PREDICTING CLAIM COUNTS FOR MOTORCYCLE INSURANCE

Evaluation of machine learning approaches in adding value to traditional methods

Akira Kinefuchi

MQBS Data Analysis: An enhanced toolkit for Actuaries

INTRODUCTION

Rapid technological advancement in the modern era has opened a gateway for new techniques in data modelling. In this project, popular modern machine learning techniques are explored and compared against traditional actuarial approaches in predicting claim counts for a motorcycle insurance portfolio.

A well-known and established traditional approach in estimating claim counts in general insurance is Generalised Linear Model (GLM). In this paper, GLMs are used to fit the data first and will act as a benchmark.

The investigated machine learning techniques are mainly supervised and includes Neural Network (NN) and Regression Trees (RT). We will look at multiple models for each, initially fitting them independent of other models. Subsequently, we will "boost" the GLM benchmark fit using these techniques as an extension.

Finally, we will briefly touch on one of the unsupervised machine learning techniques called K-means clustering to see how much explanatory power it adds to the GLM fits.

The intention of this project is to showcase the usefulness of (or lack thereof) machine learning techniques on top of the traditional approaches. I have received strong guidance and inspiration from the course material as well as from the case study paper "Case Study: French Motor Third-Party Liability Claims" by Wuthrich M.V., Salzmann R., Noll A. (2020). This project can be understood as a branched version of the paper on Swedish motorcycle claims with an extension to various GLM boosting methods and studying the effect of using unsupervised learning in optimising traditional models.

Relevant files including the R scripts and report is also available on GitHub (https://github.com/AkiraK92/MQBSCapstoneProject.git).

EXPLORATORY DATA ANALYSIS

Information on the data

The project utilises data from a former insurance company in Sweden, Wasa, which provided partial casco insurance to motorcyclists[1]. The data is from 1994 to 1998 and is aggregated on all policies and claims. Unfortunately, more recent data was not publicly available due to confidentiality issues.

The data is taken directly from an R package called insuranceData (dataOhlsson). It originally has 64,548 observations and 9 variables. Since the original variable names are in Swedish, they are converted into English names for better interpretability. A brief data description and converted names (in brackets) for each variable is shown below.

1. agarald (ownAge): *numeric;* owner age of the motorcycle
2. kon (gender): *factor;* gender of the owner with two level K (female) and M (male)
3. zon (zone): *numeric;* geographic zones from 1 to 7 representing standard Swedish parishes
4. mcklass (mcCl): *numeric;* classification based on EV ratio (Engine power in kW x 100)/(Vehicle weight in kg + 75) with seven classes 1 to 7. EV ratio can be understood as standardised vehicle power
5. fordald (vehAge): *numeric;* vehicle age of the motorcycle
6. bonuskl (bonCl): *numeric;* bonus class from 1 to 7, with higher classes representing safer drivers. A new driver starts from 1 and moves up a class for each year with no claims. If a claim is made, the driver moves down two bonus classes
7. duration (duration): *numeric;* exposure in policy year units
8. antskad (claimsNum): *numeric;* total number of claims associated with a policy
9. skadkost (claimsAmt): *numeric;* total claims cost associated with a policy

Figure 1.1: Command output from R on data structure

```
'data.frame':   64548 obs. of  9 variables:
 $ ownAge   : int  0 4 5 5 6 9 9 9 10 10 ...
 $ gender   : Factor w/ 2 levels "K","M": 2 2 1 1 1 1 2 2 2 ...
 $ zone     : int  1 3 3 4 2 3 4 4 2 4 ...
 $ mcCl     : int  4 6 3 1 1 3 3 4 3 2 ...
 $ vehAge   : int  12 9 18 25 26 8 6 20 16 17 ...
 $ bonCl    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ duration : num  0.175 0 0.455 0.173 0.181 ...
 $ claimsNum: int  0 0 0 0 0 0 0 0 0 0 ...
 $ claimsAmt: int  0 0 0 0 0 0 0 0 0 0 ...
```

Note the response variable of interest is the claimsNum. Assuming independence between claim frequency and severity, we discard claimsAmt from this analysis[2]. There are counterarguments on the independence assumption, however, this is outside the scope of this project [3].

Pre-processing of data

While the data was already available as a data frame in tidy format, basic checks were performed

[1] Ohlsson E., Johansson B. (2010), Non-life insurance pricing with generalized linear models, Springer
[2] Renshaw A.E. (1994) Modelling the claims process in the presence of covariates, ASTIN Bulletin, 24 (2) (1994), pp. 265-285
[3] Schulz J., Genest C., Garrido J. (2016), Generalized linear models for dependent frequency and severity of insurance claims, Insurance: Mathematics and Economics, Volume 70, September 2016, Pages 205-215

to validate the entries before proceeding to data analysis.

Firstly, I searched for any missing entries, but none were found. Hence, we can carry forward without the need to treat unknown observations.

Figure 2.1: Command output from R on missing entry search

```
> apply(data, 2, function(x) any(is.na(x)))
  ownAge   gender     zone     mcCl   vehAge    bonCl duration claimsNum claimsAmt
   FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE     FALSE     FALSE
> apply(data, 2, function(x) any(is.null(x)))
  ownAge   gender     zone     mcCl   vehAge    bonCl duration claimsNum claimsAmt
   FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE     FALSE     FALSE
```

Secondly, we investigated and searched for any anomalies in each variable. Those requiring attention are outlined as below.

1. Owner Age (ownAge)
From Figure 2.2, the age range varies from 0 to 92. The minimum legal driving age for motorcycles in Sweden is 16 and it is unrealistic for any owner to be below this age[4]. Those below 16 only account for 0.03% of the total exposure (duration) and has zero claim counts in total, and we expect these to be data entry errors so decide to put a floor at 16.

2. Vehicle Age (VehAge)
Motorcycles that have extreme ages seem unreasonable. For example, if we look at the proportion of observations with the vehicle age above 20, it accounts for 3.6% of the total exposure and 0.8% of total number of claims. While this seems to be low in proportion, given the original size of the dataset it may create biases if they are simply removed. In fact, insured vehicles could be very old in age if they are well-maintained or been garaged for a long time without being used too frequently, so we decide in the end to keep these observations as they are.

3. Duration (duration)
The data is from 1994 to 1998, so the maximum policy duration should be 5 years. However, there are policies above this ceiling which accounts for 13.4% and 4.2% respectively of the total exposure and claims in the dataset. This is significant so cannot be blindly removed. Ideally, we should speak to relevant stakeholders responsible for data entry (e.g. IT, policy administrator, sales department), however, this is near impossible given the nature of the dataset. Therefore, we tackle this issue by capping the duration at the maximum 5 years for the purpose of this project.

In addition, we found some entries with zero duration. There were over 2000 such policies (3.2 % of policy counts) which accounted for 0.5% of the total claim counts. While this seems relatively low, we cannot remove them without further information for reasons similar to above. It was decided that these values should be replaced by the mean duration of the rest of the dataset, after the capping procedure.

Figure 2.2: Summary of the original dataset (R command)

---

[4] Gregersen N.P. et al. (2000), Sixteen years age limit for learner drivers in Sweden – an evaluation of safety effects <https://pubmed.ncbi.nlm.nih.gov/10576673/>

```
    ownAge         gender        zone          mcCl          vehAge        bonCl         duration        claimsNum
 Min.   : 0.00   K: 9853   Min.   :1.000   Min.   :1.0   Min.   : 0.00   Min.   :1.000   Min.   : 0.0000   Min.   :0.0000
 1st Qu.:31.00   M:54695   1st Qu.:2.000   1st Qu.:3.0   1st Qu.: 5.00   1st Qu.:2.000   1st Qu.: 0.4630   1st Qu.:0.0000
 Median :44.00             Median :3.000   Median :4.0   Median :12.00   Median :4.000   Median : 0.8274   Median :0.0000
 Mean   :42.42             Mean   :3.213   Mean   :3.7   Mean   :12.54   Mean   :4.025   Mean   : 1.0107   Mean   :0.0108
 3rd Qu.:52.00             3rd Qu.:4.000   3rd Qu.:5.0   3rd Qu.:16.00   3rd Qu.:7.000   3rd Qu.: 1.0000   3rd Qu.:0.0000
 Max.   :92.00             Max.   :7.000   Max.   :7.0   Max.   :99.00   Max.   :7.000   Max.   :31.3397   Max.   :2.0000
```

Figure 2.3 shows the descriptive statistics on the modified dataset. Since what we omitted was only the variable on claims amount, the total number of observations remain at 64,548 policies with 8 variables.

Figure 2.3: Summary of the modified dataset (R command)

```
    ownAge         gender        zone          mcCl          vehAge        bonCl         duration        claimsNum
 Min.   :16.00   K: 9853   Min.   :1.000   Min.   :1.0   Min.   : 0.00   Min.   :1.000   Min.   :0.00274   Min.   :0.0000
 1st Qu.:31.00   M:54695   1st Qu.:2.000   1st Qu.:3.0   1st Qu.: 5.00   1st Qu.:2.000   1st Qu.:0.49589   1st Qu.:0.0000
 Median :44.00             Median :3.000   Median :4.0   Median :12.00   Median :4.000   Median :0.92329   Median :0.0000
 Mean   :42.42             Mean   :3.213   Mean   :3.7   Mean   :12.54   Mean   :4.025   Mean   :0.98921   Mean   :0.0108
 3rd Qu.:52.00             3rd Qu.:4.000   3rd Qu.:5.0   3rd Qu.:16.00   3rd Qu.:7.000   3rd Qu.:1.00274   3rd Qu.:0.0000
 Max.   :92.00             Max.   :7.000   Max.   :7.0   Max.   :99.00   Max.   :7.000   Max.   :5.00000   Max.   :2.0000
```

Exploratory Data Analysis

First, we look at the histogram of each explanatory variable (Figure 3.1). General observations we make are a) age group is bimodal with the cut-off around 30-35 years of age, separating apart relatively young and old drivers, b) male drivers are dominant in the dataset, c) zonal accumulation seems to cluster around zone 4 but no clear trend in the zone number as expected so it is best treated as a non-ordinal categorical variable, d) MC class represents vehicle power and we see most vehicles are centred around the middle (i.e. class 4), e) vehicle age has a positive skew meaning there are more vehicles that are relatively new and as we saw earlier we will treat them as they are, and finally f) majority of observed policies are in the highest bonus class which suggests good driving history, whereas the second highest is within the first class which is expected as it includes new drivers/policyholders.

Figure 3.1: Histogram of each explanatory variable
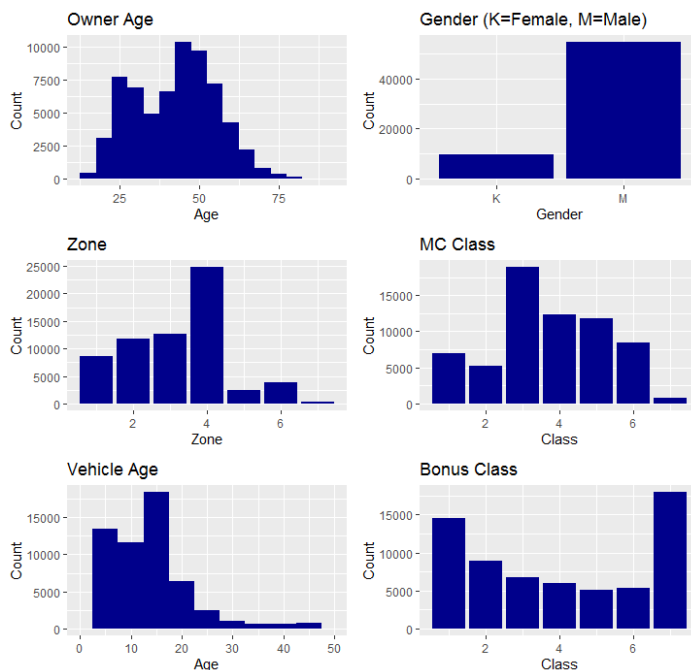
For the variable of interest, Figure 3.2 shows respective histograms and boxplots. Most policy duration is within 1 year and only a small fraction above 1. Similarly, the number of claims is at max 2 and most policies have nil claims. Therefore, we have positive skews for both these variables, which supports a Poisson model. The overall claim frequency (computed as total number of claims divided by total policy durations) from the dataset is 1.092%.

Figure 3.2: Histogram and boxplots of duration and number of claims
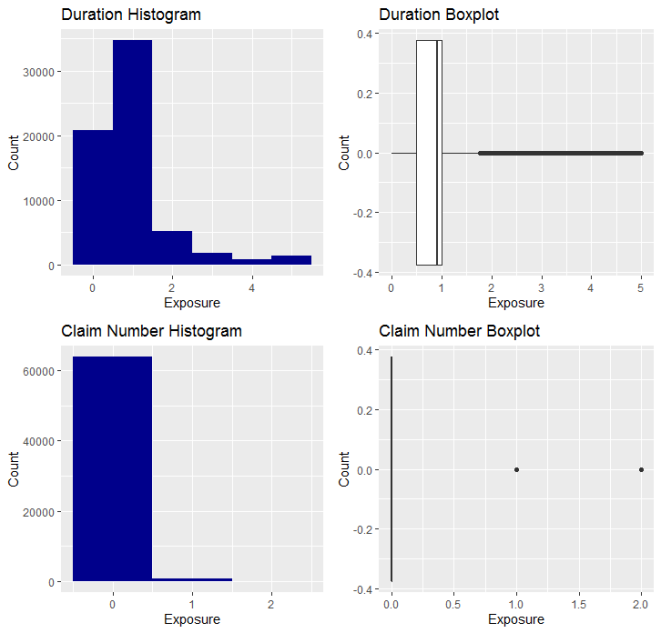
We then explore the relationship between claims frequency by each possible explanatory variable.

In Figure 4.1, we see that the exposures naturally centres around age group 4, which represents those between 46 and 55. We also observe a negative relationship with owner age group and claim frequency. In other words, the claim frequency per policy year is higher for younger owners.

Figure 4.1: Exposure and claim frequency by owner age group

| Age Category	
1	<= 25
2	26-35
3	36-45
4	46-55
5	56-65
6	66-75
7	76-85
8	>85
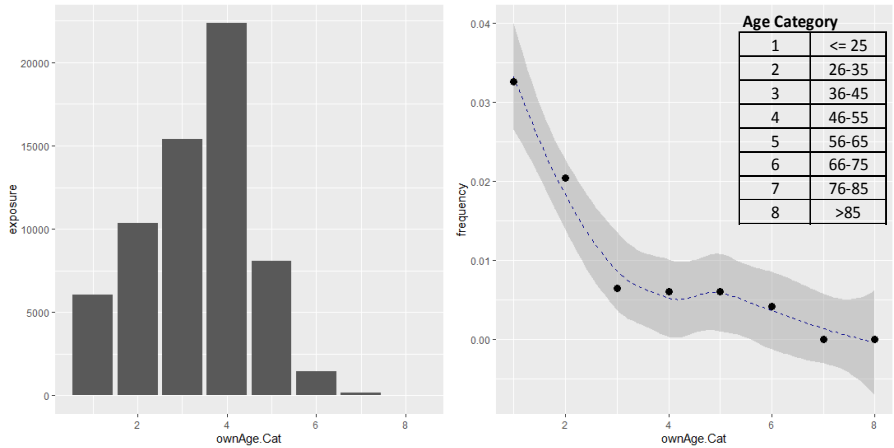
In Figure 4.2, there is an obvious accumulation in zone 4 which is in line with what was observed in the histogram. It is interesting that there is a negative relationship between the zones and claim frequency as this may suggest zones are ordinal. As we will show later in the correlation plot there is no strong collinearity with other explanatory variables so there could be other underlying reason for this relationship. However, we could not obtain detailed information on parishes neither from the source nor online searches, so we assume that this is not due to causation and continue to use zone as non-ordinal categorical variable in our following data analysis.

Figure 4.2: Exposure and claim frequency by geographical zone
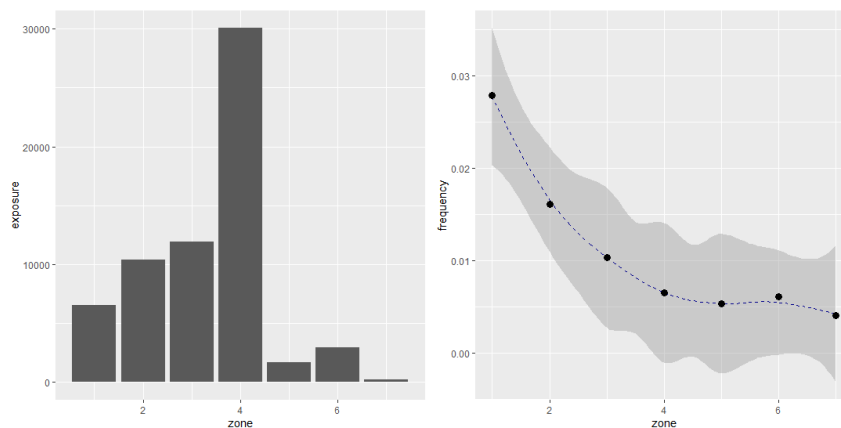
From Figure 4.3, we observe that both exposure and claim frequency is much higher for male owners. However, note the frequency scale here is different to other graphs and the difference between genders may be smaller than how it appears.

Figure 4.3: Exposure and claim frequency by gender (K=Female, M=Male)
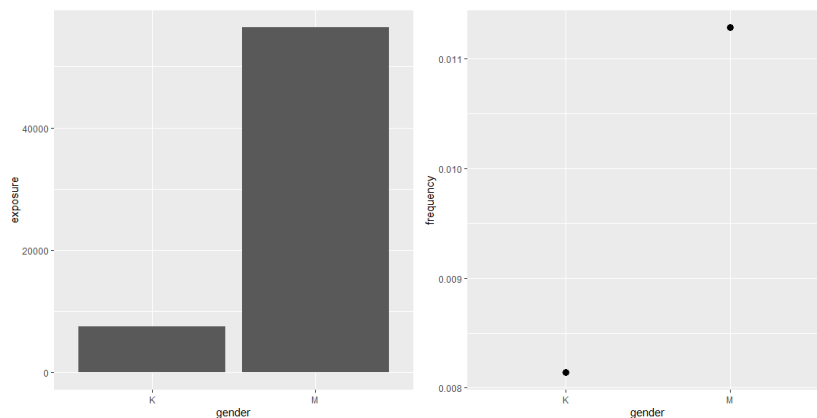
In Figure 4.4, we observe that exposure pattern in MC class is similar to what was seen in the histogram. There is a very slight positive relationship between vehicle power and claim frequency, therefore, the higher the vehicle power the higher the chance of having a claim. However, given the size of the standard error (the light-grey area in the plot) relative to the slope, the relationship does not seem to be strong.

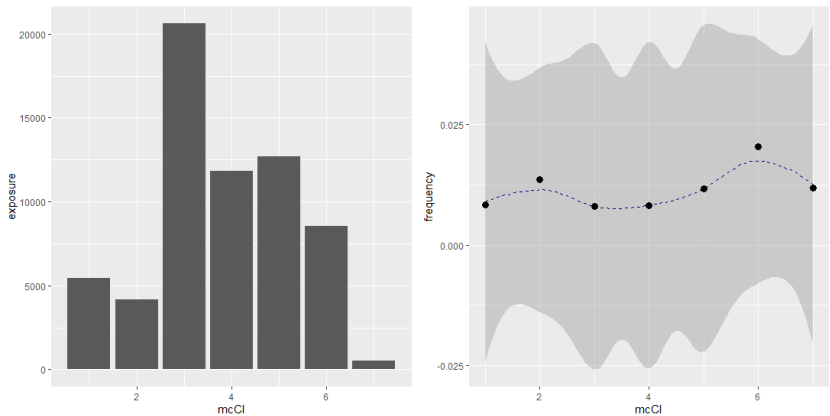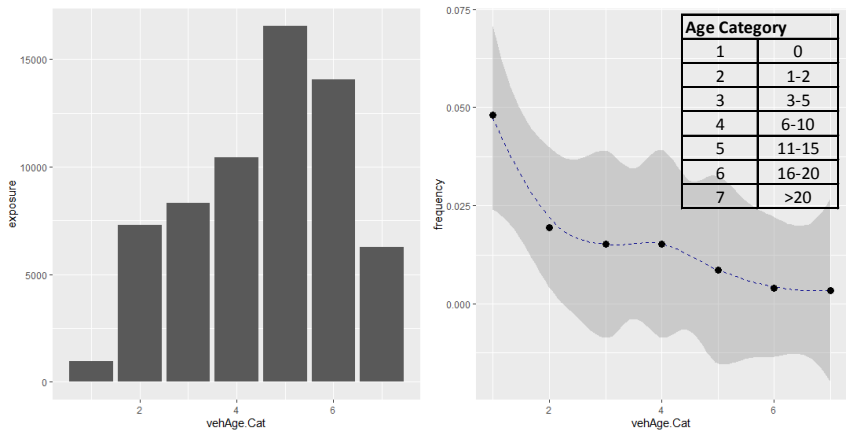*Figure 4.4: Exposure and claim frequency by MC class*

Figure 4.5 shows the relationship of the duration and claim frequency by different vehicle age groups. There is a noticeable negative relationship between claim frequency and vehicle age. This may be due to newer cars being driven more frequently than other vehicles and/or being subject to a particular type of claim such as theft.

Figure 4.5: Exposure and claim frequency by vehicle age group

| Age Category	
1	0
2	1-2
3	3-5
4	6-10
5	11-15
6	16-20
7	>20

In Figure 4.6, looking at the plots for bonus class, we can see that exposure concentrates in the highest class, 7. Unfortunately, there is no straightforward relationship with claim frequency.

Figure 4.6: Exposure and claim frequency by bonus class
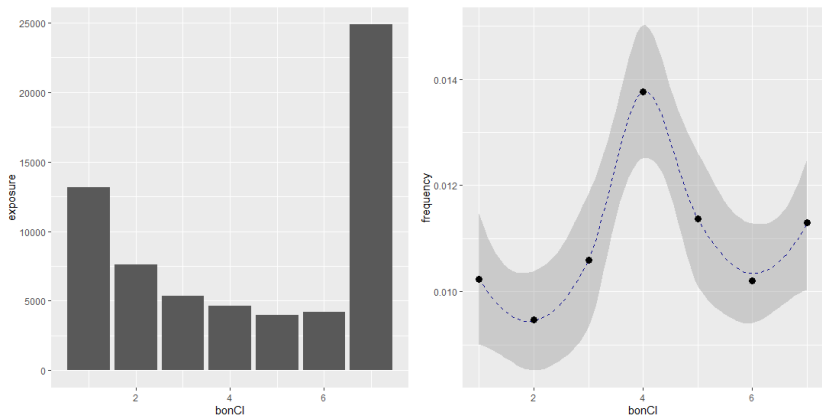
Finally, we look at the correlation plots for all the variables and see if we find any notable relationships between the features. From Figure 4.7, no strong collinearity seems to be present so will be able to use them as explanatory variables for the forthcoming section on regression.

Figure 4.7: Correlation plot for all variables (claimFreq=claimNum/duration)
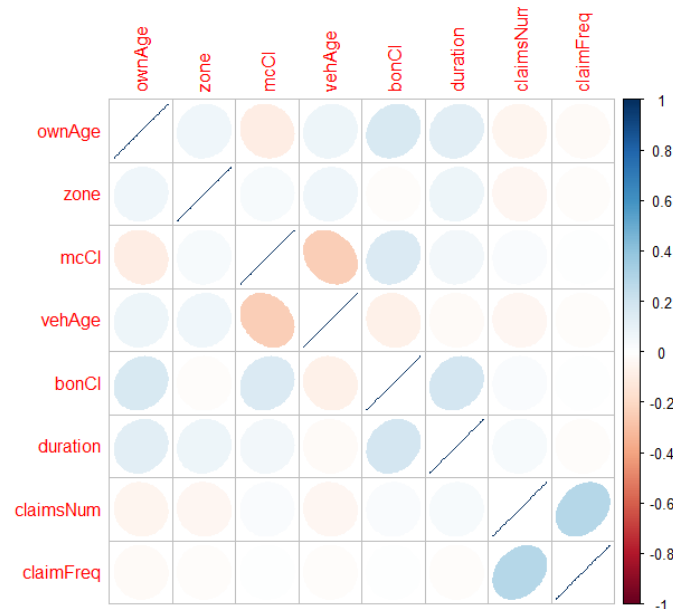
Final processing of data

Before proceeding to data modelling, we will factorise zone and MC class as well as re-level all factor variables with the reference group chosen based on majority class. The result is shown in Figure 4.8 below. The first class that appears in the level is the reference group (e.g. Male for gender and 4 for zone).

Figure 4.8: Final data structure (R command output)

```
'data.frame':   64548 obs. of  8 variables:
 $ ownAge   : num  16 16 16 16 16 16 16 16 16 16 ...
 $ gender   : Factor w/ 2 levels "M","K": 1 1 2 2 2 2 2 1 1 1 ...
 $ zone     : Factor w/ 7 levels "4","1","2","3",..: 2 4 4 1 3 4 1 1 3 1 ...
 $ mcCl     : Factor w/ 7 levels "3","1","2","4",..: 4 6 1 2 2 1 1 4 1 3 ...
 $ vehAge   : int  12 9 18 25 26 8 6 20 16 17 ...
 $ bonCl    : Factor w/ 7 levels "7","1","2","3",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ duration : num  0.175 1.044 0.455 0.173 0.181 ...
 $ claimsNum: int  0 0 0 0 0 0 0 0 0 0 ...
```

DATA MODELLING

Having done EDA, we will move on to fitting different models to the data in order to find the best model that can predict claim counts. We will fit ten different models in total using Generalised Linear Model (GLM), Neural Network (NN), Regression Tree (RT) and Clustering. While the criteria for goodness-of-fit will be judged based on in-sample error, model performance will be based on out-of-sample error as we are interested in the ability for a model to predict the response variable. Therefore, we will randomly select 70% of the observations for training and the remaining 30% for testing predictability.

<u>Generalised Linear Model (GLM)</u>

GLM is widely used in the general insurance industry in solving actuarial problems. This traditional method will be fit first to be benchmarked against other machine learning methods. Since we are looking at claim counts, we will conduct a Poisson regression whereby duration will be offset in order to estimate the claim frequency.

When we fit the training data with all explanatory variables, the summary output is as per Figure 5.1. Since we treat gender, zone and MC class as factors, we get in total 21 predictors. All the variables have a certain degree of significance in explaining the variability in claim frequency except for zones 5 to 7, MC class 1/4/7 and most bonus classes. Nevertheless, we keep all the variables in and compute the in-sample and out-of-sample error to be 9.197364 and 8.982713 respectively (scaled by 100 for better visibility). Note that we are using Poisson deviances and we call this model fit GLM_1.

Figure 5.1: Regression output (R command)

```
Call:
glm(formula = claimsNum ~ gender + zone + mcCl + bonCl + ownAge +
    vehAge, family = poisson(), data = train, offset = log(duration))

Deviance Residuals:
    Min      1Q   Median      3Q     Max
-0.8891  -0.1529  -0.0959  -0.0577   4.6858

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.399784   0.232968 -10.301  < 2e-16 ***
genderK     -0.505057   0.167681  -3.012 0.002595 **
zone1        1.385520   0.125555  11.035  < 2e-16 ***
zone2        0.922709   0.126958   7.268 3.65e-13 ***
zone3        0.492532   0.135601   3.632 0.000281 ***
zone5       -0.329327   0.417736  -0.788 0.430486
zone6       -0.013628   0.301856  -0.045 0.963991
zone7       -0.032504   1.004019  -0.032 0.974174
mcCl1        0.348760   0.207574   1.680 0.092924 .
mcCl2        0.542251   0.189024   2.869 0.004122 **
mcCl4        0.134319   0.152046   0.883 0.377014
mcCl5        0.473231   0.139413   3.394 0.000688 ***
mcCl6        0.978990   0.133988   7.307 2.74e-13 ***
mcCl7        0.091058   0.422167   0.216 0.829228
bonCl1      -0.215495   0.133569  -1.613 0.106667
bonCl2      -0.424462   0.175956  -2.412 0.015851 *
bonCl3      -0.122697   0.172193  -0.713 0.476121
bonCl4       0.029555   0.167787   0.176 0.860179
bonCl5      -0.204469   0.192578  -1.062 0.288351
bonCl6      -0.224472   0.194129  -1.156 0.247556
ownAge      -0.051431   0.004131 -12.449  < 2e-16 ***
vehAge      -0.082820   0.007864 -10.532  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 4751.8  on 45182  degrees of freedom
Residual deviance: 4155.6  on 45161  degrees of freedom
AIC: 5153.8

Number of Fisher Scoring iterations: 8
```

Now we look at an alternative GLM model using LASSO regularisation to see if we get any improvement in the predictability by shrinking the coefficients for some predictors. In order to proceed, we will first need to define what for shrinkage parameter (lambda) to use and we choose based on 10-fold cross validation (CV) on the training dataset. In Figure 5.2, the dark orange line indicates the value of lambda that produces the lowest level of cross validation error which reduces the number of predictors to 17 (call this GLM_2). We can instead choose to adopt 1-standard error rule and choose to shrink further at the dark blue line, but this seems to over-penalise the model as it reduces the number of predictors to only 5. The resulting in-sample and out-of-sample error using the GLM_2 is 9.208842 and 8.971021. The fit is slightly worse against the training data set which is expected as we are synthetically shrinking the coefficients, but its predictability has slightly improved.

Figure 5.2: 10-fold CV result (left = coefficients, right = deviance)
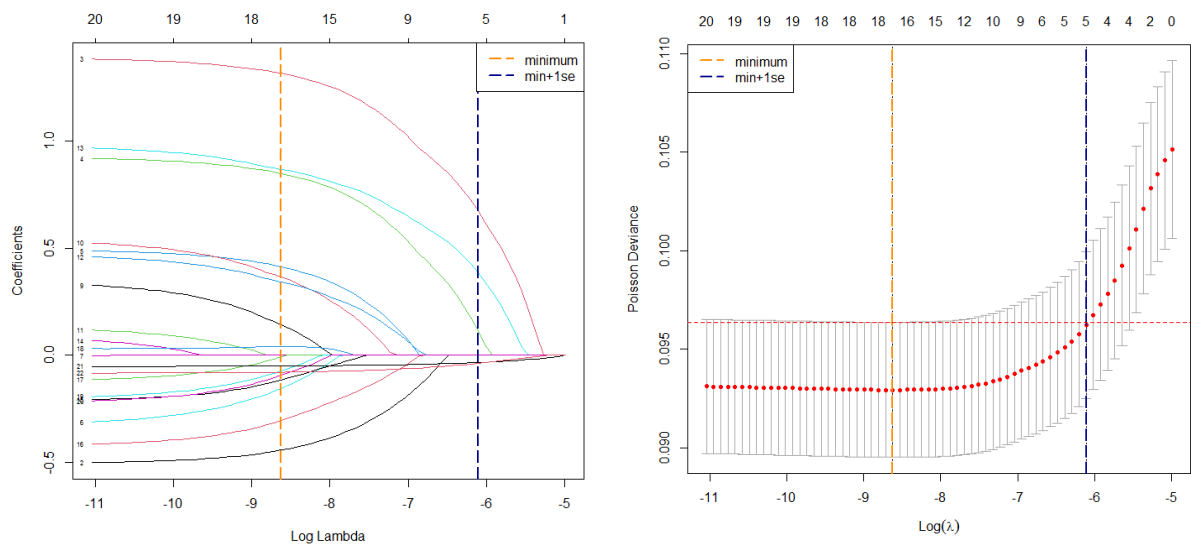
We have also attempted at conducting forward and backward selection, however, the result was not superior to either model so we do not show this in the report. Nevertheless, the R code is included in the appendix for reference.

Figure 5.3: GLM performance (errors based on Poisson deviance and scaled by 100)

Model	Description	In-sample error	Out-of-sample error
GLM_1	Poisson regression using all predictors	9.197364	8.982713
GLM_2	LASSO regularisation - shrinkage with min CV error	9.208842	8.971021

Neural Network (NN)

Next, we look at Neural Network (NN), a popular supervised machine learning method which is a non-linear generalisation of the GLM framework. In other words, non-linear transformations of linear combination of covariates called neurons are utilised in predicting claim counts. This is a parametric model.

The model requires all covariates to live on the same scale, therefore, we first normalise numerical variables namely ownAge and vehAge on a [0,1] scale. Categorical covariates need no treatment as they are already dummy coded.

Furthermore, there is a need to define the level and number of neurons to be formed. While this is an arbitrary choice that a modeller needs to make, there needs to be sufficiently large for the model to produce meaningful results. Given we have 21 original inputs, we present two models, the first being a shallow NN (i.e. single level) with 50 neurons and another being a deep NN on two levels with 50 and 25 neurons in each. This is the final selection after testing many different combinations. We call the former model NN_1 and the latter NN_2.

Finally, we choose to use hyperbolic tangent activation function to link the covariates to neurons and the number of epochs in optimising the fit through gradient descent method is chosen based on cross validation. Selection is based on judgement where we stop the iteration where the fit is improved sufficiently without overfitting (i.e. around where validation error starts increasing).

Below figures summarise the output of each model. As per Figure 6.1 and 6.3, the number of estimated parameters is 1,153 for NN_1 and 2,403 for NN_2, which are significantly more than what was required under GLM and suggests its structure complexity.

Figure 6.1: Shallow NN with 50 neurons on 21 original covariates (NN_1)

Layer (type)	Output Shape	Param #	Connected to
Design (InputLayer)	[(None, 21)]	0	
Layer1 (Dense)	(None, 50)	1100	Design[0][0]
Network (Dense)	(None, 1)	51	Layer1[0][0]
LogVol (InputLayer)	[(None, 1)]	0	
Add (Add)	(None, 1)	0	Network[0][0] LogVol[0][0]
Response (Dense)	(None, 1)	2	Add[0][0]

```
Total params: 1,153
Trainable params: 1,151
Non-trainable params: 2
```
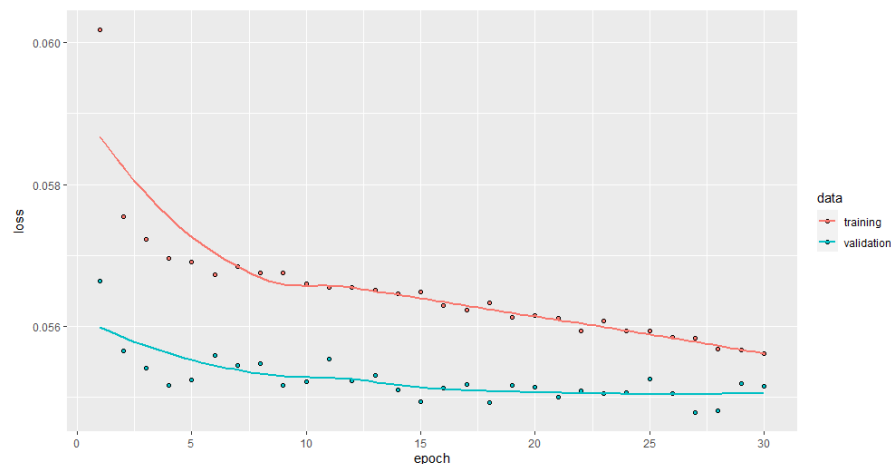
Figure 6.2: Deep NN with two levels; 50 and 25 neurons on 21 original covariates (NN$_2$)

Layer (type)	Output Shape	Param #	Connected to
Design (InputLayer)	[(None, 21)]	0	
Layer1 (Dense)	(None, 50)	1100	Design[0][0]
Layer2 (Dense)	(None, 25)	1275	Layer1[0][0]
Network (Dense)	(None, 1)	26	Layer2[0][0]
LogVol (InputLayer)	[(None, 1)]	0	
Add (Add)	(None, 1)	0	Network[0][0] LogVol[0][0]
Response (Dense)	(None, 1)	2	Add[0][0]

Total params: 2,403
Trainable params: 2,401
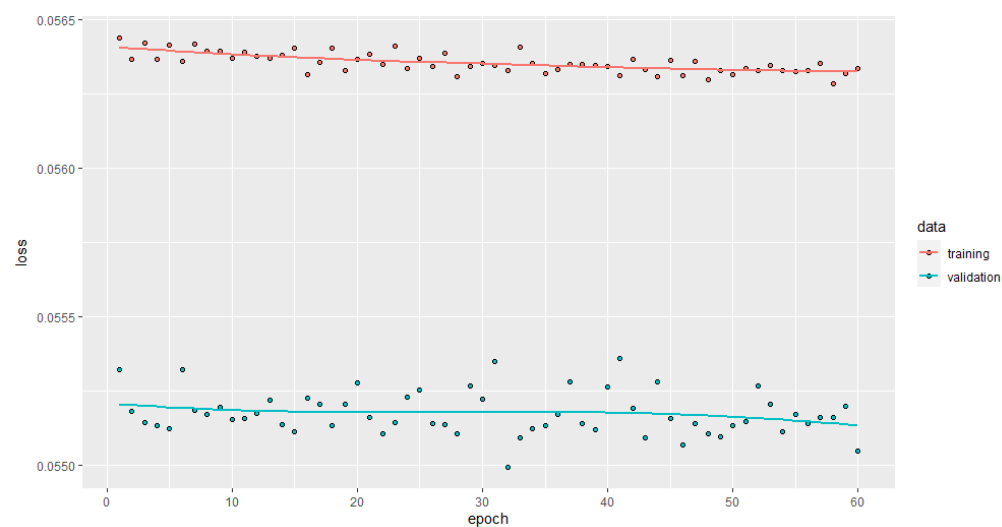Non-trainable params: 2
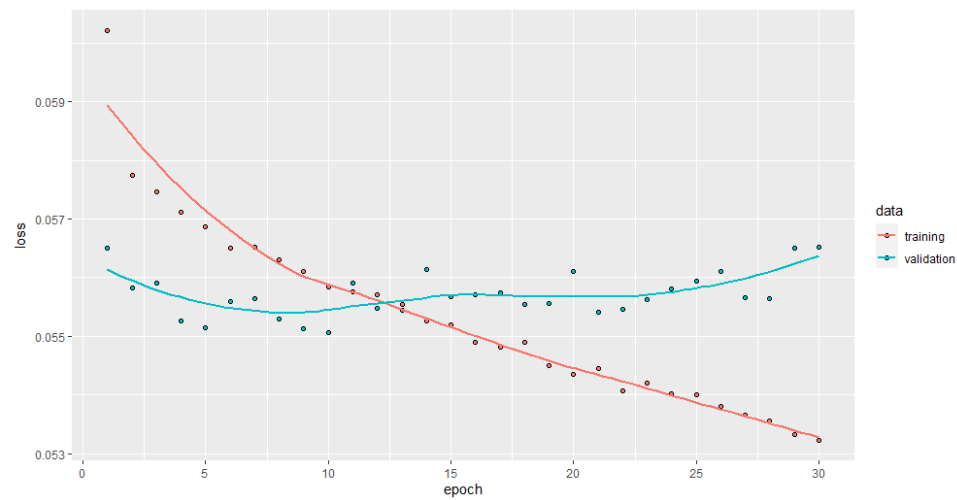
To evaluate these models, we use in-sample and out-of-sample errors based on Poisson deviances for consistency and comparability. They are 9.02851 and 8.982198 for NN$_1$ and 8.466588 and 9.256198 for NN$_2$ respectively. This shows that the shallow NN is superior in terms of predictability. The in-sample error suggests the deeper NN to have a better fit but this may just be as a result of over-fitting.

In addition to the two, we fit a combined GLM-NN model. The concept is to use the GLM output as an initial estimate and boost the result with NN. Taking the better fit from the GLM, we use GLM$_2$ (the regularised model) as our starting point and "boost" the fit with a shallow NN with the same functional form as NN$_1$. The result is shown in Figure 6.3 below.

Figure 6.3: Combined model with GLM$_2$ and NN$_1$

Layer (type)	Output Shape	Param #	Connected to
Design (InputLayer)	[(None, 21)]	0	
Layer1 (Dense)	(None, 50)	1100	Design[0][0]
Network (Dense)	(None, 1)	51	Layer1[0][0]
LogVol (InputLayer)	[(None, 1)]	0	
Add (Add)	(None, 1)	0	Network[0][0] LogVol[0][0]
Response (Dense)	(None, 1)	2	Add[0][0]

Total params: 1,153
Trainable params: 1,151
Non-trainable params: 2

The in-sample and out-of-sample errors for NN_3 are 9.192447 and 8.96024 respectively. The fit is not as good as the other NN models but provides the best predictability. In fact, NN_3 is superior to both GLM models in terms of fit and predictability, therefore, it provides evidence that Neural Network can empower traditional methods. The model performance of the NN models are summarised in Figure 6.4.

Figure 6.4: NN model performance (errors based on Poisson deviance and scaled by 100)

| Model	Description	In-sample error	Out-of-sample error
NN_1	Shallow NN with 50 neurons	9.028510	8.982198
NN_2	Deep NN - two levels with 50 and 25 neurons	8.466588	9.256198
NN_3	Boosted GLM_2 with NN_1	9.192447	8.960240

Regression Tree (RT)

Thirdly, we test using decision trees which is another supervised machine learning method but non-parametric in nature. The optimisation is based on finding the best homogeneous partitions of the feature space. Similar to methods discussed so far, this will be achieved through minimising Poisson deviance statistics.

We will explore three models first being a simple tree, another being boosted trees and finally combining GLM and tree boosting. They will be labelled as RT_1, RT_2 and RT_3 respectively.

Using all covariates available, we adopt the standardised binary split (SBS) algorithm to efficiently partition the feature space to sufficient granularity which produces a large tree. This large tree is then subjected to 10-fold cross validation (CV) in order to find the optimal number of terminal nodes/leaves. We then prune the tree to arrive at our best fit. The result of the CV is shown in Figure 7.1 and if we take the 1-standard error rule, the optimal tree is with four leaves.

Figure 7.1: Cross validation result (left = original-scale, right = log-scale)
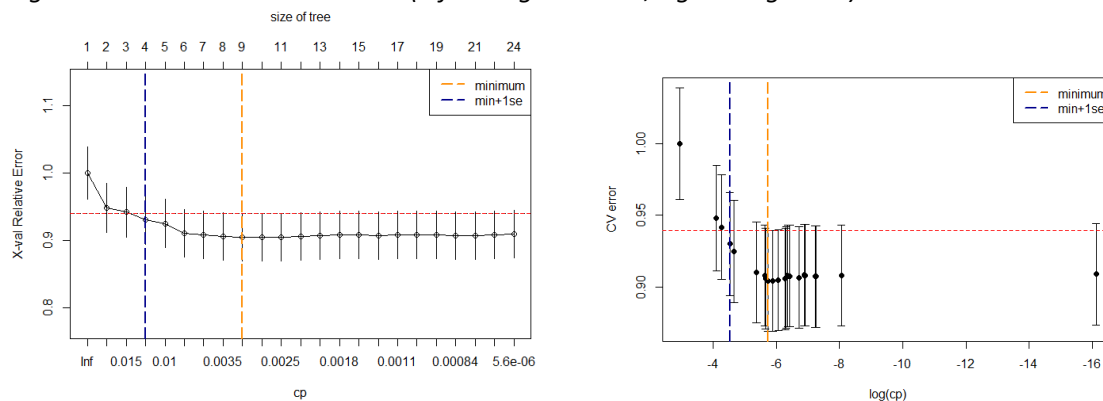
This simplifies the tree diagram as per Figure 7.2 and this shows that owner age, vehicle age and zone are important differentiators. It firstly suggests that owners younger than 31 has higher risk and further if they reside in zone 1, the policyholder is categorised to have the highest risk with claim frequency rate of 7.1%. This is significantly higher compared to the initial estimate without any partitioning of 1.1%. In contrast, those with the lowest risk are owners above 31 who rides a motorcycle older than or equal to 13 years. This is consistent to our initial EDA where we observed older owners and older vehicles having lower claim frequency.

Figure 7.2: Tree diagram after pruning with 1 s.e. rule (RT_1)
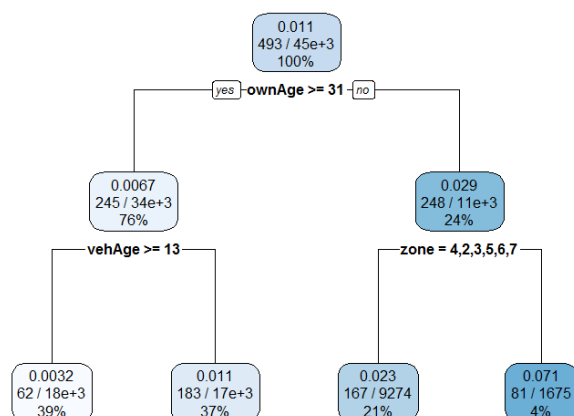
The in-sample and out-of-sample error from RT_1 fit is 9.643599 and 9.34932 and can see that this is not quite as good as the GLMs nor NN models we have fit so far. While we conducted CV, the number of leaves presented here are relatively small and it suggests that a simple regression tree is not the best at fitting the dataset. However, it still provides useful and easy to understand insight on the importance of covariates and optimal partitions.

Another useful decision tree method is boosting where small trees (called weak learners) are fit iteratively on the residuals from preceding trees. There are no definitive rules in choosing the size of trees, but for the purpose of this project, we choose to fit 80 trees each with 1 layer depth (i.e. two leaves). In addition, we assume a shrinkage parameter of 0.5 in order to ensure each tree is weak enough. Figure 7.3 shows the fit for RT_2. We observe that vehicle and owner age continue to be the most influential factor in partitioning. The other two plots show the in-sample and out-of-sample errors for each iteration (i.e. additional tree) and can see the number of trees we fit is sufficient to get a good fit without under nor over-fitting.

Figure 7.3: Summary output (RT₂)
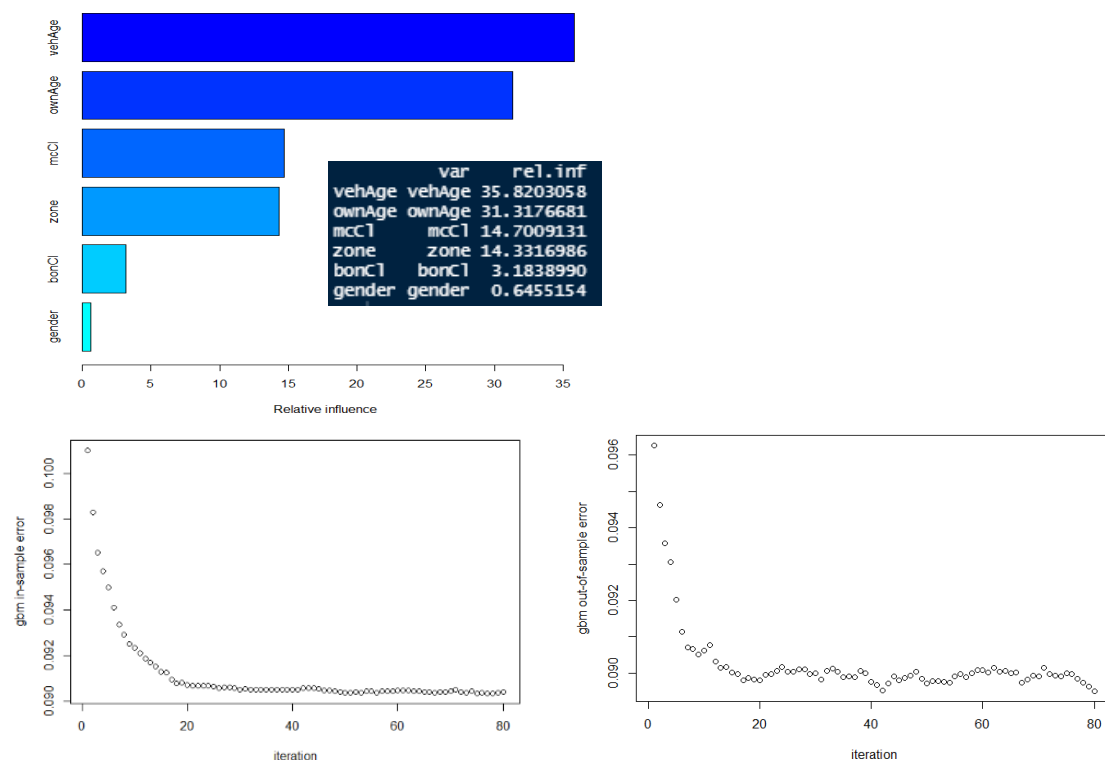
We then move to the third model (RT$_3$) which combines RT boosting with GLM in the hope to improve the fit. We use the GLM$_2$ as our initial starting point and fit the same functional form as RT$_2$. Again, we show the summary output in Figure 7.4 and observe that the covariate influence is very much the same with RT$_2$ but with much less on zone, bonus class and gender as such is already taken into account within the GLM fit. While the iterational reduction in in-sample and out-of-sample errors look more jitterish, there is no particular sign of under or overfitting and we conclude that fitting 80 trees for boosting is sufficient.

Figure 7.4: Summary output (RT$_3$)
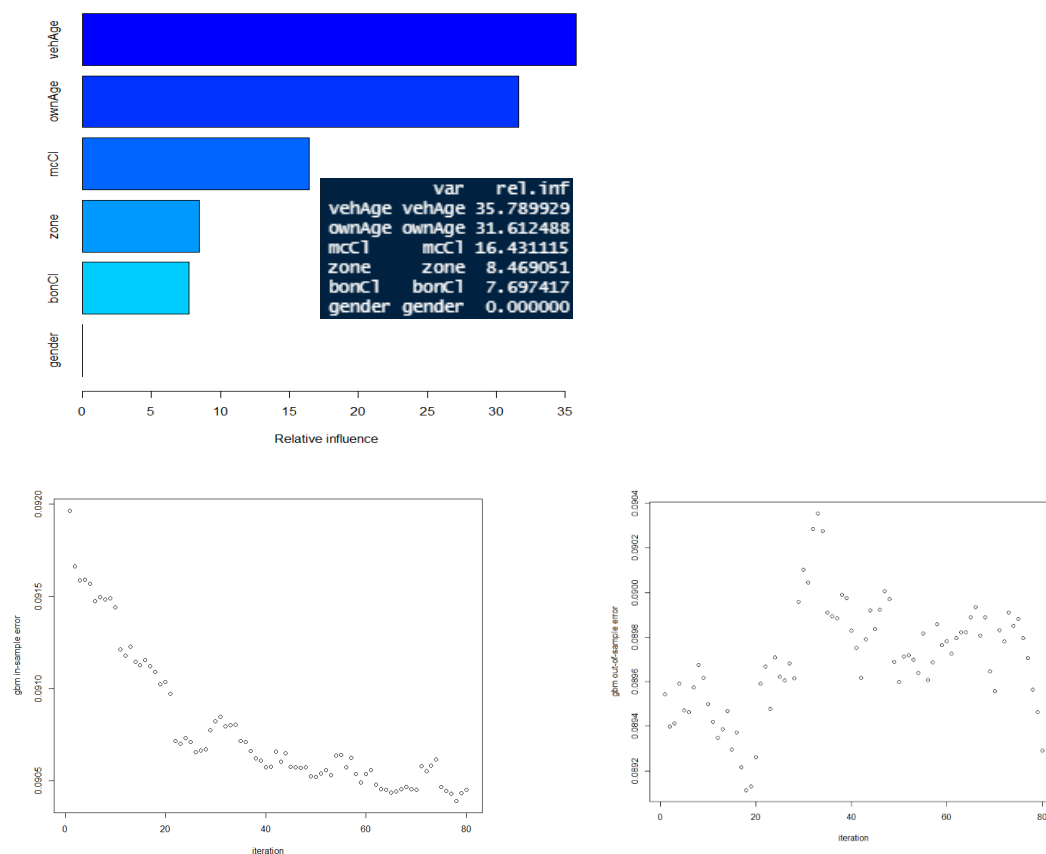
Finally, we show in Figure 7.5 the in-sample and out-of-sample errors for the three RT models. Notably, the boosted trees are better both in terms of fit and prediction power over a single pruned tree as well as to any of the GLM fits. While the GLM boosted tree is slightly worse than the pure boosted tree in terms of fit, its predictability is the best out of the three. However, we note the observed variability we see in the out-of-sample error with different choices of number of trees being fit as per the bottom right plot in Figure 7.4.

Figure 7.5: RT model performance (errors based on Poisson deviance and scaled by 100)

Model	Description	In-sample error	Out-of-sample error
RT$_1$	A pruned tree with 4 leaves	9.643599	9.349320
RT$_2$	Boosted tree - 100 trees with single depth	9.038952	8.950128
RT$_3$	Boosted GLM$_2$ with RT$_2$	9.045287	8.929113

K-means clustering and GLM

Finally, we look at utilising a clustering technique which is an unsupervised machine learning method that looks at grouping observations based on their similarity. Unlike other methods discussed so far, it does not optimise for any response variable, hence, the application here is to consider the generated cluster labels as a new covariate to the feature space which is then used to fit GLM.

Before we run the clustering algorithm, we convert categorical variables to dummy numerical values such as gender so that distance can be computed. For simplification, we have chosen to use K-means clustering with Euclidian distance measure. The optimal number of clusters are chosen where within group sum of squared (WSS) errors are minimised. This is analogous to using a gap statistic and choosing where the gap is maximised[5]. This is shown in Figure 8.1 below and from the gap statistics, the number of clusters is chosen to be two. This suggests the dataset is already fairly homogeneous as there is only one significant partition. We then use this cluster label as a new covariate to be included in the GLM fits. We will call the new non-regularised model GLM_3 and the LASSO regularised model GLM_4.

Figure 8.1: Gap statistic over number of clusters
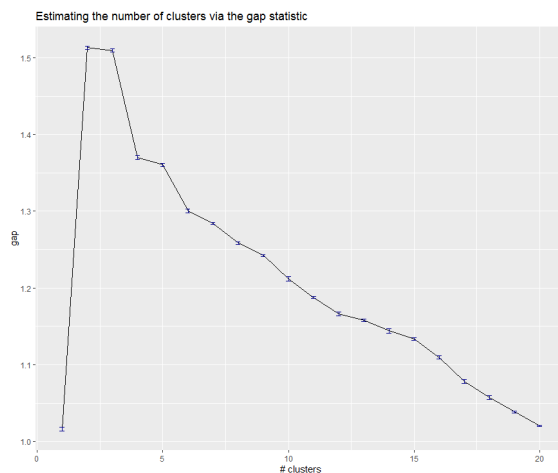
To avoid overcrowding the report, we will just mention that the cluster labels are not a significant covariate in explaining the variability of claims frequency given all other original covariates according to the p-value. In fact, if we proceed with including the cluster label in the model, we get in-sample and out-of-sample errors as in Figure 8.2 where the out-of-sample error increases compared to the original GLM models. Therefore, we conclude that using K-means clustering labels for this data does not add extra value which is not unexpected since there are only limited number of features available to start off with and they are all well-defined.

Figure 8.2: Clustering model performance (errors based on Poisson deviance and scaled by 100)

Model	Description	In-sample error	Out-of-sample error
GLM_3	GLM_1 with cluster labels	9.191648	8.983568
GLM_4	GLM_2 with cluster labels	9.206854	8.971790

---

[5] Gap statistic function is adopted from <https://github.com/echen/gap-statistic>

Summary

Below shows the summary of in-sample and out-of-sample errors of all the models we have tested in this project for easy comparison. Using GLM model fits as a benchmark, we observed mixed results both in terms of fit and predictability of the model performance of machine learning methods. It is clear that machine learning methods do perform well especially when they are combined with the benchmark GLM models (NN_3 and RT_3).

Figure 9: Summary of all tested models
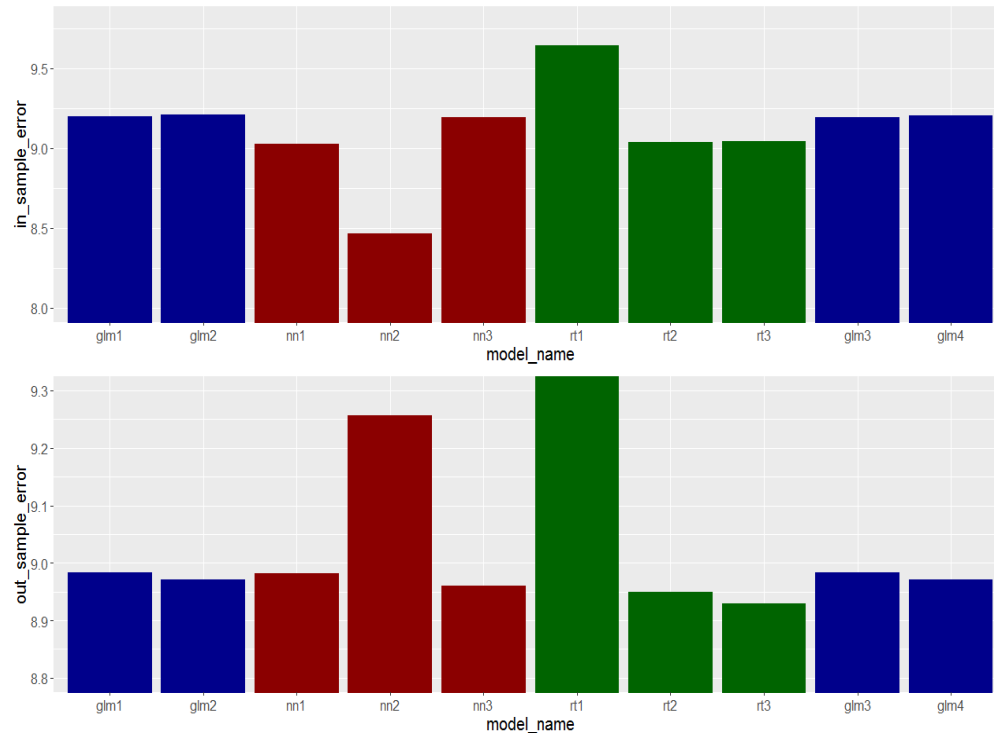
Model	Description	In-sample error	Out-of-sample error
GLM_1	Poisson regression using all predictors	9.197364	8.982713
GLM_2	LASSO regularisation - shrinkage with min CV error	9.208842	8.971021
NN_1	Shallow NN with 50 neurons	9.028510	8.982198
NN_2	Deep NN - two levels with 50 and 25 neurons	8.466588	9.256198
NN_3	Boosted GLM_2 with NN_1	9.192447	8.960240
RT_1	A pruned tree with 4 leaves	9.643599	9.349320
RT_2	Boosted tree - 100 trees with single depth	9.038952	8.950128
RT_3	Boosted GLM_2 with RT_2	9.045287	8.929113
GLM_3	GLM_1 with cluster labels	9.191648	8.983568
GLM_4	GLM_2 with cluster labels	9.206854	8.971790

CONCLUSION

In this report, we explored the dataset on motorcycle casco insurance and tested various predictive models with the response variable being the number of claims. Starting with the classical actuarial model, GLM, we looked at how modern machine learning methods perform against this benchmark (GLM_1 and GLM_2).

The analysis found that Neural Network models (NN_1 and NN_2) outperformed the benchmark both in terms of fit and prediction power. For Regression Tree models, the boosted tree (RT_2) outperformed similarly but not the simplest tree with four leaves (RT_1).

We have also explored combinations whereby the GLM benchmark was boosted by each method. Improvement in results over pure use of each method was observed and conclude that the boosted GLM regression tree (RT_3) to have the best fit and predictability of all.

Furthermore, an attempt was made to make use of an unsupervised learning method in improving the fit, however, we were not able to obtain further improvement from the benchmark (GLM_3 and GLM_4).

While machine learning methods have performed well above the benchmark in most instances, it is reliant on domain knowledge and certain assumptions such as the number of neurons and epochs for NN models and number trees for boosting RT models which are based on judgement. In addition, the interpretation of fitted parameters in machine learning methods are difficult to understand than standard GLMs.

We also need to cast light on the dataset itself as they are quite old and will be worth exploring data that is more up to date to obtain better predictive powers. Furthermore, the number of features we used were eight in total, however, there should be more variables that can be collected now-adays and utilised for modelling such as mileage, driving purpose, private/commercial and even telematic data to incorporate driving behaviour. The project was limited to the use of publicly available data as the author did not have access to further information but is worth considering for future work.

As a conclusive remark, the project was able to showcase that modern machine learning techniques can add value to traditional methods that are widely accepted and employed within the actuarial industry. An important note is that this report is case specific to a motorcycle insurance portfolio from a particular Swedish company and the usefulness of those more modern techniques may vary case-by-case and modellers will always need to make sure that the chosen model is fit for purpose before being used.

Appendix: R-code

Exploratory Data Analysis

```r
#################################################
### Exploratory Data Analysis ###
#################################################

###### Set up ########
rm(list=ls())
setwd(choose.dir()) #choose appropriate working directory
getwd()

library(tidyverse)
library(dplyr)
library(psych)

### Import and check structure of original data

#install.packages("insuranceData")
library(insuranceData)
data("dataOhlsson")

?dataOhlsson

dim(dataOhlsson)
str(dataOhlsson)
head(dataOhlsson)
summary(dataOhlsson)
describe(dataOhlsson)

# $ agarald (owner age): int  0 4 5 5 6 9 9 9 10 10 ...
# $ kon     (gender (K=Female, M=Male)): Factor w/ 2 levels "K","M": 2 2 1 1 1 1 1 2 2 2 ...
# $ zon     (geographic region): int  1 3 3 4 2 3 4 4 2 4 ...
# $ mcklass (EV ratio class - Vehicle power feature) : int  4 6 3 1 1 3 3 4 3 2 ...
# $ fordald (vehicle age) : int  12 9 18 25 26 8 6 20 16 17 ...
# $ bonuskl (bonus class - higher the better) : int  1 1 1 1 1 1 1 1 1 ...
# $ duration (policy exposure in year unit): num  0.175 0 0.455 0.173 0.181 ...
# $ antskad (number of claims) : int  0 0 0 0 0 0 0 0 0 0 ...
# $ skadkost (cost of claims) : int  0 0 0 0 0 0 0 0 0 0 ...

### Change the headers to comprehensible English names
data <- dataOhlsson
names(data) <- c("ownAge", "gender","zone","mcCl","vehAge","bonCl","duration","claimsNum","claimsAmt")
str(data)

### Check for N/As
apply(data, 2, function(x) any(is.na(x)))
apply(data, 2, function(x) any(is.null(x)))

library(simputation)
library(naniar)
miss_var_summary(data) #double-check

### Check for Uniqueness
## owner age
  unique(data$ownAge)[order(unique(data$ownAge))]
  prop.table(table(data$ownAge))

  ownAge.summ <- data %>% filter(ownAge < 16) %>%
    summarise(exposure = sum(duration), claims = sum(claimsNum), frequency = sum(claimsNum)/sum(duration))
  ownAge.summ$exposure/sum(data$duration)
  ownAge.summ$claims/sum(data$claimsNum)
  # those below 16 will be floored at 16 (later)

## vehicle Age
  unique(data$vehAge)[order(unique(data$vehAge))]
  prop.table(table(data$vehAge))

  #It is unnatural to have cars beyond 50 years but possible so have left it as it is (not influential)
  vehAge.summ <- data %>% filter(vehAge > 20) %>%
    summarise(exposure = sum(duration), claims = sum(claimsNum), frequency = sum(claimsNum)/sum(duration))
  vehAge.summ$exposure/sum(data$duration) #low in exposure
  vehAge.summ$claims/sum(data$claimsNum) # too high to be omitted

## gender
unique(data$gender)
prop.table(table(data$gender))

## zone
unique(data$zone)[order(unique(data$zone))]
prop.table(table(data$zone))

## mcClass
unique(data$mcCl)[order(unique(data$mcCl))]
prop.table(table(data$mcCl))

## Bonus Class
unique(data$bonCl)[order(unique(data$bonCl))]
prop.table(table(data$bonCl))
```

```r
## duration
range(data$duration)
data %>% ggplot(aes(duration)) + geom_histogram(binwidth = 1)
duration.summ <- data %>% filter(duration > 5) %>%
  summarise(policies = n(), exposure = sum(duration),
            claims = sum(claimsNum), frequency = sum(claimsNum)/sum(duration))
duration.summ$exposure/sum(data$duration)
duration.summ$claims/sum(data$claimsNum)
# -> alternatively this could be due to typo in which case 32 means 3.2.
# But without domain knowledge, difficult to tell so cap.

duration.summ2 <- data %>% filter(duration <= 0) %>%
  summarise(policies = n(),exposure = sum(duration), claims = sum(claimsNum))
duration.summ2$claims/sum(data$claimsNum)
duration.summ2$policies/nrow(data)
# non-trivial number of rows --> to avoid data loss, we will fill this with the mean duration

### Claim number
unique(data$claimsNum)[order(unique(data$claimsNum))]
prop.table(table(data$claimsNum))

##############################################
###  data cleansing 1 ###
dur.mod.mean <- data %>% filter(pmin(duration,5) > 0) %>% summarise(mean = mean(duration))
data1 <- data[,c(-9)] %>% mutate(duration = as.numeric(ifelse(duration <=0, dur.mod.mean, duration)))
data1$duration <- pmin(data1$duration,5)
data1$ownAge <- pmax(data1$ownAge,16)

head(data1 %>% arrange(duration))
str(data1)
summary(data1)
##############################################

### Histograms (for cont) and Barplot (for discrete)
ownAge.hist <- data1 %>% ggplot(aes(ownAge)) +
  geom_histogram(binwidth = 5,fill = "darkblue") +
  labs(x = "Age" , y = "Count", title = "Owner Age")
gender.hist <- data1 %>% ggplot(aes(gender)) +
  geom_bar(fill = "darkblue") + labs(x = "Gender" , y = "Count", title = "Gender (K=Female, M=Male)")
zone.hist <- data1 %>% ggplot(aes(zone)) +
  geom_bar(fill = "darkblue") + labs(x = "Zone" , y = "Count", title = "Zone")
mcCl.hist <- data1 %>% ggplot(aes(mcCl)) +
  geom_bar(fill = "darkblue") + labs(x = "Class" , y = "Count", title = "MC Class")
vehAge.hist <- data1 %>% ggplot(aes(vehAge)) +
  geom_histogram(binwidth = 5, fill = "darkblue") +
  labs(x = "Age" , y = "Count", title = "Vehicle Age") + xlim(0,50)
bonCl.hist <- data1 %>% ggplot(aes(bonCl)) +
  geom_bar(fill = "darkblue") + labs(x = "Class" , y = "Count", title = "Bonus Class")

duration.hist <- data1 %>% ggplot(aes(duration)) +
  geom_histogram(binwidth = 1, fill = "darkblue") +
  labs(x = "Exposure" , y = "Count", title = "Duration Histogram")
duration.box <- data1 %>% ggplot(aes(duration)) +
  geom_boxplot(fill = "white") + labs(x = "Exposure" , y = "Count", title = "Duration Boxplot")

claimsNum.hist <- data1 %>% ggplot(aes(claimsNum)) +
  geom_histogram(binwidth = 1,fill = "darkblue") +
  labs(x = "Exposure" , y = "Count", title = "Claim Number Histogram")
claimsNum.box <- data1 %>% ggplot(aes(claimsNum)) +
  geom_boxplot(fill = "white")  + labs(x = "Exposure" , y = "Count", title = "Claim Number Boxplot")

data1.freq <- sum(data1$claimsNum)/sum(data1$duration)
data1.freq

library(gridExtra)
grid.arrange(ownAge.hist,gender.hist,zone.hist,mcCl.hist,vehAge.hist,bonCl.hist)
grid.arrange(duration.hist,duration.box,claimsNum.hist,claimsNum.box)

### Frequency summary and charts
library(gridExtra)
library(ggrepel)
library(ggthemes)

## by owner age bands
ownAge.summary <- data1  %>%
  mutate(ownAge.Cat = case_when(ownAge <=25 ~ 1,
                                ownAge >25 & ownAge <=35 ~ 2,
                                ownAge >35 & ownAge <=45 ~ 3,
                                ownAge >45 & ownAge <=55 ~ 4,
                                ownAge >55 & ownAge <=65 ~ 5,
                                ownAge >65 & ownAge <=75 ~ 6,
                                ownAge >75 & ownAge <=85 ~ 7,
                                ownAge >85 ~ 8)) %>%
  group_by(ownAge.Cat) %>%
  summarise(exposure = sum(duration), claims = sum(claimsNum), frequency = sum(claimsNum)/sum(duration))
ownAge.exp <- ownAge.summary %>% ggplot(aes(ownAge.Cat,exposure)) + geom_col()
ownAge.num <- ownAge.summary %>% ggplot(aes(ownAge.Cat,claims)) + geom_col()
ownAge.freq <- ownAge.summary %>%
  ggplot(aes(ownAge.Cat,frequency)) +
  geom_smooth(color = "darkblue", linetype = "dashed", size = 0.5, se = TRUE) +  geom_point(size=3)
grid.arrange(ownAge.exp,ownAge.freq, ncol=2)

## by zone
zone.summary <- data1 %>% group_by(zone) %>%
  summarise(exposure = sum(duration), claims = sum(claimsNum), frequency = sum(claimsNum)/sum(duration))
zone.exp <- zone.summary %>% ggplot(aes(zone,exposure)) + geom_col()
zone.num <- zone.summary %>% ggplot(aes(zone,claims)) + geom_col()
zone.freq <- zone.summary %>% ggplot(aes(zone,frequency)) +
  geom_smooth(color = "darkblue", linetype = "dashed", size = 0.5, se = TRUE) + geom_point(size=3)
grid.arrange(zone.exp,zone.freq, ncol=2)
# -> negative correlation with frequency
```

```r
zone.summary2 <- data1 %>%
  mutate(ownAge.Cat = case_when(ownAge <=25 ~ 1,
                                ownAge >25 & ownAge <=35 ~ 2,
                                ownAge >35 & ownAge <=45 ~ 3,
                                ownAge >45 & ownAge <=55 ~ 4,
                                ownAge >55 & ownAge <=65 ~ 5,
                                ownAge >65 & ownAge <=75 ~ 6,
                                ownAge >75 & ownAge <=85 ~ 7,
                                ownAge >85 ~ 8)) %>%
  group_by(zone, ownAge.Cat) %>%
  summarise(exposure2 = sum(duration), claims2 = sum(claimsNum), frequency2 = sum(claimsNum)/sum(duration))

zone.ownAge.summ <- left_join(zone.summary2, zone.summary, key = zone) %>%
  mutate(relexp = exposure2/exposure, relclaims = claims2/claims, relfreq = claims2/exposure)
zone.ownAge.summ %>%
  ggplot(aes(zone, relclaims, fill = ownAge.Cat)) + geom_bar(stat = "identity")

zone.summary3 <- data1 %>%
  mutate(vehAge.Cat = case_when(vehAge == 0 ~ 1,
                                vehAge >0 & vehAge <=2 ~ 2,
                                vehAge >2 & vehAge <=5 ~ 3,
                                vehAge >5 & vehAge <=10 ~ 4,
                                vehAge >10 & vehAge <=15 ~ 5,
                                vehAge >15 & vehAge <=20 ~ 6,
                                vehAge >20 ~ 7)) %>%
  group_by(zone, vehAge.Cat) %>%
  summarise(exposure2 = sum(duration), claims2 = sum(claimsNum), frequency2 = sum(claimsNum)/sum(duration))

zone.vehAge.summ <- left_join(zone.summary3, zone.summary, key = zone) %>%
  mutate(relexp = exposure2/exposure, relclaims = claims2/claims, relfreq = claims2/exposure)
zone.vehAge.summ %>% ggplot(aes(zone, relclaims, fill = vehAge.Cat)) + geom_bar(stat = "identity")
# -> checked for correlation with owner age group but nothing significant was found.
# This is not included in the report as the corr plot summarises it better

## by gender
gender.summary <- data1 %>% group_by(gender) %>%
  summarise(exposure = sum(duration), claims = sum(claimsNum), frequency = sum(claimsNum)/sum(duration))
gender.exp <- gender.summary %>% ggplot(aes(gender,exposure)) + geom_col()
gender.num <- gender.summary %>% ggplot(aes(gender,claims)) + geom_col()
gender.freq <- gender.summary %>% ggplot(aes(gender,frequency)) + geom_point(size=3)

grid.arrange(gender.exp,gender.freq, ncol=2)


data1 %>% ggplot(aes(zone, ownAge)) + geom_jitter()

## by MC Class (equivalent to the power of the vehicle)
mcCl.summary <- data1 %>% group_by(mcCl) %>%
  summarise(exposure = sum(duration), claims = sum(claimsNum), frequency = sum(claimsNum)/sum(duration))
mcCl.exp <- mcCl.summary %>% ggplot(aes(mcCl,exposure)) + geom_col()
mcCl.num <- mcCl.summary %>% ggplot(aes(mcCl,claims)) + geom_col()
mcCl.freq <- mcCl.summary %>% ggplot(aes(mcCl,frequency)) +
  geom_smooth(color = "darkblue", linetype = "dashed", size = 0.5, se = TRUE) + geom_point(size=3)

grid.arrange(mcCl.exp,mcCl.freq, ncol=2)


## by vehicle age bands
vehAge.summary <- data1 %>%
  mutate(vehAge.Cat = case_when(vehAge == 0 ~ 1,
                                vehAge >0 & vehAge <=2 ~ 2,
                                vehAge >2 & vehAge <=5 ~ 3,
                                vehAge >5 & vehAge <=10 ~ 4,
                                vehAge >10 & vehAge <=15 ~ 5,
                                vehAge >15 & vehAge <=20 ~ 6,
                                vehAge >20 ~ 7)) %>%
  group_by(vehAge.Cat) %>%
  summarise(exposure = sum(duration), claims = sum(claimsNum), frequency = sum(claimsNum)/sum(duration))
vehAge.exp <- vehAge.summary %>% ggplot(aes(vehAge.Cat,exposure)) + geom_col()
vehAge.num <- vehAge.summary %>% ggplot(aes(vehAge.Cat,claims)) + geom_col()
vehAge.freq <- vehAge.summary %>% ggplot(aes(vehAge.Cat,frequency)) +
  geom_smooth(color = "darkblue", linetype = "dashed", size = 0.5, se = TRUE) + geom_point(size=3)
grid.arrange(vehAge.exp,vehAge.freq, ncol=2)

## by Bonus Class
bonCl.summary <- data1 %>%
  group_by(bonCl) %>%
  summarise(exposure = sum(duration), claims = sum(claimsNum), frequency = sum(claimsNum)/sum(duration))
bonCl.exp <- bonCl.summary %>% ggplot(aes(bonCl,exposure)) + geom_col()
bonCl.num <- bonCl.summary %>% ggplot(aes(bonCl,claims)) + geom_col()
bonCl.freq <- bonCl.summary %>% ggplot(aes(bonCl,frequency)) +
  geom_smooth(color = "darkblue", linetype = "dashed", size = 0.5, se = TRUE) + geom_point(size=3)

grid.arrange(bonCl.exp,bonCl.freq, ncol=2)


#compute correlations
library(corrplot)
str(data1)
data1.cor <- data1 %>% mutate(claimFreq = claimsNum/duration)
corrplot(cor(data1.cor[,c(-2)]), method = "ellipse") #excludes Gender (non-numeric)
```

```r
###################################################
### Cleanse the data for use in data modelling ###
###################################################
# re-levelling with reference class (for all)

###  data cleansing 2 (for modelling)
data2 <- data1


data2[,"gender"] <- relevel(data2[,"gender"], ref = "M")

data2$mcCl <- as.factor(data2$mcCl)
data2[,"mcCl"] <- relevel(data2[,"mcCl"], ref = "3")


data2$bonCl <- as.factor(data2$bonCl)
data2[,"bonCl"] <- relevel(data2[,"bonCl"], ref = "7")


data2$zone <- as.factor(data2$zone)
data2[,"zone"] <- relevel(data2[,"zone"], ref = "4")

str(data2)
summary(data2)
head(data2)
```

Generalised Linear Modelling

```r
##########################################################
### Data Modelling: Generalised Linear Model (GLM) ###
##########################################################
set.seed(123)
trainsampleindex <- sample(c(1:nrow(data2)),0.7*nrow(data2),replace=FALSE)
#random sample index of size 70% of total rowcounts without replacement
train <- data2[trainsampleindex,]
test <- data2[-trainsampleindex,]

###################################################
### GLM using all predictors
modelfit.glm1 <- glm(claimsNum~gender+zone+mcCl+bonCl+ownAge+vehAge,data=train, family=poisson(),offset=log(duration))

summary(modelfit.glm1) # 64548 data and 21 predictors (1 for beta_0)

train$fit.glm1 <- predict(modelfit.glm1, newdata=train, type="response")
test$fit.glm1 <- predict(modelfit.glm1, newdata=test, type="response")

in_sample_error.glm1 <-
    2*(sum(log((train$claimsNum/train$fit.glm1)^train$claimsNum))-sum(train$claimsNum)+sum(train$fit.glm1))/nrow(train)
out_sample_error.glm1 <-
    2*(sum(log((test$claimsNum/test$fit.glm1)^test$claimsNum))-sum(test$claimsNum)+sum(test$fit.glm1))/nrow(test)
cat(" GLM1 in-sample error:", in_sample_error.glm1*100, "\n","GLM1 out-of-sample error:", out_sample_error.glm1*100)


###################################################
### Regularisation (LASSO)
library(glmnet)
Xdata.train <- model.matrix(~gender+zone+mcCl+bonCl+ownAge+vehAge, data=train)
Xdata.test <- model.matrix(~gender+zone+mcCl+bonCl+ownAge+vehAge, data=test)

modelfit.glm2 <- glmnet(x=Xdata.train, y=train$claimsNum, family = "poisson", alpha = 1, offset = log(train$duration))
cv.glm2 <- cv.glmnet(Xdata.train, train$claimsNum, type.measure ="deviance",
                     family = "poisson", alpha = 1, offset = log(train$duration), nfolds = 10)

plot(cv.glm2)
abline(v=log(cv.glm2$lambda.min), lty = 5,lwd = 2, col = "darkorange")
abline(v=log(cv.glm2$lambda.1se), lty = 5,lwd = 2, col = "darkblue")
abline(h=cv.glm2$cvup[which(cv.glm2$lambda == cv.glm2$lambda.min)], lty = 2,lwd = 1, col = "red")
legend("topleft", legend = c("minimum", "min+1se"), col = c("darkorange", "darkblue"), lty = c(5,5), lwd = c(2,2))

plot(modelfit.glm2, xvar="lambda", label="TRUE")
abline(v=log(cv.glm2$lambda.min), lty = 5,lwd = 2, col = "darkorange")
abline(v=log(cv.glm2$lambda.1se), lty = 5,lwd = 2, col = "darkblue")
legend("topright", legend = c("minimum", "min+1se"), col = c("darkorange", "darkblue"), lty = c(5,5), lwd = c(2,2))

# here choose to just fit based on lambda that gives lowest CV error (i.e. the orange line = 24 parameters)
train$fit.glm2 <-
    predict(modelfit.glm2, newx = Xdata.train, newoffset = log(train$duration), type="response", s = cv.glm2$lambda.min)
test$fit.glm2 <-
    predict(modelfit.glm2, newx = Xdata.test, newoffset = log(test$duration), type="response", s = cv.glm2$lambda.min)

in_sample_error.glm2 <-
    2*(sum(log((train$claimsNum/train$fit.glm2)^train$claimsNum))-sum(train$claimsNum)+sum(train$fit.glm2))/nrow(train)
out_sample_error.glm2 <-
    2*(sum(log((test$claimsNum/test$fit.glm2)^test$claimsNum))-sum(test$claimsNum)+sum(test$fit.glm2))/nrow(test)
cat(" GLM2 in-sample error:", in_sample_error.glm2*100, "\n","GLM2 out-of-sample error:", out_sample_error.glm2*100)
###################################################
### forward and backward selection (not used in report)
#need to manually create dummy variables for step-wise regression
library(dummies)
gender.dum <- data.frame(dummy(train2$gender)[,-1])
names(gender.dum) <- c("genderK")
zone.dum <- data.frame(dummy(train2$zone)[,-1])
mcCl.dum <- data.frame(dummy(train2$mcCl)[,-1])
bonCl.dum <- data.frame(dummy(train2$bonCl)[,-1])
train2 <- train
train2 <- cbind(train2[,c(1,5,7,8)],gender.dum,zone.dum,mcCl.dum,bonCl.dum)
str(train2)
modelfit.glm.full <- glm(claimsNum~.-duration,data=train2, family=poisson(),offset=log(duration))
#check that the fit is the same as glm1
summary(modelfit.glm.full)
summary(modelfit.glm1)

#step-wise regression
library(MASS)
step <- stepAIC(modelfit.glm.full, direction = "both", trace = TRUE)
summary(step)

test2 <- test
gender.dum <- data.frame(dummy(test2$gender)[,-1])
names(gender.dum) <- c("genderK")
zone.dum <- data.frame(dummy(test2$zone)[,-1])
mcCl.dum <- data.frame(dummy(test2$mcCl)[,-1])
bonCl.dum <- data.frame(dummy(test2$bonCl)[,-1])
test2 <- cbind(test2[,c(1,5,7,8)],gender.dum,zone.dum,mcCl.dum,bonCl.dum)
str(test2)

train$fit.glmstep <- predict(step, train2, type="response")
test$fit.glmstep <- predict(step, test2, type="response")

in_sample_error.glmstep <-
    2*(sum(log((train$claimsNum/train$fit.glmstep)^train$claimsNum))
       -sum(train$claimsNum)+sum(train$fit.glmstep))/nrow(train)
out_sample_error.glmstep <-
    2*(sum(log((test$claimsNum/test$fit.glmstep)^test$claimsNum))
       -sum(test$claimsNum)+sum(test$fit.glmstep))/nrow(test)
cat(" GLM Step in-sample error:", in_sample_error.glmstep*100,
    "\n","GLM Step out-of-sample error:", out_sample_error.glmstep*100)
```

Neural Network

```r
###########################################################
### Data Modelling: Neural Network (NN) ###
###########################################################
### Use the same training and testing splits from the original
### normalise (categorical values already live in the [0,1] space so no need)
ownAge.min <- min(data2$ownAge)
ownAge.max <- max(data2$ownAge)
vehAge.min <- min(data2$vehAge)
vehAge.max <- max(data2$vehAge)


feature.train <- cbind(scale(train$ownAge,center=ownAge.min,scale=ownAge.max-ownAge.min),
                       scale(train$vehAge,center=vehAge.min,scale=vehAge.max-vehAge.min))
Xdata.train <- list(as.matrix(cbind(model.matrix(~gender+zone+mcCl+bonCl, data=train)[,-1],feature.train)),
                    as.matrix(log(train$duration)))

feature.test<-cbind(scale(test$ownAge,center=ownAge.min,scale=ownAge.max-ownAge.min),
                    scale(test$vehAge,center=vehAge.min,scale=vehAge.max-vehAge.min))
Xdata.test <- list(as.matrix(cbind(model.matrix(~gender+zone+mcCl+bonCl, data=test)[,-1],feature.test)),
                   as.matrix(log(test$duration)))

Ydata.train <- as.matrix(train$claimsNum)
Ydata.test <- as.matrix(test$claimsNum)

library(keras)
library(tensorflow)
library(tfdatasets)
library(tidyverse)

###################################################
### Shallow NN with 50 neurons (with 21 original covariates)
# build model
set_random_seed(123)
q0 <- 21 # no of covariates
q1 <- 50 # no of neurons
lambda0 <- sum(train$claimsNum)/sum(train$duration)

Design <- layer_input(shape=c(q0),dtype='float32', name='Design') #size of input
LogVol <- layer_input(shape=c(1),dtype='float32', name='LogVol') #for exposure/volume adj

Network = Design %>%
  layer_dense(units=q1, activation='tanh',name='Layer1') %>%
  layer_dense(units=1, activation='linear', name= 'Network',
              weights=list(array(0, dim=c(q1,1)),array(log(lambda0), dim=c(1))))

Response = list(Network,LogVol) %>%
  layer_add(name='Add') %>%
  layer_dense(units=1,activation=k_exp,name='Response',trainable = FALSE,
              weights = list(array(1,dim=c(1,1)),array(0,dim=c(1))))

# fit
modelfit.nn1 <- keras_model(inputs = c(Design, LogVol), outputs = c(Response))
modelfit.nn1 %>% compile(optimizer = optimizer_nadam(), loss = 'poisson')
summary(modelfit.nn1) #shows number of parameters to be estimated/trained


history.nn1 <- modelfit.nn1 %>%
  fit(Xdata.train, Ydata.train, epochs=30,batch_size=nrow(Xdata.train),view_metrics = TRUE,
                      validation_data = list(Xdata.test, Ydata.test), verbose=0)
plot(history.nn1)

train$fit.nn1  <- modelfit.nn1 %>% predict(Xdata.train)
test$fit.nn1 <- modelfit.nn1 %>% predict(Xdata.test)

# in-out sample error
in_sample_error.nn1 <-
  2*(sum(log((Ydata.train/train$fit.nn1)^Ydata.train))-sum(Ydata.train)+sum(train$fit.nn1))/nrow(Ydata.train)
out_sample_error.nn1 <-
  2*(sum(log((Ydata.test/test$fit.nn1)^Ydata.test))-sum(Ydata.test)+sum(test$fit.nn1))/nrow(Ydata.test)
cat(" NN1 in-sample error =", in_sample_error.nn1*100, "\n", "NN1 out-sample error =", out_sample_error.nn1*100)



###################################################
### Deep NN (2 layers with 50 and 30 neurons each)
feature.train <- cbind(scale(train$ownAge,center=ownAge.min,scale=ownAge.max-ownAge.min),
                       scale(train$vehAge,center=vehAge.min,scale=vehAge.max-vehAge.min))
Xdata.train <- list(as.matrix(cbind(model.matrix(~gender+zone+mcCl+bonCl, data=train)[,-1],feature.train)),
                    as.matrix(log(train$duration)))

feature.test<-cbind(scale(test$ownAge,center=ownAge.min,scale=ownAge.max-ownAge.min),
                    scale(test$vehAge,center=vehAge.min,scale=vehAge.max-vehAge.min))
Xdata.test <- list(as.matrix(cbind(model.matrix(~gender+zone+mcCl+bonCl, data=test)[,-1],feature.test)),
                   as.matrix(log(test$duration)))

Ydata.train <- as.matrix(train$claimsNum)
Ydata.test <- as.matrix(test$claimsNum)

# build model
set_random_seed(123)
q0 <- 21 # no of covariates
q1 <- 50 # no of neurons (1st layer)
q2 <- 25 # no of neurons (2nd layer)
lambda0 <- sum(train$claimsNum)/sum(train$duration)

Design <- layer_input(shape=c(q0),dtype='float32', name='Design') #size of input
LogVol <- layer_input(shape=c(1),dtype='float32', name='LogVol') #for exposure/volume adj
```

```r
Network = Design %>%
  layer_dense(units=q1, activation='tanh',name='Layer1') %>%
  layer_dense(units=q2, activation='tanh',name='Layer2') %>%
  layer_dense(units=1, activation='linear', name= 'Network',
              weights=list(array(0, dim=c(q2,1)),array(log(lambda0), dim=c(1))))

Response = list(Network,LogVol) %>%
  layer_add(name='Add') %>%
  layer_dense(units=1,activation=k_exp,name='Response',trainable = FALSE,
              weights = list(array(1,dim=c(1,1)),array(0,dim=c(1))))

# fit
modelfit.nn2 <- keras_model(inputs = c(Design, LogVol), outputs = c(Response))
modelfit.nn2 %>% compile(optimizer = optimizer_nadam(), loss = 'poisson')
summary(model) #shows number of parameters to be estimated/trained

history.nn2 <- modelfit.nn2 %>% fit(Xdata.train, Ydata.train, epochs=30,batch_size=nrow(Xdata.train),view_metrics = TRUE,
                          validation_data = list(Xdata.test, Ydata.test), verbose=0)
plot(history.nn2)

train$fit.nn2  <- modelfit.nn2 %>% predict(Xdata.train)
test$fit.nn2 <- modelfit.nn2 %>% predict(Xdata.test)

# in-out sample error
in_sample_error.nn2 <-
  2*(sum(log((Ydata.train/train$fit.nn2)^Ydata.train))-sum(Ydata.train)+sum(train$fit.nn2))/nrow(Ydata.train)
out_sample_error.nn2 <-
  2*(sum(log((Ydata.test/test$fit.nn2)^Ydata.test))-sum(Ydata.test)+sum(test$fit.nn2))/nrow(Ydata.test)
cat(" NN2 in-sample error =", in_sample_error.nn2*100, "\n", "NN2 out-sample error =", out_sample_error.nn2*100)


####################################################
### Using shallow NN with q = 50, use neural net boosted GLM
feature.train <- cbind(scale(train$ownAge,center=ownAge.min,scale=ownAge.max-ownAge.min),
                       scale(train$vehAge,center=vehAge.min,scale=vehAge.max-vehAge.min))
Xdata.train <- list(as.matrix(cbind(model.matrix(~gender+zone+mcCl+bonCl, data=train)[,-1],feature.train)),
                    as.matrix(log(train$fit.glm2)))


feature.test <- cbind(scale(test$ownAge,center=ownAge.min,scale=ownAge.max-ownAge.min),
                      scale(test$vehAge,center=vehAge.min,scale=vehAge.max-vehAge.min))
Xdata.test <- list(as.matrix(cbind(model.matrix(~gender+zone+mcCl+bonCl, data=test)[,-1],feature.test)),
                   as.matrix(log(test$fit.glm2)))

Ydata.train <- as.matrix(train$claimsNum)
Ydata.test <- as.matrix(test$claimsNum)

# build model
set_random_seed(123)
q0 <- 21 # no of covariates
q1 <- 50 # no of neurons
lambda0 <- sum(train$claimsNum)/sum(train$fit.glm2) #initial NN lambda estimate to be based on GLM2 prediction

Design <- layer_input(shape=c(q0),dtype='float32', name='Design') #size of input
LogVol <- layer_input(shape=c(1),dtype='float32', name='LogVol') #for exposure/volume adj


Network = Design %>%
  layer_dense(units=q1, activation='tanh',name='Layer1') %>%
  layer_dense(units=1, activation='linear', name= 'Network',
              weights=list(array(0, dim=c(q1,1)),array(log(lambda0), dim=c(1))))

Response = list(Network,LogVol) %>%
  layer_add(name='Add') %>%
  layer_dense(units=1,activation=k_exp,name='Response',trainable = FALSE,
              weights = list(array(1,dim=c(1,1)),array(0,dim=c(1))))

# fit
modelfit.nn3 <- keras_model(inputs = c(Design, LogVol), outputs = c(Response))
modelfit.nn3 %>% compile(optimizer = optimizer_nadam(), loss = 'poisson')
summary(modelfit.nn3) #shows number of parameters to be estimated/trained
history.nn3 <- modelfit.nn3 %>%
  fit(Xdata.train, Ydata.train, epochs=60,batch_size=nrow(Xdata.train),view_metrics = TRUE,
                          validation_data = list(Xdata.test, Ydata.test), verbose=0)
plot(history.nn3)

train$fit.nn3  <- modelfit.nn3 %>% predict(Xdata.train)
test$fit.nn3 <- modelfit.nn3 %>% predict(Xdata.test)

# in-out sample error
in_sample_error.nn3 <-
  2*(sum(log((Ydata.train/train$fit.nn3)^Ydata.train))-sum(Ydata.train)+sum(train$fit.nn3))/nrow(Ydata.train)
out_sample_error.nn3 <-
  2*(sum(log((Ydata.test/test$fit.nn3)^Ydata.test))-sum(Ydata.test)+sum(test$fit.nn3))/nrow(Ydata.test)
cat(" NN3 in-sample error =", in_sample_error.nn3*100, "\n", "NN3 out-sample error =", out_sample_error.nn3*100)

# -> not better than GLM (for out-sample error) but slightly better than the other NNs -- NN1 and NN2
# -> when GLM and NN is combined, it results in better fit and predictive power
```

Regression Tree

```
#######################################################
### Data Modelling: Decision Trees ###
#######################################################

#######################################################
### Regression Tree (using SBS) ###
# -> efficient tree with optimal cp (chosen as within 1 SE rule)

library(dplyr)
library(tidyverse)
library(readr)
library(tidyr)

require(rpart)
require(rpart.plot)
set.seed(123)
tree.fit <- rpart(cbind(duration, claimsNum) ~ gender+zone+mcCl+bonCl+ownAge+vehAge,
                  data = train,
                  method = "poisson",
                  parms = list(shrink=1),
                  control = rpart.control(xval = 10, minbucket = 1000, cp = 0.0000001))
# plot Relative error vs cp
plotcp(tree.fit, minline = T) #minline draws a horizontal line within 1SE of cp (min)
printcp(tree.fit)
summary(tree.fit)
rpart.plot(tree.fit)

min.cverr <- min(tree.fit$cp[,4])
min.cverr.sd <- tree.fit$cp[,5][which(tree.fit$cp[,4] == min.cverr)]

# create function called closest
# returns the closest point to x with the condition that x > y
# (e.g. x = min+1se, y = all CVs then outcome is the data closest to and within min+1se)
closest <- function (x, y){
  dist <- ifelse(x<y,1e9,x-y)
  min.dist.row <- which(dist == min(dist))
  min.dist.row
}

min.cp <- tree.fit$cp[closest(min.cverr, tree.fit$cp[,4]),1]
min.size <- tree.fit$cp[closest(min.cverr, tree.fit$cp[,4]),2] + 1

one.se.cp <- tree.fit$cp[closest(min.cverr+min.cverr.sd, tree.fit$cp[,4]),1]
one.se.size <- tree.fit$cp[closest(min.cverr+min.cverr.sd, tree.fit$cp[,4]),2] + 1

plotcp(tree.fit, minline = T)
abline(h = min.cverr + min.cverr.sd, lty = 2,lwd = 1, col = "red")
abline(v= min.size, lty = 5,lwd = 2, col = "darkorange")
abline(v = one.se.size, lty = 5,lwd = 2, col = "darkblue")
legend("topright", legend = c("minimum", "min+1se"), col = c("darkorange", "darkblue"), lty = c(5,5), lwd = c(2,2))
cat("optimal cp baed on 1-se rule:", one.se.cp)

# plot CV error vs log(cp) (better)
plot(log(tree.fit$cp[,1]),tree.fit$cp[,4],
     ylim=range(c(tree.fit$cp[,4]-tree.fit$cp[,5],tree.fit$cp[,4]+tree.fit$cp[,5])), #limit window to 1 se
     pch=19, xlab="log(cp)",ylab="CV error",
     main=" ", xlim = rev(range(log(tree.fit$cp[,1]))))
arrows(log(tree.fit$cp[,1]), tree.fit$cp[,4]-tree.fit$cp[,5], log(tree.fit$cp[,1]),
       tree.fit$cp[,4]+tree.fit$cp[,5], length=0.05, angle =90, code = 3)

abline(h = min.cverr + min.cverr.sd, lty = 2,lwd = 1, col = "red")
abline(v= log(min.cp), lty = 5,lwd = 2, col = "darkorange")
abline(v = log(one.se.cp), lty = 5,lwd = 2, col = "darkblue")
legend("topright", legend = c("minimum", "min+1se"), col = c("darkorange", "darkblue"), lty = c(5,5), lwd = c(2,2))

cat("optimal cp baed on 1-se rule:", one.se.cp)


# Prune the tree at the chosen cp parameter
tree.fit.pruned <- prune(tree.fit, cp = one.se.cp)
summary(tree.fit.pruned)
rpart.plot(tree.fit.pruned)

#in-out sample error
train$fit.rt1 <- train$duration * predict(tree.fit.pruned,  newdata = train)
test$fit.rt1 <- test$duration * predict(tree.fit.pruned, newdata = test)

in_sample_error.rt1 <-
  2*(sum(log((train$claimsNum/train$fit.rt1)^train$claimsNum))-sum(train$claimsNum)+sum(train$fit.rt1))/nrow(train)
out_sample_error.rt1 <-
  2*(sum(log((test$claimsNum/test$fit.rt1)^test$claimsNum))-sum(test$claimsNum)+sum(test$fit.rt1))/nrow(test)
cat(" RT1 in-sample error:", in_sample_error.rt1*100, "\n","RT1 out-of-sample error:", out_sample.error.rt1*100)


#######################################################
### Boosting Tree ###
# -> using 80 trees/weak learners with 1 layer-depth each (allowing for shrinking (with 0.5))
library(gbm)
oldw <- getOption("warn")
options(warn = -1) #to avoid printing warnings
set.seed(123)
K <- 80 #number of trees
idepth <- 1 #depth of each tree
fit.boost <- gbm(claimsNum~offset(log(duration))+gender+zone+mcCl+bonCl+ownAge+vehAge,
                 data=train,
                 distribution="poisson",
                 n.trees = K,
                 shrinkage = 0.5,
                 interaction.depth = idepth)
```

```r
summary(fit.boost)
train$fit.rt2 <- train$duration
test$fit.rt2 <- test$duration
in_sample_error.rt2 <- vector()
out_sample_error.rt2 <- vector()

for (k in 1:K){
  train$fit.rt2 <- train$duration*suppressWarnings(predict(fit.boost, newdata=train, n.trees=k, type="response"))
  test$fit.rt2 <- test$duration*suppressWarnings(predict(fit.boost, newdata=test, n.trees=k, type="response"))
  in_sample_error.rt2[k] <-
    2*(sum(log((train$claimsNum/train$fit.rt2)^train$claimsNum))-sum(train$claimsNum)+sum(train$fit.rt2))/nrow(train)
  out_sample_error.rt2[k] <-
    2*(sum(log((test$claimsNum/test$fit.rt2)^test$claimsNum))-sum(test$claimsNum)+sum(test$fit.rt2))/nrow(test)
} #suppresswarnings since it produces a unneeded warning that predicted values don't have exposure taken into account

plot(seq(1:K),in_sample_error.rt2,xlab="iteration",ylab="gbm in-sample error")
plot(seq(1:K),out_sample_error.rt2,xlab="iteration",ylab="gbm out-of-sample error")

options(warn = oldw) #to revert back the setting on warnings

out_sample_error.rt2[K]*100
cat(" RT2 in-sample error:", in_sample_error.rt2[K]*100, "\n","RT2 out-of-sample error:", out_sample_error.rt2[K]*100)
# -> produces slightly better in and out-of-sample error out of all (especially against GLM)

########################################################
### GLM Boosting with regression tree improvements ###

# -> using 80 trees/weak learners with 1 layer-depth each (allowing for shrinking (with 0.5))
library(gbm)
oldw <- getOption("warn")
options(warn = -1) #to avoid printing warnings
set.seed(123)
K <- 80 #number of trees
idepth <- 1 #depth of each tree
fit.boost2 <- gbm(claimsNum~offset(log(train$fit.glm2))+gender+zone+mcCl+bonCl+ownAge+vehAge,
                  #define offset as glm fitted value of number of claims
                  data=train,
                  distribution="poisson",
                  n.trees = K,
                  shrinkage = 0.5,
                  interaction.depth = idepth)
summary(fit.boost2)
train$fit.rt3 <- train$fit.glm2
test$fit.rt3 <- test$fit.glm2
in_sample_error.rt3 <- vector()
out_sample_error.rt3 <- vector()

for (k in 1:K){
  train$fit.rt3 <- train$fit.glm2*suppressWarnings(predict(fit.boost2, newdata=train, n.trees=k, type="response"))
  test$fit.rt3 <- test$fit.glm2*suppressWarnings(predict(fit.boost2, newdata=test, n.trees=k, type="response"))
  in_sample_error.rt3[k] <-
    2*(sum(log((train$claimsNum/train$fit.rt3)^train$claimsNum))-sum(train$claimsNum)+sum(train$fit.rt3))/nrow(train)
  out_sample_error.rt3[k] <-
    2*(sum(log((test$claimsNum/test$fit.rt3)^test$claimsNum))-sum(test$claimsNum)+sum(test$fit.rt3))/nrow(test)
}

plot(seq(1:K),in_sample_error.rt3,xlab="iteration",ylab="gbm in-sample error")
plot(seq(1:K),out_sample_error.rt3,xlab="iteration",ylab="gbm out-of-sample error")

options(warn = oldw) #to revert back the setting on warnings

out_sample_error.rt3[K]*100
cat(" RT3 in-sample error:", in_sample_error.rt3[K]*100, "\n","RT3 out-of-sample error:", out_sample_error.rt3[K]*100)

# -> produces slightly better in and out-of-sample error out of all
# probably given the improvement from boosting the results base on GLM non-parametrically
# However,the OOSE will vary quite a lot (i.e. not stable), so for stable results RT1 may be better
```

K-Means Clustering

```r
###################################################
### Modelling using K-Means Clustering ###
###################################################
### K-means clustering
# set up
#data1 stores original data after cleansing (but before re-leveling)
data3 <- data1
data3$gender <- as.numeric(data3$gender)
data3 <- data.frame(data3)
str(data3)
x.cluster <- as.matrix(data3[,!(names(data3) %in% c("claimsNum","duration"))])

# compute WSS for K-means clustering
wss <- (nrow(x.cluster)-1)*sum(apply(x.cluster,2,var))

set.seed(123)
for (i in 2:20) wss[i] <- sum(kmeans(x.cluster,i, nstart=10)$withinss) #store the wss for k=2-15
plot(1:20, wss, type="b", xlab = "Number of Clusters", ylab = "Within Groups Sum of Squared (WSS)")


#check with Gap Statistics
### Gap Statistic Method
library(plyr)
library(ggplot2)


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# build gap statistic (adopted from https://github.com/echen/gap-statistic)
gap_statistic <- function(data, min_num_clusters = 1, max_num_clusters = 20, num_reference_bootstraps = 10){
  num_clusters <-  min_num_clusters:max_num_clusters
  actual_dispersions <- maply(num_clusters, function(n) dispersion(data, n))
  ref_dispersions <- maply(num_clusters, function(n) reference_dispersion(data, n, num_reference_bootstraps))
  mean_ref_dispersions <- ref_dispersions[,1]
  stddev_ref_dispersions <- ref_dispersions[,2]
  gaps <- mean_ref_dispersions - actual_dispersions

  print(plot_gap_statistic(gaps, stddev_ref_dispersions, num_clusters))

  print(paste("The estimated numnber of clusters is ", num_clusters[which.max(gaps)],".", sep = ""))

  list(gaps = gaps, gap_stddevs = stddev_ref_dispersions)
}

# Plot the gaps along with error bars.
plot_gap_statistic = function(gaps, stddevs, num_clusters) {
  qplot(num_clusters, gaps, xlab = "# clusters", ylab = "gap", geom = "line",
      main = "Estimating the number of clusters via the gap statistic") +
    geom_errorbar(aes(num_clusters, ymin = gaps - stddevs, ymax = gaps + stddevs),
                size = 0.3, width = 0.2, colour = "darkblue")
}
# Calculate log(sum_i(within-cluster_i sum of squares around cluster_i mean)).
dispersion = function(data, num_clusters) {
  # R's k-means algorithm doesn't work when there is only one cluster.
  if (num_clusters == 1) {
    cluster_mean = aaply(data, 2, mean)
    distances_from_mean = aaply((data - cluster_mean)^2, 1, sum)
    log(sum(distances_from_mean))
  } else {
    # Run the k-means algorithm `nstart` times. Each run uses at most `iter.max` iterations.
    k = kmeans(data, centers = num_clusters, nstart = 10, iter.max = 50)
    # Take the sum, over each cluster, of the within-cluster sum of squares around the cluster mean.
    # Then take the log. This is `W_k` in Tw's notation.
    log(sum(k$withinss))
  }
}

# For an appropriate reference distribution (in this case, uniform points in the same range as `data`),
# simulate the mean and standard deviation of the dispersion.
reference_dispersion = function(data, num_clusters, num_reference_bootstraps) {
  dispersions = maply(1:num_reference_bootstraps, function(i) dispersion(generate_uniform_points(data), num_clusters))
  mean_dispersion = mean(dispersions)
  stddev_dispersion = sd(dispersions) / sqrt(1 + 1 / num_reference_bootstraps)
  # the extra factor accounts for simulation error
  c(mean_dispersion, stddev_dispersion)
}

# Generate uniform points within the range of `data`.
generate_uniform_points = function(data) {
  # Find the min/max values in each dimension, so that we can generate uniform numbers in these ranges.
  mins = aaply(data, 2, min)
  maxs = aaply(data, 2, max)

  num_datapoints = nrow(data)
  # For each dimension, generate `num_datapoints` points uniformly in the min/max range.
  uniform_pts = maply(1:length(mins), function(dim) runif(num_datapoints, min = mins[dim], max = maxs[dim]))
  uniform_pts = t(uniform_pts)
}
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# run gap statistic
gap_statistic(data=x.cluster, min_num_clusters = 1, max_num_clusters = 20, num_reference_bootstraps = 10)
# -> clarifies that the optimal number of clusters is two

###################################################
### Fit K-means with two clusters
fit.kmeans <- kmeans(x.cluster, 2, nstart=10, iter.max=50)
prop.table(table(fit.kmeans$cluster))
data3$cluster <- fit.kmeans$cluster
train$cluster <- data3$cluster[trainsampleindex]
test$cluster <- data3$cluster[-trainsampleindex]
```

```r
######################################################
### GLM (improved with cluster)
modelfit.glm3 <- glm(claimsNum~gender+zone+mcCl+bonCl+ownAge+vehAge+as.factor(cluster),
                     data=train, family=poisson(),offset=log(duration))
summary(modelfit.glm3)
train$fit.glm3 <- predict(modelfit.glm3, newdata=train, type="response")
test$fit.glm3 <- predict(modelfit.glm3, newdata=test, type="response")

in_sample_error.glm3 <-
  2*(sum(log((train$claimsNum/train$fit.glm3)^train$claimsNum))-sum(train$claimsNum)+sum(train$fit.glm3))/nrow(train)
out_sample_error.glm3 <-
  2*(sum(log((test$claimsNum/test$fit.glm3)^test$claimsNum))-sum(test$claimsNum)+sum(test$fit.glm3))/nrow(test)
cat(" GLM3 in-sample error:", in_sample_error.glm3*100, "\n","GLM3 out-of-sample error:", out_sample_error.glm3*100)

######################################################
### GLM LASSO (improved with cluster)
library(glmnet)
Xdata.train <- model.matrix(~gender+zone+mcCl+bonCl+ownAge+vehAge+as.factor(cluster), data=train)
Xdata.test <- model.matrix(~gender+zone+mcCl+bonCl+ownAge+vehAge+as.factor(cluster), data=test)

modelfit.glm4 <- glmnet(x=Xdata.train, y=train$claimsNum, family = "poisson", alpha = 1, offset = log(train$duration))
cv.glm4 <- cv.glmnet(Xdata.train, train$claimsNum, type.measure ="deviance",
                     family = "poisson", alpha = 1, offset = log(train$duration), nfolds = 10)
plot(cv.glm4)

train$fit.glm4 <- predict(modelfit.glm4, newx=Xdata.train, newoffset = log(train$duration),
                          type="response", s = cv.glm4$lambda.min)
test$fit.glm4 <- predict(modelfit.glm4, newx=Xdata.test, newoffset = log(test$duration),
                         type="response", s = cv.glm4$lambda.min)

in_sample_error.glm4 <-
  2*(sum(log((train$claimsNum/train$fit.glm4)^train$claimsNum))-sum(train$claimsNum)+sum(train$fit.glm4))/nrow(train)
out_sample_error.glm4 <-
  2*(sum(log((test$claimsNum/test$fit.glm4)^test$claimsNum))-sum(test$claimsNum)+sum(test$fit.glm4))/nrow(test)

cat(" GLM4 in-sample error:", in_sample_error.glm4*100, "\n","GLM4 out-of-sample error:", out_sample_error.glm4*100)

# -> no improvement against benchmark GLM fits
```

Final Summary Plot

```r
####################################################
## Final Summary ##
####################################################
##### Sample Error Plots ########
model_name <- c("glm1","glm2","nn1","nn2","nn3","rt1","rt2","rt3","glm3","glm4")
in_sample_error <- c(in_sample_error.glm1, in_sample_error.glm2,
                     in_sample_error.nn1, in_sample_error.nn2,in_sample_error.nn3,
                     in_sample_error.rt1, in_sample_error.rt2[K], in_sample_error.rt3[K],
                     in_sample_error.glm3, in_sample_error.glm4)*100
out_sample_error <- c(out_sample_error.glm1, out_sample_error.glm2,
                      out_sample_error.nn1,out_sample_error.nn2,out_sample_error.nn3,
                      out_sample_error.rt1, out_sample_error.rt2[K], out_sample_error.rt3[K],
                      out_sample_error.glm3, out_sample_error.glm4)*100
sample_error <- data.frame(cbind(model_name, in_sample_error, out_sample_error))
sample_error$in_sample_error <- as.double(in_sample_error)
sample_error$out_sample_error <- as.double(out_sample_error)
str(sample_error)

insampleplot <- sample_error %>%
  ggplot(aes(model_name, in_sample_error)) +
  geom_col(fill = c(rep("darkblue",2), rep("darkred",3), rep("darkgreen",3), rep("darkblue",2))) +
  scale_x_discrete(limit = c("glm1","glm2","nn1","nn2","nn3","rt1","rt2","rt3","glm3","glm4") ) +
  coord_cartesian(ylim = c(8.0, 9.8)) +
  theme(text = element_text(size = 15))

outsampleplot <- sample_error %>%
  ggplot(aes(model_name, out_sample_error)) +
  geom_col(fill = c(rep("darkblue",2), rep("darkred",3), rep("darkgreen",3), rep("darkblue",2))) +
  scale_x_discrete(limit = c("glm1","glm2","nn1","nn2","nn3","rt1","rt2","rt3","glm3","glm4") ) +
  coord_cartesian(ylim = c(8.8, 9.3)) +
  theme(text = element_text(size = 15))

grid.arrange(insampleplot, outsampleplot)
```