df_DBtest <- Rename_DF_caravan_toUpper ( df_DBtest)

  df_DBtest <- rename(df_DBtest,c(CARAVAN="Purchase"))

  caravan_df_test <- df_DBtest


} else { # load data from file

  caravan_df_maindataset <- read.csv('C:\\Users\\Laura\\Desktop\\Caravan Insurance -
R\\DATA\\TrainingData_V2.csv')

  caravan_df_train <- read.csv('C:\\Users\\Laura\\Desktop\\Caravan Insurance -
R\\DATA\\TrainingData_V2.csv')

  caravan_df_test <- read.csv('C:\\Users\\Laura\\Desktop\\Caravan Insurance -
R\\DATA\\TestData_V2 (comma).csv')

  caravan_df_maindataset <-
rename(caravan_df_maindataset,c(Number_of_mobile_home_policies="Purchase"))

  caravan_df_train <- rename(caravan_df_train,c(Number_of_mobile_home_policies="Purchase"))

  caravan_df_test <- rename(caravan_df_test,c(Number_of_mobile_home_policies="Purchase"))

}


select_features = 1

if (select_features == 1) { # Select features for logistic regression


  keeps_Lau <- c("fire_policies", "private_third_party_insurance", "car_policies",
"Customer_Subtype",

            "Average_income", "Purchasing_power", "High_level_education",

            "Income_._30", "Social_class_A", "Number_of_boat_policies", "Lower_level_education",

            "boat_policies", "No_car", "Married", "social_security_insurance_policies", "Purchase")


  keeps <-
c("car_policies","Purchasing_power","Married","Lower_level_education","fire_policies","Number_o
f_boat_policies", "Purchase")
```

```
  caravan_df_maindataset <- caravan_df_maindataset[keeps]
  caravan_df_train <- caravan_df_train[keeps]
  caravan_df_test <- caravan_df_test[keeps]
}
#Check Shape of Data and List Out All Columns
str(caravan_df_maindataset)
#Sample of the Data
head(caravan_df_maindataset)
# Ensure there are no missing values.
paste0("Missing values: ", sum(is.na(caravan_df_maindataset)))
# Finding which features are numeric.
numeric_caravan <- which(sapply(caravan_df_maindataset,is.numeric))
str(caravan_df_maindataset[,numeric_caravan])
# All features are numeric
caravan_df_maindataset$Purchase <- factor(caravan_df_maindataset$Purchase, labels=c(0,1))
#Let's Check How Unbalanced Our Data Is By Counting The Number of 1s and 0s In The Feature Purchase
Non_Caravan_Insurance  <- sum(caravan_df_maindataset$Purchase == 0)
Caravan_Insurance_Holder <- sum(caravan_df_maindataset$Purchase == 1)
dat <- data.frame(
  Caravan_Insurance_Holders = factor(c("Non_Caravan_Insurance","Caravan_Insurance_Holder"), levels=c("Non_Caravan_Insurance","Caravan_Insurance_Holder")),
  Count = c( Non_Caravan_Insurance , Caravan_Insurance_Holder)
)
ggplot(data=dat, aes(x=Caravan_Insurance_Holders, y=Count, fill=Caravan_Insurance_Holders)) +
  geom_bar(colour="black", stat="identity")
# [7]
Frequency.Purchase <- data.frame(Purchase = levels(caravan_df_maindataset$Purchase), Count = as.numeric(table(caravan_df_maindataset$Purchase)))
Frequency.Purchase
# characteristics of the observations that actually bought the 'mobile home insurance'.
# In [313]:

TrainDataset <-caravan_df_maindataset


car_policies <- sum(TrainDataset$Purchase == 1 & TrainDataset$car_policies  != 0)

Purchasing_power <- sum(TrainDataset$Purchase == 1 & TrainDataset$Purchasing_power  != 0)

Married <- sum(TrainDataset$Purchase == 1 & TrainDataset$Married  != 0)

fire_policies <- sum(TrainDataset$Purchase == 1 & TrainDataset$fire_policies  != 0)

Lower_level_education <- sum(TrainDataset$Purchase == 1 & TrainDataset$Lower_level_education
!= 0)


dat <- data.frame(

  Selected_Features = factor(c("car_policies" , "Purchasing_power " , "Married" , "fire_policies"  ,
"Lower_level_education" ), levels=c("car_policies" , "Purchasing_power " , "Married" , "fire_policies"
, "Lower_level_education")),

  Count = c( car_policies  ,  Purchasing_power  , Married , fire_policies  , Lower_level_education ))


ggplot(data=dat, aes(x=Selected_Features, y=Count, fill=Selected_Features)) +

  geom_bar(colour="black", stat="identity")


#                                   -- LOGISTIC REGRESSION --


# 3. Classification using Logistic Regression - Unbalanced Data, Undersampled Data, Oversampled
Data - Including Business Cost For Each Model

# Logistic Using Unbalanced Data

# In [209]:


str(caravan_df_test)

paste0("Missing values: ", sum(is.na(caravan_df_test)))


#LR Pre-Processing

caravan_df_trainLR <- caravan_df_train

caravan_df_testLR <- caravan_df_test
```

```
caravan_df_trainLR$Purchase <- factor(caravan_df_trainLR$Purchase, labels = c(0,1))


caravan_df_trainLR <- dummy.data.frame(caravan_df_trainLR, sep = ".", names =
c("Customer_main_type","Customer_Subtype"))

caravan_df_testLR <- dummy.data.frame(caravan_df_testLR, sep = ".", names =
c("Customer_main_type","Customer_Subtype"))


# caravan_df_trainLR <- dummy.data.frame(caravan_df_trainLR, sep = ".", names =
c("Purchasing_power"))

# caravan_df_testLR <- dummy.data.frame(caravan_df_testLR, sep = ".", names =
c("Purchasing_power"))


# FINAL SET !!!!!!!!!!!!!!!!!!!!!!!!!!!!!


rm(caravan_df_maindataset)

rm(caravan_df_test)

rm(caravan_df_train)


Select_subset =0


if (Select_subset == 1){


  caravan_df_trainLR <- subset(caravan_df_trainLR, select = c(car_policies
,Average_income,Purchasing_power ,private_third_party_insurance ,

                                      fire_policies ,Number_of_boat_policies
,social_security_insurance_policies ,Lower_level_education ,

                                      High_level_education  ,No_car ,Customer_Subtype.38
,Customer_Subtype.37 ,

                                      Customer_Subtype.3 ,Customer_Subtype.39
,Customer_Subtype.36 ,Customer_Subtype.12 ,Customer_Subtype.8 ,

                                      Customer_Subtype.20 ,Customer_Subtype.33 ,

                                      Purchase))

  caravan_df_testLR <- subset(caravan_df_testLR, select = c(car_policies ,Average_income,
Purchasing_power ,private_third_party_insurance ,
                                     fire_policies ,Number_of_boat_policies
,social_security_insurance_policies ,Lower_level_education ,
                                     High_level_education  ,No_car ,Customer_Subtype.38
,Customer_Subtype.37 ,
                                     Customer_Subtype.3 ,Customer_Subtype.39
,Customer_Subtype.36 ,Customer_Subtype.12 ,Customer_Subtype.8 ,
                                     Customer_Subtype.20 ,Customer_Subtype.33 ,
                                     Purchase))
}

# Lower level education greater and smaller than 5 ... ???

#Classify Using Logistic Regression
logisticTrainingFit <- glm(Purchase ~ ., family = "binomial", data = caravan_df_trainLR)



#In [217]: Predict Class And Display Confusion Matrix
#Predict Class And Display Confusion Matrix
LOutPredicted <- predict(logisticTrainingFit, caravan_df_testLR, type = "response")
LOutPredictedClass <- ifelse(LOutPredicted>0.5, 1, 0)
LOutActual <- caravan_df_testLR$Purchase
LConfusionOutPredicted <- table(LOutActual, LOutPredictedClass)
rownames(LConfusionOutPredicted) <- c("0","1")
colnames(LConfusionOutPredicted) <- c("0","1")
LConfusionOutPredicted
for (i in 1:4000){
  probabilityDF$LR[i] <- LOutPredicted[i]
}

# Plot ROC and AUC for LR
probs <- LOutPredicted
library(ROCR)
```

```
LRPred <- prediction(probs, caravan_df_testLR$Purchase)
LRPerf <- performance(LRPred, "tpr", "fpr")

plotROC = 0
if (plotROC == 1) { # plots ROC
  dev.new()
  plot(LRPerf, colorize=TRUE)
  abline(a=0, b=1, lty=2, lwd=3, col="black")
}

#AUC
auc <- performance(LRPred, "auc")

#Corresponding Performance Measures
LRPrediction <- factor(as.factor(LOutPredictedClass), c(0, 1),
                       labels = c("Not Purchased", "Purchased"))
LRActual <- factor(as.factor(caravan_df_testLR$Purchase),
                   c(0, 1), labels = c("Not Purchased", "Purchased"))
CMLR <- confusionMatrix(LRActual, LRPrediction, negative = "Not Purchased" )




# diagnosticErrors computes various diagnostic errors useful for evaluating
# the performance of a diagnostic test or a classifier:
# accuracy (acc), sensitivity (sens), specificity (spec),
# positive predictive value (ppv), negative predictive value (npv), and log-odds ratio (lor).

# percentage of correct classification (Accuracy)
mean(LRPrediction ==LRActual)
diagnosticErrors(CMLR)
```

```
# comparing models
normal.LRPerf <- LRPerf
normal.LRPred <- LRPred
normal.auc <- performance(LRPred, "auc")
normal.diagnosticErrors <- diagnosticErrors(CMLR)
normal.CMLR <- CMLR
normal.logisticTrainingFit <- summary(logisticTrainingFit)

#                                  -- LOGISTIC Undersampled --

# Logistic Regression Using Undersampled Data
# In [221]:
# Under Sampling Data
# Taking all the observations with dependent variable = 1

caravan_df_train_LR_Under <- caravan_df_trainLR[caravan_df_trainLR$Purchase==1,]
length(caravan_df_train_LR_Under$Purchase)
#Randomly select observations with dependent variable = 0
zeroObs <- caravan_df_trainLR[caravan_df_trainLR$Purchase==0,]
set.seed(123457)

# SAMPLING SELECTION !!!!!
samplin_level = 1
samplin_level = 2.5


rearrangedZeroObs <-  zeroObs[sample(nrow(zeroObs),
length(caravan_df_train_LR_Under$Purchase) * samplin_level ),]
```

```
#Appending rows of randomly selected 0s in our undersampled data frame

caravan_df_train_LR_Under <- rbind(caravan_df_train_LR_Under, rearrangedZeroObs)

length(caravan_df_train_LR_Under$Purchase)


#Let's verify that number of 1s and 0s in our undersampled data are equal

undersampled.Frequency.Purchase <- data.frame(Purchase =
levels(as.factor(caravan_df_train_LR_Under$Purchase)), Count =
as.numeric(table(caravan_df_train_LR_Under$Purchase)))

undersampled.Frequency.Purchase


#Classify Using Logistic Regression with Undersampling

logisticTrainingFit <- glm(Purchase ~ ., family = "binomial", data = caravan_df_train_LR_Under)




#Predict Class And Display Confusion Matrix

LOutPredicted <- predict(logisticTrainingFit, caravan_df_testLR, type = "response")

LOutPredictedClass <- ifelse(LOutPredicted>0.5, 1, 0)

LOutActual <- caravan_df_testLR$Purchase

LConfusionOutPredicted <- table(LOutActual, LOutPredictedClass)

rownames(LConfusionOutPredicted) <- c("0","1")

colnames(LConfusionOutPredicted) <- c("0","1")

LConfusionOutPredicted

for (i in 1:4000){

  probabilityDF$LRU[i] <- LOutPredicted[i]

}



# Plot ROC and AUC for LR

probs <- LOutPredicted

library(ROCR)

LRPred <- prediction(probs, caravan_df_testLR$Purchase)
```

```
LRPerf <- performance(LRPred, "tpr", "fpr")


if (plotROC == 1) { # plots ROC
  plot(LRPerf, colorize=TRUE)
  abline(a=0, b=1, lty=2, lwd=3, col="black")
}


#AUC
performance(LRPred, "auc")


#Corresponding Performance Measures
LRPrediction <- factor(as.factor(LOutPredictedClass), c(0, 1), labels = c("Not Purchased",
"Purchased"))
LRActual <- factor(as.factor(caravan_df_testLR$Purchase), c(0, 1), labels = c("Not Purchased",
"Purchased"))
library(crossval)
CMLR <- confusionMatrix(LRActual, LRPrediction, negative = "Not Purchased" )
diagnosticErrors(CMLR)


# comparing models
usample.LRPerf <- LRPerf
usample.LRPred <- LRPred
usample.auc <- performance(LRPred, "auc")
usample.diagnosticErrors <- diagnosticErrors(CMLR)
usample.CMLR <- CMLR
usample.logisticTrainingFit <- summary(logisticTrainingFit)


#                                   -- LOGISTIC OVERsampled --


# Logistic Regression Using Oversampling
# In [224]:
```

```
#First let's recall the ratio of 1s and 0s in our original training dataset

paste0("Ratio of 1s to 0s- 1:", 
as.numeric(table(caravan_df_trainLR$Purchase))[1]/as.numeric(table(caravan_df_trainLR$Purchase)
)[2])


#Now let's duplicate the observations with dependent variable = 1 to make the ratio approximately 
1:2

caravan_df_train_LR_OverDummy <- caravan_df_trainLR[caravan_df_trainLR$Purchase==1,]

caravan_df_train_LR_Over <- NULL

# SAMPLING SELECTION !!!!!

samplin_level <- 7

samplin_level <- 6


for (i in 1:samplin_level){

  caravan_df_train_LR_Over <- rbind(caravan_df_train_LR_Over, caravan_df_train_LR_OverDummy)

}


caravan_df_train_LR_Over <- rbind(caravan_df_train_LR_Over, 
caravan_df_trainLR[caravan_df_trainLR$Purchase==0,])

#Let's verify the number of 1s and 0s in our oversampled data

oversampled.Frequency.Purchase <- data.frame(Purchase = 
levels(as.factor(caravan_df_train_LR_Over$Purchase)), Count = 
as.numeric(table(caravan_df_train_LR_Over$Purchase)))

oversampled.Frequency.Purchase


#Let's verify the ratio of 1s to 0s after over sampling

paste0("Ratio of 1s to 0s After Oversampling- 1:", 
as.numeric(table(caravan_df_train_LR_Over$Purchase))[1]/as.numeric(table(caravan_df_train_LR_O
ver$Purchase))[2])


#Classify Using Logistic Regression with Oversampling

logisticTrainingFit <- glm(Purchase ~ ., family = "binomial", data = caravan_df_train_LR_Over)
```

```
#Predict Class And Display Confusion Matrix
LOutPredicted <- predict(logisticTrainingFit, caravan_df_testLR, type = "response")
LOutPredictedClass <- ifelse(LOutPredicted>0.5, 1, 0)
LOutActual <- caravan_df_testLR$Purchase
LConfusionOutPredicted <- table(LOutActual, LOutPredictedClass)
rownames(LConfusionOutPredicted) <- c("0","1")
colnames(LConfusionOutPredicted) <- c("0","1")
LConfusionOutPredicted
for (i in 1:4000){
  probabilityDF$LRO[i] <- LOutPredicted[i]
}
for (i in 1:4000){
  probabilityDF$Actual[i] <- LOutActual[i]
}



# Plot ROC and AUC for LR
probs <- LOutPredicted
library(ROCR)
LRPred <- prediction(probs, caravan_df_testLR$Purchase)
LRPerf <- performance(LRPred, "tpr", "fpr")

if (plotROC == 1) { # plots ROC
  plot(LRPerf, colorize=TRUE)
  abline(a=0, b=1, lty=2, lwd=3, col="black")
}

#AUC
auc <- performance(LRPred, "auc")

#Corresponding Performance Measures
```

```
LRPrediction <- factor(as.factor(LOutPredictedClass), c(0, 1), labels = c("Not Purchased",
"Purchased"))

LRActual <- factor(as.factor(caravan_df_testLR$Purchase), c(0, 1), labels = c("Not Purchased",
"Purchased"))

library(crossval)

CMLR <- confusionMatrix(LRActual, LRPrediction, negative = "Not Purchased" )

diagnosticErrors(CMLR)


# comparing models

osample.LRPerf <- LRPerf

osample.LRPred <- LRPred

osample.auc <- performance(LRPred, "auc")

osample.diagnosticErrors <- diagnosticErrors(CMLR)

osample.CMLR <- CMLR

osample.logisticTrainingFit <- summary(logisticTrainingFit)


# save to excel

wb = createWorkbook()


sheet = createSheet(wb, "Normal LR")


addDataFrame(normal.diagnosticErrors, sheet=sheet, startRow = 1, startColumn=2,
row.names=TRUE)

addDataFrame(normal.CMLR, sheet=sheet, startRow =10, startColumn=2, row.names=TRUE)


sheet = createSheet(wb, "Undersampled LR")


addDataFrame(usample.diagnosticErrors, sheet=sheet, startRow = 1, startColumn=2,
row.names=TRUE)

addDataFrame(usample.CMLR, sheet=sheet, startRow =10, startColumn=2, row.names=TRUE)


sheet = createSheet(wb, "Oversampled LR")
```

```
addDataFrame(osample.diagnosticErrors, sheet=sheet, startRow = 1, startColumn=2,
row.names=TRUE)

addDataFrame(osample.CMLR, sheet=sheet, startRow =10, startColumn=2, row.names=TRUE)


saveWorkbook(wb, "My_File.xlsx")



# PLOT ROC CURVE

ppi<-300

png(filename="ROC curve.png", width=6*ppi, height=6*ppi,res=ppi)

plot(normal.LRPerf, col=2, main='ROC curve')

abline(a=0, b=1, lty=2, lwd=3, col="black")

legend(0.5,0.5, c('Actual data','Under sampled data','Over sampled data', ''), 2:4)

plot(usample.LRPerf, col=3, add=TRUE)

plot(osample.LRPerf, col=4, add=TRUE)


dev.off()

osample.logisticTrainingFit
```

## Decision Tree [Back]

```
library(ISLR)

a<-table(Caravan$Purchase)


colors=c("red","green")

col=colors

pie(a,main = "CUSTOMERS OF CARAVAN POLICY",col=colors)

box()



library(rpart)

library(rattle)
```

```
## Loading required package: RGtk2
## Rattle: A free graphical interface for data mining with R.
## Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
library(rpart.plot)
library(RColorBrewer)
library(crossval)
library(gplots)
library(rpart)
library(rattle)
library(rpart.plot)
library(RColorBrewer)
library(crossval)
library(gplots)
library(vcd)
library(Metrics)
##
## Attaching package: 'gplots'
##
## The following object is masked from 'package:stats':
##
##     lowess
library(vcd)
## Loading required package: grid
##
## Attaching package: 'vcd'
##
## The following object is masked from 'package:ISLR':
##
##     Hitters
library(Metrics)
library("plyr")


######## Read the Caravan.csvfile


library(readr)

caravan_df_train <- read_csv("~/.System/Desktop/Caravan Insurance - 
R/DATA/TrainingData_V1.csv")

caravan_df_train <- read.csv('~/.System/Desktop/Caravan Insurance - 
R/DATA/Training&Test_Data.csv')

caravan_df_test <- read.csv('~/.System/Desktop/Caravan Insurance - R/DATA/TestDataV3.csv')

caravan_df_test <- rename(caravan_df_test,c(CARAVAN="Purchase"))

caravan_df_train <- rename(caravan_df_train,c(CARAVAN="Purchase"))


# View(caravan_df_train)


select_features =1

if (select_features == 1) { # Select features


  keeps_Lau <- c("fire_policies", "private_third_party_insurance", "car_policies", 
"Customer_Subtype",

                         "Average_income", "Purchasing_power", "High_level_education",

                         "Income_._30", "Social_class_A", "Number_of_boat_policies", 
"Lower_level_education",

                         "boat_policies", "No_car", "Married", "social_security_insurance_policies", 
"Purchase")


  keeps1 <- 
c("MOPLHOOG","MOPLMIDD","MOPLLAAG","MINKM30","MINK3045","MINK4575","MINK7512",

            "MINK123M","MINKGEM","MAUT1","MAUT2","MAUT0","PPERSAUT","APERSAUT", 
"Purchase")


  keeps2 <- 
c("ALEVEN","MAUT0","MBERBOER","MINKM30","MKOOPKLA","MOSTYPE","MSKA","PBRAND","PPE
RSAUT","MOPLLAAG","MRELGE","APLEZIER","PWAPART"
```

```
          ,"MSKB2","MGODGE","MRELOV","MOPLHOOG","MOPLMIDD","MINK3045","MINK4575","M
INK7512","MINK123M","MINKGEM","MAUT1","MAUT2","APERSAUT", "Purchase")


  keeps3 <- c("MOPLHOOG","MOPLMIDD","MOPLLAAG","MINKM30","MINK4575",
"MINK3045","MINK7512",

          "MINK123M","MINKGEM","MAUT1","MAUT2","MAUT0","PPERSAUT","APERSAUT",
"Purchase")




  keeps_boruta <-
c("PBRAND","PWAPART","PPERSAUT","MOSTYPE","MINKGEM","MKOOPKLA","MOPLHOOG","MOSH
OOFD","MINKM30","MSKA","APLEZIER",

              "MOPLLAAG","PPLEZIER","MAUT0","MRELGE","MINK4575",


"MRELOV","MHHUUR","MHKOOP","MAUT1","MSKC","MZPART","MFALLEEN","MBERHOOG","MFW
EKIND", "MINK3045", "Purchase")




  keeps_vikas <-
c("PPERSAUT","MAUT0","MOPLLAAG","MINK3045","MINKM30","MKOOPKLA","MOSTYPE","MOPLH
OOG","MINKGEM", "Purchase")

  keeps_vikas <- c("PPERSAUT","MAUT0","MOPLLAAG", "MINK3045", "Purchase")


    caravan_df_train <- caravan_df_train[keeps_vikas]


    library(dummies)

    #caravan_df_train <- dummy.data.frame(caravan_df_train, sep = ".", names = c("MINK3045"))





  }
```

```
undersampled = 1


if (undersampled == 1){
   # **********************************************
   #                              -- Data Undersampled --
   # **********************************************
   caravan_df_train_LR_Under <- caravan_df_train[caravan_df_train$Purchase==1,]
   length(caravan_df_train_LR_Under$Purchase)
   #Randomly select observations with dependent variable = 0
   zeroObs <- caravan_df_train[caravan_df_train$Purchase==0,]
   set.seed(123457)
   # SAMPLING SELECTION !!!!!
   samplin_level = 1
   samplin_level = 1.5
   samplin_level = 1.5
   rearrangedZeroObs <-  zeroObs[sample(nrow(zeroObs), 
length(caravan_df_train_LR_Under$Purchase) * samplin_level ),]


   #Appending rows of randomly selected 0s in our undersampled data frame
   caravan_df_train_LR_Under <- rbind(caravan_df_train_LR_Under, rearrangedZeroObs)
   length(caravan_df_train_LR_Under$Purchase)


   #Let's verify that number of 1s and 0s in our undersampled data are equal
   undersampled.Frequency.Purchase <- data.frame(Purchase = 
levels(as.factor(caravan_df_train_LR_Under$Purchase)), Count = 
as.numeric(table(caravan_df_train_LR_Under$Purchase)))
   undersampled.Frequency.Purchase
}
 d1<- caravan_df_train
 d1<- caravan_df_train_LR_Under
 d1.ori<-d1
 set.seed(99)


 tr <- d1.ori[sample(row.names(d1.ori), size = round(nrow(d1.ori)*0.5)), ]
 te <- d1.ori[!(row.names(d1.ori) %in% row.names(tr)), ]


 tr1 <- tr
 te1  <- te
 te2 <-te


#tr1 <- caravan_df_train
#te1  <- caravan_df_test
#te2 <-caravan_df_test


te2$Purchase <- rep(0,nrow(te2))
tr1$Purchase = as.factor(tr1$Purchase)



######################




fit3 <- rpart(formula=Purchase ~ .,data=tr1,
            control=rpart.control(minsplit=10, minbucket=5, cp=0.01444043))


fit1 <- rpart(formula=Purchase ~ .,data=tr1,
             control=rpart.control(minsplit=50, minbucket=10, cp=0.01))
fit1



fit1$cptable[which.min(fit1$cptable[,"xerror"]),"CP"] # Prunning tree
```

```
summary(fit1)


gc()
fancyRpartPlot(fit1)


rsq.rpart(fit1)


printcp(fit1)


print(fit1)


plot(fit1)
text(fit1)



fit1$cptable[which.min(fit1$cptable[,"xerror"]),"CP"] # PRUNE THE


Prediction<-predict(fit1,te1,type="class")


te2$Purchase <- Prediction


Pred = factor(as.factor(te2$Purchase), c(0, 1), labels = c("Not purchased", "Purchased"))
Actual = factor(as.factor(te1$Purchase), c(0, 1), labels = c("Not purchased", "Purchased"))
table(te1$Purchase)


cm1 = confusionMatrix(Actual,Pred, negative = "Not purchased")
cm1


# corresponding accuracy, sensitivity etc.
diagnosticErrors(cm1)
```

```
# Compute the classification error

ce(Actual,Pred)



gc()

fancyRpartPlot(fit1)
```

## Random Forest [Back]

```
#Function to build random forest model
buildModel<-function(trainData,testData,ntrees=100,nodeSize=1){
  #build random forest model
  model<-randomForest(trainData[,-ncol(trainData)],
                      trainData[,ncol(trainData)],
                      xtest=testData[,-ncol(testData)],
                      ytest=testData[,ncol(testData)],
                      ntree=ntrees,
                      nodesize=nodeSize,
                      proximity=TRUE,
                      importance=TRUE,
                      keep.forest=TRUE)
  #Return model
  return(model)
}

#Print Error rates and accuracies
displayResultsFromModel<-function(model,trainRows,testRows){
  print("TRAIN")
  #Train OOB Error
  print(paste("Train OOB Error: ",
              model$err.rate[nrow(model$test$err.rate),
                             1,
                             drop=FALSE],sep=""))
  #Train Factor Level 0 Error
  print(paste("Train CARAVAN=0 Error: ",model$err.rate[nrow(model$test$err.rate),
                                          2,
                                          drop=FALSE],sep=""))
  #Train Factor Level 1 Error
  print(paste("Train CARAVAN=1 Error: ",model$err.rate[nrow(model$test$err.rate),
                                          3,
                                          drop=FALSE],sep=""))
  #Train Accuracy
  trainAuc<-sum(diag(model$confusion))/trainRows
  print(paste("Train Accuracy: ",trainAuc,sep=""))

  #Print blank line between train and test results
  print(" ")

  print("TEST")
  #Test Error
  print(paste("Test Error: ",model$test$err.rate[nrow(model$test$err.rate),
                                      1,
                                      drop=FALSE],sep=""))
  #Train Factor Level 0 Error
  print(paste("Test CARAVAN=0 Error: ",model$test$err.rate[nrow(model$test$err.rate),
                                             2,
                                             drop=FALSE],sep=""))
  #Train Factor Level 1 Error
  print(paste("Test CARAVAN=1 Error: ",model$test$err.rate[nrow(model$test$err.rate),
                                             3,
                                             drop=FALSE],sep=""))
  #Test Accuracy
  testAuc<-sum(diag(model$test$confusion))/testRows
```

```
  print(paste("Test Accuracy: ",testAuc,sep=""))


}


#Function to perform 10 fold cross validation
validateModel <- function(data,ntrees=100,nodeSize=1){
  #Frame to hold results
  results<-data.frame(OOB=as.numeric(),
                      trainFalseError=as.numeric(),
                      trainTrueError=as.numeric(),
                      testError=as.numeric(),
                      testFalseError=as.numeric(),
                      testTrueError=as.numeric(),
                      trainAccuracy=as.numeric(),
                      testAccuracy=as.numeric())
  #Folds generated using Caret packages createFolds
  folds<-createFolds(data$CARAVAN,k=10,list=TRUE,returnTrain=FALSE)
  for (i in 1:10){
    #Keep one set for testing, rest training
    trainData<-data[-c(folds[[i]]),]
    testData<-data[c(folds[[i]]),]
    model<-randomForest(trainData[,-ncol(trainData)],
                        trainData[,ncol(trainData)],
                        xtest=testData[,-ncol(testData)],
                        ytest=testData[,ncol(testData)],
                        ntree=ntrees,
                        nodesize=nodeSize,
                        proximity=TRUE)
    #TRAIN
    oob<-model$err.rate[nrow(model$test$err.rate),1,drop=FALSE]
    trainFalse<-model$err.rate[nrow(model$test$err.rate),2,drop=FALSE]
    trainTrue<-model$err.rate[nrow(model$test$err.rate),3,drop=FALSE]
    trainAccuracy<-sum(diag(model$confusion))/nrow(trainData)
    #TEST
    testError<-model$test$err.rate[nrow(model$test$err.rate),1,drop=FALSE]
    testFalse<-model$test$err.rate[nrow(model$test$err.rate),2,drop=FALSE]
    testTrue<-model$test$err.rate[nrow(model$test$err.rate),3,drop=FALSE]
    testAccuracy<-sum(diag(model$test$confusion))/nrow(testData)
    #Create new Row in results with values
    results[nrow(results)+1,]<-c(oob,
                            trainFalse,
                            trainTrue,
                            testError,
                            testFalse,
                            testTrue,
                            trainAccuracy,
                            testAccuracy)
  }
  #Return results
  return(results)
}

#Takes results and displays them as a whole and with averages
displayResults<-function(results){
  Position=c(1:10)
  #PLOT COLUMNS
  #TRAIN
  #OOB
  plot<-ggplot(results,aes(x=Position,y=OOB))
  plot<-plot + geom_point()
  plot<-plot + geom_smooth()
  plot<-plot + labs(title="OOB")
```

```
print(plot)
#Train Caravan=0 Error
plot<-ggplot(results,aes(x=Position,y=trainFalseError))
plot<-plot + geom_point()
plot<-plot + geom_smooth()
plot<-plot + labs(title="Train Caravan=0 Error")
print(plot)
#Train Caravan=1 Error
plot<-ggplot(results,aes(x=Position,y=trainTrueError))
plot<-plot + geom_point()
plot<-plot + geom_smooth()
plot<-plot + labs(title="Train Caravan=1 Error")
print(plot)
#Train Accuracy
plot<-ggplot(results,aes(x=Position,y=trainAccuracy))
plot<-plot + geom_point()
plot<-plot + geom_smooth()
plot<-plot + labs(title="Train Accuracy")
print(plot)

#TEST
#Test Error
plot<-ggplot(results,aes(x=Position,y=testError))
plot<-plot + geom_point()
plot<-plot + geom_smooth()
plot<-plot + labs(title="Test Error")
print(plot)
#Test Caravan=0 Error
plot<-ggplot(results,aes(x=Position,y=testFalseError))
plot<-plot + geom_point()
plot<-plot + geom_smooth()
```

```
plot<-plot + labs(title="Test Caravan=0 Error")
print(plot)
#Test Caravan=1 Error
plot<-ggplot(results,aes(x=Position,y=testTrueError))
plot<-plot + geom_point()
plot<-plot + geom_smooth()
plot<-plot + labs(title="Test Caravan=1 Error")
print(plot)
#Test Accuracy
plot<-ggplot(results,aes(x=Position,y=testAccuracy))
plot<-plot + geom_point()
plot<-plot + geom_smooth()
plot<-plot + labs(title="Test Accuracy")
print(plot)

#AVERAGES
#TRAIN
#OOB
print(paste("Average OOB: ",
        sum(results$OOB)/nrow(results),sep=""))
#Train CARAVAN=0 Error
print(paste("Average CARAVAN=0 Error: ",
        sum(results$trainFalseError)/nrow(results),sep=""))
#Train Caravan=1 Error
print(paste("Average CARAVAN=1 Error: ",
        sum(results$trainTrueError)/nrow(results),sep=""))
#Train Accuracy
print(paste("Average Train Accuracy: ",
        sum(results$trainAccuracy)/nrow(results),sep=""))

#Print blank line between train and test results
```

```
  print(" ")

  #Test Error
  print(paste("Average Test Error: ",
            sum(results$testError)/nrow(results),sep=""))
  #Test CARAVAN=0 Error
  print(paste("Average CARAVAN=0 Error: ",
            sum(results$testFalseError)/nrow(results),sep=""))
  #Test CARAVAN=1 Error
  print(paste("Average CARAVAN=1 Error: ",
            sum(results$testTrueError)/nrow(results),sep=""))
  #Test Accuracy
  print(paste("Average Test Accuracy: ",
            sum(results$testAccuracy)/nrow(results),sep=""))
}



#Using same train and test set as before
#Tweak number of trees
testNTrees <- function(trainData,testData){
  ntrees<-20
  results<-NULL
  results<-data.frame(NTrees=as.numeric(),
                  OOB=as.numeric(),
                  trainFalseError=as.numeric(),
                  trainTrueError=as.numeric(),
                  testError=as.numeric(),
                  testFalseError=as.numeric(),
                  testTrueError=as.numeric(),
                  trainAccuracy=as.numeric(),
          testAccuracy=as.numeric())
 for (i in 1:9){
  trainData=train
  testData=test
  model<-randomForest(trainData[,-ncol(trainData)],
           trainData[,ncol(trainData)],
           xtest=testData[,-ncol(testData)],
           ytest=testData[,ncol(testData)],
           ntree=ntrees,
           proximity=TRUE)
  #TRAIN
  oob<-model$err.rate[nrow(model$test$err.rate),1,drop=FALSE]
  trainFalse<-model$err.rate[nrow(model$test$err.rate),2,drop=FALSE]
  trainTrue<-model$err.rate[nrow(model$test$err.rate),3,drop=FALSE]
  trainAccuracy<-sum(diag(model$confusion))/nrow(trainData)
  #TEST
  testError<-model$test$err.rate[nrow(model$test$err.rate),1,drop=FALSE]
  testFalse<-model$test$err.rate[nrow(model$test$err.rate),2,drop=FALSE]
  testTrue<-model$test$err.rate[nrow(model$test$err.rate),3,drop=FALSE]
  testAccuracy<-sum(diag(model$test$confusion))/nrow(testData)
  #Create new row in results with new data
  results[nrow(results)+1,]<-c(ntrees,
               oob,
               trainFalse,
               trainTrue,
               testError,
               testFalse,
               testTrue,
               trainAccuracy,
               testAccuracy)
  results
```

```
    ntrees <-ntrees + 10
  }
  #return max row
  ntrees<-results$NTrees[which.max(results$testAccuracy)]
  return(ntrees)
}


#Tweek Nodesize
testNodeSize <- function(trainData,testData,ntrees){
  nsize<-0
  results<-data.frame(Nodesize=as.numeric(),
                    OOB=as.numeric(),
                    trainFalseError=as.numeric(),
                    trainTrueError=as.numeric(),
                    testError=as.numeric(),
                    testFalseError=as.numeric(),
                    testTrueError=as.numeric(),
                    trainAccuracy=as.numeric(),
                    testAccuracy=as.numeric())
  for (i in 1:floor(nrow(trainData)/100)){
    model<-randomForest(trainData[,-ncol(trainData)],
                      trainData[,ncol(trainData)],
                      xtest=testData[,-ncol(testData)],
                      ytest=testData[,ncol(testData)],
                      ntree=ntrees,
                      proximity=TRUE)
    #TRAIN
    oob<-model$err.rate[nrow(model$test$err.rate),1,drop=FALSE]
    trainFalse<-model$err.rate[nrow(model$test$err.rate),2,drop=FALSE]
    trainTrue<-model$err.rate[nrow(model$test$err.rate),3,drop=FALSE]
    trainAccuracy<-sum(diag(model$confusion))/nrow(trainData)
    #TEST
    testError<-model$test$err.rate[nrow(model$test$err.rate),1,drop=FALSE]
    testFalse<-model$test$err.rate[nrow(model$test$err.rate),2,drop=FALSE]
    testTrue<-model$test$err.rate[nrow(model$test$err.rate),3,drop=FALSE]
    testAccuracy<-sum(diag(model$test$confusion))/nrow(testData)
    results[nrow(results)+1,]<-c(nsize,
                      oob,
                      trainFalse,
                      trainTrue,
                      testError,
                      testFalse,
                      testTrue,
                      trainAccuracy,
                      testAccuracy)
    nsize<-nsize+1
  }
  #Return node size
  nodeSize<-results$Nodesize[which.max(results$testAccuracy)]
  return(nodeSize)
}



library(ISLR)
library(rpart)
library(rattle)
library(rpart.plot)
library(RColorBrewer)
library(crossval)
library(gplots)
library(rpart)
library(rattle)
library(rpart.plot)

library(RColorBrewer)

library(crossval)

library(gplots)

library(vcd)

library(Metrics)

library(vcd)

library(Metrics)

library("plyr")

library(ggplot2) #Data Visualisation

library(dplyr) #Renaming

library(ROSE) #Sampling

library(caret) #Partitioning

library(randomForest) #rf


library(randomForestExplainer)





######## Read the Caravan.csvfile


library(readr)

# caravan_df_train <- read_csv("~/.System/Desktop/Caravan Insurance -
R/DATA/TrainingData_V1.csv")

caravan_df_all <- read.csv('~/.System/Desktop/Caravan Insurance -
R/DATA/Training&Test_Data.csv')

# caravan_df_test <- read.csv('~/.System/Desktop/Caravan Insurance - R/DATA/TestDataV3.csv')

# caravan_df_test <- rename(caravan_df_test,c(CARAVAN="CARAVAN"))

# caravan_df_all <- rename(caravan_df_all,c(CARAVAN="CARAVAN"))


# View(caravan_df_all)
```

```
select_features =1

if (select_features == 1) { # Select features


  keeps_Lau <- c("fire_policies", "private_third_party_insurance", "car_policies",
"Customer_Subtype",

            "Average_income", "Purchasing_power", "High_level_education",

            "Income_._30", "Social_class_A", "Number_of_boat_policies", "Lower_level_education",

            "boat_policies", "No_car", "Married", "social_security_insurance_policies", "CARAVAN")

  keeps1 <-
c("MOPLHOOG","MOPLMIDD","MOPLLAAG","MINKM30","MINK3045","MINK4575","MINK7512",

          "MINK123M","MINKGEM","MAUT1","MAUT2","MAUT0","PPERSAUT","APERSAUT",
"CARAVAN")

  keeps2 <-
c("ALEVEN","MAUT0","MBERBOER","MINKM30","MKOOPKLA","MOSTYPE","MSKA","PBRAND","PPE
RSAUT","MOPLLAAG","MRELGE","APLEZIER","PWAPART"


,"MSKB2","MGODGE","MRELOV","MOPLHOOG","MOPLMIDD","MINK3045","MINK4575","MINK751
2","MINK123M","MINKGEM","MAUT1","MAUT2","APERSAUT", "CARAVAN")

  keeps3 <- c("MOPLHOOG","MOPLMIDD","MOPLLAAG","MINKM30","MINK4575",
"MINK3045","MINK7512",

          "MINK123M","MINKGEM","MAUT1","MAUT2","MAUT0","PPERSAUT","APERSAUT",
"CARAVAN")

  keeps_boruta <-
c("PBRAND","PWAPART","PPERSAUT","MOSTYPE","MINKGEM","MKOOPKLA","MOPLHOOG","MOSH
OOFD","MINKM30","MSKA","APLEZIER",

             "MOPLLAAG","PPLEZIER","MAUT0","MRELGE","MINK4575",


"MRELOV","MHHUUR","MHKOOP","MAUT1","MSKC","MZPART","MFALLEEN","MBERHOOG","MFW
EKIND", "MINK3045", "CARAVAN")

  keeps_vikas <-
c("PPERSAUT","MAUT0","MOPLLAAG","MINK3045","MINKM30","MKOOPKLA","MOSTYPE","MOPLH
OOG","MINKGEM", "CARAVAN")

  keeps_vikas <- c("PPERSAUT","MAUT0","MOPLLAAG", "MINK3045", "CARAVAN")


  caravan_df_all <- caravan_df_all[keeps_vikas]
  library(dummies)

  #caravan_df_all <- dummy.data.frame(caravan_df_all, sep = ".", names = c("MINK3045"))


}


undersampled = 1

if (undersampled == 1){

  # *********************************************

  #                                   -- Data Undersampled --

  # *********************************************

  caravan_df_all_LR_Under <- caravan_df_all[caravan_df_all$CARAVAN==1,]

  length(caravan_df_all_LR_Under$CARAVAN)

  #Randomly select observations with dependent variable = 0

  zeroObs <- caravan_df_all[caravan_df_all$CARAVAN==0,]

  set.seed(123457)

  # SAMPLING SELECTION !!!!!

  samplin_level = 1

  samplin_level = 1.5

  samplin_level = 1.5

  rearrangedZeroObs <-  zeroObs[sample(nrow(zeroObs),
length(caravan_df_all_LR_Under$CARAVAN) * samplin_level ),]


  #Appending rows of randomly selected 0s in our undersampled data frame

  caravan_df_all_LR_Under <- rbind(caravan_df_all_LR_Under, rearrangedZeroObs)

  length(caravan_df_all_LR_Under$CARAVAN)


  #Let's verify that number of 1s and 0s in our undersampled data are equal

  undersampled.Frequency.CARAVAN <- data.frame(CARAVAN =
levels(as.factor(caravan_df_all_LR_Under$CARAVAN)), Count =
as.numeric(table(caravan_df_all_LR_Under$CARAVAN)))

  undersampled.Frequency.CARAVAN

  df<- caravan_df_all_LR_Under
```

```
} else { # Normal
  df<- caravan_df_all
}

df$CARAVAN = as.factor(df$CARAVAN)

# Random forest: create functions

random_forest_functions = 1
if (random_forest_functions == 1){
  source("~/.System/Desktop/Caravan Insurance - R/CODE/Random_Forest(Functions).R")
}

#Partition dataset using caret

part<-createDataPartition(y=df$CARAVAN,p=0.7,list=FALSE)
train<-df[part,]
test<-df[-part,]

#Final model
model<-buildModel(train,test,ntrees=1,nodeSize=24)
#Display results
trainRows<-nrow(train)
testRows<-nrow(test)
displayResultsFromModel(model,trainRows,testRows)

ntrees <- testNTrees(trainData,testData)



results <- validateModel(df)
```

```
getTree(model, k=1, labelVar=TRUE)
```

```
ntree <- 5
library("party")
cf <- cforest(CARAVAN~., data=train,controls=cforest_control(ntree=ntree))
```

```
for(i in 1:ntree){
  pt <- prettytree(cf@ensemble[[i]], names(cf@data@get("input")))
  nt <- new("BinaryTree")
  nt@tree <- pt
  nt@data <- cf@data
  nt@responses <- cf@responses

  pdf(file=paste0("filex",i,".pdf"))
  plot(nt, type="simple")
  dev.off()

}
```