MA429 Coursework

Executive Summary

This report employs data from a motor insurance firm in Spain. The primary aims of this report are to build predictive models for insurance premiums and identify the key features that influence these. The models developed include Random Forest, Linear Regression, Gradient Boosting, and Neural Networks. The analyses revealed that key indicators of premiums include vehicle age, the driver's age, and years of driving experience. However, while the models provided valuable insights, their predictive power was not optimal, with the highest-performing model only having a 58% accuracy for classification and for regression a R^2 of 0.61. As the dataset is sourced from a single Spanish insurance company this may limit the model's ability to generalise to other countries and potentially other insurance companies. Quality of the data was also put into question which could be a key reason for the moderate results.

2025

Contents

1 Purpose of Report

This report draws on data from the European Actuarial Journal, which presents a rare dataset from a Spanish non-life motor insurance portfolio. The report aims to model from a customer perspective how firms determine insurance premiums and what is considered a low, medium or high premium. By applying predictive models, a way to estimate expected premium based on observable characteristics is provided along with a qualitative low, medium and high measure of how expensive the premium is. The models reveal which features have the greatest influence on premium pricing, helping to demystify the underwriting process.

2 Literature Source Discussion

Data from the European Actuarial Journal hereafter referred to as 'the dataset' was sourced from Segura-Gisbert, J. Lledó, J. Pavía, J.M. (2025) which itself does not undertake substantial analysis; rather, its primary objective is to make insurance data publicly available, given the scarcity of such datasets. With this in mind, the present study elects to pursue its own direction, building models and deriving insights independently from the original work.

3 Data Cleaning and Preparation

3.1 Data Cleaning:

The dataset is in CSV format containing some 105,555 rows and encompassing a wide array of 30 variables. The dataset was imported directly imported into Rstudio and it was observed that data cleaning was required due to the following features:

Date Formatting:

1/6th of the data variables were provided in a potentially incompatible date format likely leading to incorrect analysis between `dd-mm`. To avoid such a pitfall all observations with a date value were converted to `yyyy-mm-dd` format.

Policy lapse date:

This feature was missing around 66% of its values. As a lapse indicates premature contract termination, infilling using `Lapse`, `Date_last_renewal`, and `Date_next_renewal`, based on the hypotheses was attempted: if `Lapse` = 0 then `Date_lapse` = `Date_next_renewal`. If `Laspe` > 0 then `Date_lapse` = `Date_last_renewal`. This approach only matched 71% of known values leading to significant infilling requirement and associated noise introduction. As `Lapse` prediction was not the focus of the analysis it was removed.

Vehicle length:

`Type_risk` 1 (Motorbikes): Only 2 values were present recorded out of approximately 8500; undirected infilling would be very unreliable and type exclusion would result in substantial data loss. Infilling was performed using guidelines from the Institute of Highway Engineers (2005) and choosing a midpoint of 2.2m for `Type_risk 1 Length`.

`Type_risk` 2 (Vans): A relatively complete data field - likely because van length is a key factor affecting risk and would typically be reported. Imputation using the mean or median was considered for missing values. The distribution showed significant skew, making mean imputation unsuitable. `Length` had strong correlations with `Weight` (0.83), `value of vehicle` (0.73), and `cylinder capacity` (0.65) hence missing values were infilled using `Weight`, as they had the highest correlation.

Type_risk 3 (Passenger vehicles): An imputation scheme similar to that for Type_risk 2 was employed.

Type_risk 4 (Agricultural Vehicles): Agricultural had observations less than 1% and do not contribute to road vehicle analysis. These observations were removed.

Fuel:

Type_fuel has two values: 'P' for petrol and 'D' for diesel, converted to 0 and 1 respectively. For motorcycles, 24% of data was missing, while passenger vehicles and vans had complete fuel data. Side-research showed a complete dearth of diesel motorbikes as diesel engines are presumably too heavy and don't produce much power without forced induction (a turbocharger) making them unwieldy. The known Type_fuel values for motorbikes also confirmed this with an overwhelming majority being petrol making it reasonable to impute missing values with petrol.

3.2 Preparation:

Feature Engineering:

To increase the predictive power of the models several new features were added to create variables that may possess strong relationships with Premium. Modified features were designated with a _mod suffix to indicate dataset change/addition.

Type_fuel_mod and Length_mod: both of these were modified features as discussed above.

Age_mod: A new feature calculated as Date_last_renewal minus Date_birth to assess policyholder experience/maturity.

Vehicle_age_mod: An estimate of the vehicle age Date_last_renewal minus Year_matriculation. Note this does not take into account cars that have been re-registered that were already in a used condition e.g. imported cars.

Years_driving_mod: A new feature calculated as Date_last_renewal minus Date_driving_licence to measure driving experience.

Power_to_weight_mod: Calculated as Power / Weight. Vehicle performance potential is often expressed as power / weight where power is strongly related to tractive force. Increased or decreased tractive force divided by vehicular model mass produces a greater or lesser acceleration potential Newton (1687). This feature may offer insights into vehicle performance potential, especially when combined with Years_driving_mod.

Premium_category_mod: as a means to elucidate pre-existing risk analysis, the Premium feature values were in light of data skewing, subjected to binning with some 34,900 observations per three newly introduced categories (Table 1) within the premium_category_mod observations.

Premium Category		
£0 $\leq$	Low (0)	$<$ £260
£260 $\leq$	Medium (1)	$<$ £334
£334 $\leq$	High (2)	

Table 1: Insurance Premium Categories

4 Machine Learning Models

Three modelling approaches were undertaken in parallel as detailed in 4.1, 4.2 and 4.3. Rstudio and its associated libraries were utilised for all work in these sections. Random forests (4.1) was tasked to achieve premium classification where as regression (4.2) and neural networks (4.3) were used to estimate the premium value.

4.1 Random Forest - Premium Classification

4.1.1 Preparation

Loss Function

The accuracy loss function was chosen for this classification problem, predicting whether an individual should be assigned a `low`, `medium`, or `high` premium. This metric was selected because it directly reflects how often the model assigns the correct premium category, which is essential in efforts to derive a premium estimate.

Reasoning for selection

Random forests handle both classification and regression tasks well. They average the results of many decision trees, which improves accuracy and reduces overfitting. This makes them particularly useful for capturing complex, non-linear relationships between features like age, claims history, and vehicle type and therefore a strong choice for predicting insurance premiums.

Although individual predictions are less interpretable, random forests provide feature importance metrics that offer insight into which variables most influence premium pricing. This transparency is valuable for both consumers and insurers seeking to understand or refine pricing strategies.

Data splitting

Prior to subjecting the model data to training it is split 60:20:20 for the training, validation, and test sets. To prevent data leakage and ensure fair evaluation, each individual (`ID`) was assigned to only one set. Allowing the same person in multiple sets could lead to overly optimistic predictions, as the model would have already seen their data. This separation ensures a more accurate assessment of the model's ability to generalise to new customers.

After splitting the data, all categorical variables were converted to factors using `as.factor`. This conversion was applied across all datasets ensuring the model treats the task as classification rather than regression which is essential for random forest models.

Variable Exclusion:

As `ID` has no further predictive capability and `Premium` will lead to data leakage (as `Premium_category` is derived from `Premium`) - both were removed.

Model Training:

Minimal preprocessing was required because random forests work well with both categorical and numerical features. Additionally, the data does not need to be standardised, as random forests are based on decision trees, which are not sensitive to the magnitude of the variables.

Tuning Parameters:

In random forests, it is possible to adjust two key parameters: the number of trees (`ntree`) and the number of features considered at each split (`mtry`).

`ntree`: Increasing the number of trees typically reduces the variance of the model. As more trees are added, the model can generalise better and potentially provide more accurate predictions. However after a certain point, adding more trees yields diminishing returns and increases computational cost. For `ntree`, R's default of 500 trees was used, with additional tests for 400 and 600 trees.

`mtry`: This parameter defines how many features are considered for each decision split in a tree. Increasing `mtry` allows the model to consider more features at each split, potentially capturing more intricate relationships in the data. However, too many features can lead to overfitting. For `mtry`, the middle value of 5 was chosen, as R defaults (Liaw, A. Wiener, M. (2002)) to using the square root of the number of features when training. With 25 features in the training set, 5 was selected as the middle value. Additional values of 4 and 6 were also tested.

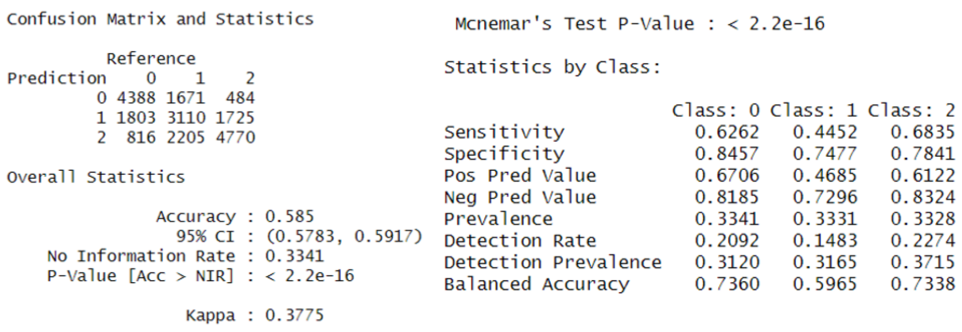
The selected combination was `mtry` = 5 and `ntree` = 600 giving the highest validation accuracy. However, accuracy varied by less than 0.004 across all combinations, suggesting only marginal performance differences (table 2).

mtry	**ntree**	**accuracy**
5	600	0.5836637
4	600	0.5823740
6	400	0.5820874
4	500	0.5817053
5	500	0.5813709
6	600	0.5812276
4	400	0.5810365
6	500	0.5806066
5	400	0.5804156

Table 2: Accuracy results for different mtry and ntree combinations.

4.1.2 Findings

The random forest model trained with `ntree` = 600 and `mtry` = 5 achieved an accuracy of 58% exceeding the 33% baseline from random guessing across three classes. Fig. 1 presents a confusion matrix and associated statistics. It can be seen that sensitivity, specificity, and balanced accuracy indicate that the model performs best in predicting `low` (0) and `high` (2) premium classes, while showing weaker performance on the `middle` (1) class.



```
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2
         0 4388 1671  484
         1 1803 3110 1725
         2  816 2205 4770

Overall Statistics

               Accuracy : 0.585
                 95% CI : (0.5783, 0.5917)
    No Information Rate : 0.3341
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3775
```

```
 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity            0.6262   0.4452   0.6835
Specificity            0.8457   0.7477   0.7841
Pos Pred Value         0.6706   0.4685   0.6122
Neg Pred Value         0.8185   0.7296   0.8324
Prevalence             0.3341   0.3331   0.3328
Detection Rate         0.2092   0.1483   0.2274
Detection Prevalence   0.3120   0.3165   0.3715
Balanced Accuracy      0.7360   0.5965   0.7338
```

Figure 1: Confusion matrix for model

Random forests excel in their ability to reveal which features most influence the model. This is measured using the `mean decrease in accuracy`, which captures how much performance drops when a variable is removed: Higher values indicate greater importance.

The top five features with the highest mean decrease in accuracy were `Payment`, `Vehicle_age_mod`, `Age_mod`, `Years_driving_mod` and `Seniority` (fig. 2). It is important to note that the feature importance scores derived from the mean decrease in accuracy are relative, not absolute. They indicate the comparative contribution of each feature to the model's predictive performance rather than providing a stand alone measure of effect size. Consequently, a higher value signifies greater importance relative to other features, but does not correspond to a direct percentage improvement in model accuracy.

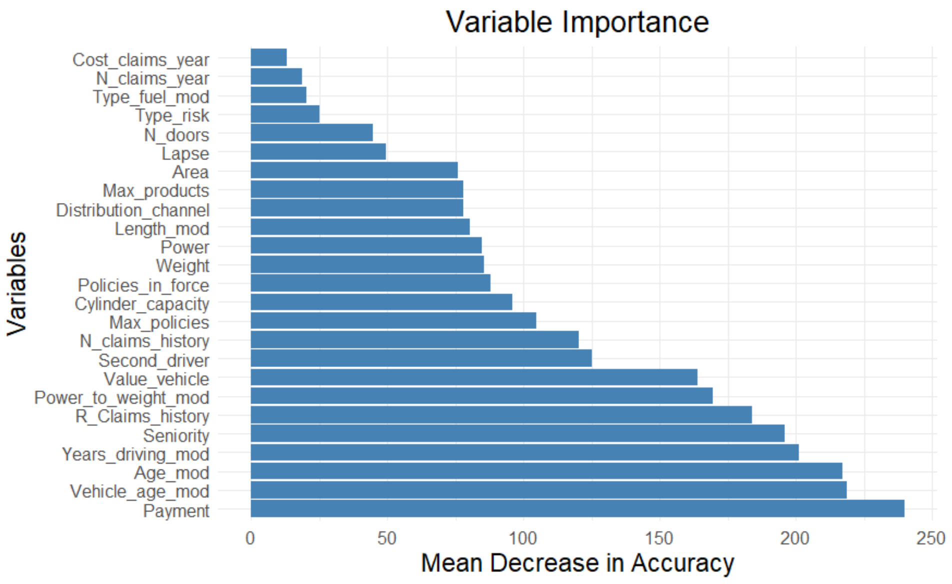


Figure 2: Importance of each feature

`Payment` being the most important feature was unexpected due to its categorical nature with only two possibilities annual or semi-annual. Unfortunately, one limitation of the random forest model is that it doesn't allow us to directly see the relationship between features, whether they are positive, negative, linear, or non-linear. Therefore, while it's not entirely clear how or why `Payment` became such an influential feature, it remains an interesting finding.

`Vehicle_age_mod`, the number of years a vehicle has been on the road, emerged as the second most important factor slightly ahead of `Age_mod`. While older cars may cost less to repair, they may also be more prone to mechanical issues and accidents, making their impact on premiums unclear. Nonetheless, its high importance suggests that at least in this case the insurer views it as a key risk indicator.

`Age_mod` and `Years_driving_mod` are expected to be strong predictors as both are believed to correlate with safer driving behaviour. Greater driving experience may improve decision-making, and increased age is generally associated with greater maturity. Notably, `Age_mod` ranks higher in importance, possibly because age offers a more consistent signal across individuals, while `Years_driving_mod` varies depending on when a person first obtained their licence.

Other Interesting Results:

`R_claims_history`, `N_claims_history`, and `N_claims_year` all reflect claims behaviour. The most predictive of these is the normalised measure `R_claims_history` which captures both frequency and duration of insurance. This suggests insurers prioritise rate-based indicators when assessing risk.

4.2 Linear regression

4.2.1 Variable Correlations:

A correlation matrix is presented (Table 3) to investigate how linearly correlated the other variables are with Insurance Premium:

Var 1	**Var 2**	**Value**
Premium	Premium_category_high_mod	0.70574553
Premium	Premium_category_low_mod	-0.58824991
Premium	Value_vehicle	0.45604352
Premium	Power	0.43161430
Premium	Weight	0.41999981
Premium	Length_mod	0.40743744
Premium	Motorbike_mod	-0.36833910
Premium	Doors_0_mod	-0.36833910
Premium	Cylinder_capacity	0.35241883
Premium	Vehicle_age_mod	-0.29392893

Table 3: Top correlated features with Premium

Post-preprocessing (section 3), the data was again split in a chosen Train (Tr)/Test (Te) 70/30 split with similar protective measures as per 4.1.1 regarding `ID` etc. As premium categories are ultimately derived from Premium their high correlation is fully expected and they must be excluded in the interests of data leakage.

Standardisation: As per Hastie et al. (2009) we apply feature standardisation to the numerical variables before training the model, so that particular numerical variables with larger magnitudes do not dominate the model's optimisation or distance calculations. This is done by computing the (sample) mean and (sample) standard deviation for each of the numerical features (Table A1, appendix) from the training set and subtracting from each instance the mean and then dividing by the standard deviation. The same mean and variance used in the standardisation of the Tr are rigourously applied to the corresponding columns in the Te to prevent Te leaking into Tr. The trained model is thus "untouched" by the test dataset and the test error will provide an unbiased estimator of the generalisation error.

Loss Function: RMSE

To predict `Premium` the loss function chosen was RMSE (Root-mean square error); this was a regression problem with the aim of minimising the difference between the predicted `Premium` and the true `Premium`; RMSE was chosen over MSE (mean squared error) to avoid a less interpretable unit of euros squared.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Other performance Metrics: MAE and $\mathbf{R^2}$

RMSE is very sensitive to outliers. MAE (mean absolute error) treats all errors equally unlike RMSE which penalises larger errors more heavily due to the squaring. MAE is more robust and if the dataset has outliers (which it does) it provides a more balanced view of how the model is performing, while also maintaining the same units.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \bar{y}_i|$$

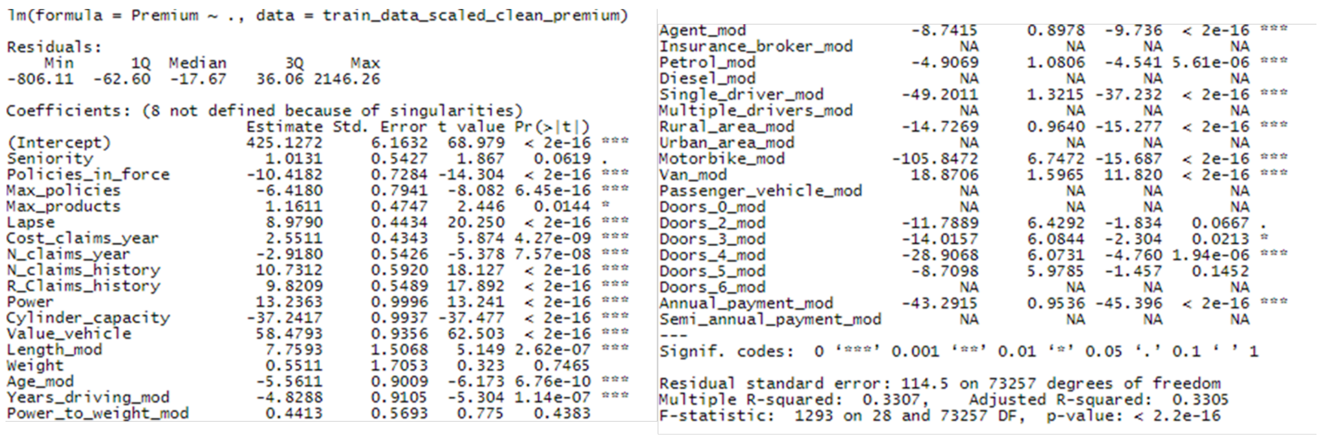
R^2 evaluates how well the model explains the variation in the `Premium`. The numerator of the 2nd term corresponds to the MSE of the model. The denominator of the 2nd term corresponds to the baseline MSE denoting the mean of the dependent variable `Premium`; a higher R^2 value implies the model explains more of the variation in the `Premium` data thus suggesting a better fit.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

4.2.2 Linear Model (LM):

Linear Model justification

The Linear Model is easily interpretable where each coefficient directly conveys how much a 1 unit increase in the predictor independently affects the `Premium`. For this project, it will also act as a baseline benchmark to compare with more complex models to ascertain if any improvement is made with added complexity.



```
lm(formula = Premium ~ ., data = train_data_scaled_clean_premium)

Residuals:
    Min      1Q  Median      3Q     Max 
-806.11  -62.60  -17.67   36.06 2146.26 

Coefficients: (8 not defined because of singularities)
                         Estimate Std. Error t value Pr(>|t|)    
(Intercept)              425.1272     6.1632  68.979  < 2e-16 ***
Seniority                  1.0131     0.5427   1.867   0.0619 .  
Policies_in_force        -10.4182     0.7284 -14.304  < 2e-16 ***
Max_policies              -6.4180     0.7941  -8.082 6.45e-16 ***
Max_products               1.1611     0.4747   2.446   0.0144 *  
Lapse                      8.9790     0.4434  20.250  < 2e-16 ***
Cost_claims_year           2.5511     0.4343   5.874 4.27e-09 ***
N_claims_year             -2.9180     0.5426  -5.378 7.57e-08 ***
N_claims_history          10.7312     0.5920  18.127  < 2e-16 ***
R_Claims_history           9.8209     0.5489  17.892  < 2e-16 ***
Power                     13.2363     0.9996  13.241  < 2e-16 ***
Cylinder_capacity        -37.2417     0.9937 -37.477  < 2e-16 ***
Value_vehicle             58.4793     0.9356  62.503  < 2e-16 ***
Length_mod                 7.7593     1.5068   5.149 2.62e-07 ***
Weight                     0.5511     1.7053   0.323   0.7465    
Age_mod                   -5.5611     0.9009  -6.173 6.76e-10 ***
Years_driving_mod         -4.8288     0.9105  -5.304 1.14e-07 ***
Power_to_weight_mod        0.4413     0.5693   0.775   0.4383    
```

```
Agent_mod                  -8.7415     0.8978  -9.736  < 2e-16 ***
Insurance_broker_mod            NA         NA      NA       NA    
Petrol_mod                 -4.9069     1.0806  -4.541 5.61e-06 ***
Diesel_mod                      NA         NA      NA       NA    
Single_driver_mod         -49.2011     1.3215 -37.232  < 2e-16 ***
Multiple_drivers_mod            NA         NA      NA       NA    
Rural_area_mod            -14.7269     0.9640 -15.277  < 2e-16 ***
Urban_area_mod                  NA         NA      NA       NA    
Motorbike_mod            -105.8472     6.7472 -15.687  < 2e-16 ***
Van_mod                    18.8706     1.5965  11.820  < 2e-16 ***
Passenger_vehicle_mod           NA         NA      NA       NA    
Doors_0_mod                     NA         NA      NA       NA    
Doors_2_mod               -11.7889     6.4292  -1.834   0.0667 .  
Doors_3_mod               -14.0157     6.0844  -2.304   0.0213 *  
Doors_4_mod               -28.9068     6.0731  -4.760 1.94e-06 ***
Doors_5_mod                -8.7098     5.9785  -1.457   0.1452    
Doors_6_mod                     NA         NA      NA       NA    
Annual_payment_mod        -43.2915     0.9536 -45.396  < 2e-16 ***
Semi_annual_payment_mod         NA         NA      NA       NA    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 114.5 on 73257 degrees of freedom
Multiple R-squared:  0.3307,	Adjusted R-squared:  0.3305 
F-statistic:  1293 on 28 and 73257 DF,  p-value: < 2.2e-16
```

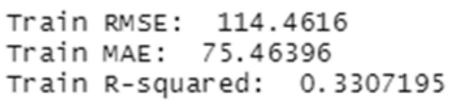
Figure 3: Linear Regression Output

Fig. 3 shows the results of the linear regression in predicting `Premium`. Among all the predictors, motorbikes had the most negative coefficient of -105.85, making it the strongest inverse predictor of `Premium`. In contrast, the value of the vehicle was the strongest positive predictor of `Premium` with positive coefficient of 58.45.

This agrees with common knowledge as motorbikes are typically less expensive than other forms of transport and so tend to be less expensive to repair, and are likely to fetch cheaper claims than other vehicles for the insurance company. This is supported by the dataset in which the average value of motorbikes (4,126.89 euros) is significantly less than both the average value of a van (19,846.63

euros) and the average value of a passenger vehicle (19,475.04 euros). Furthermore, the average yearly cost of claims for Motorbikes (36.35 euros), from the dataset, is also lower than both the average yearly cost of claims of vans (166.34 euros) and passenger vehicles (165.10 euros). These differences in average value and average cost of yearly claims partially justify the significantly lower premiums associated with motorbike policies. Lastly, method of payment also appeared to be a strong predictor of `Premium` - those who pay yearly may be incentivised with discounts compared to those on a semi-annual payment plan.

Assessing the Training Error

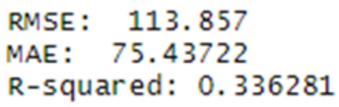


```
Train RMSE:  114.4616
Train MAE:  75.46396
Train R-squared:  0.3307195
```

Figure 4: Linear model training error

The residuals (fig. 3), difference between the actual and predicted premium values, have a median of -17.67 euros indicating minimal bias. The adjusted R^2 (R^2 penalising unnecessary predictors) is 0.3305 (fig. 5) explaining a disappointing 33.05% of the variance in `Premium`. The F-Statistic (fig. 3), which tests the null hypothesis that all the regression coefficients (except the intercept) are 0, had a large value of 1293 and p-value < 0.01 thus rejecting the null hypothesis and concluding that the linear model is statistically significant; note p-values measure the probability that an observed effect is due to chance - thus statistical significance is likely just owed to a large number of observations. The low adjusted R^2 value is likely resultant from the simplicity of a linear model being unable to capture the real world complexities of insurance pricing.

Assessing the Test Error



```
RMSE:  113.857
MAE:  75.43722
R-squared: 0.336281
```

Figure 5: Linear model test error

Similar to the Tr set the Te R^2 is 0.3363, this is consistent with the training R^2 hinting no major overfitting. The Te RMSE is 113.86 so on average the model's predictions deviate from the true premium by 113.86 euros, which is lower than that on the Tr set; again no significant overfitting. Since the Te R^2 is higher, the Te RMSE is lower than for the Tr set and MAE of both are very close, the issue lies with the lack of complexity of the model. The results of predicting `Premium` by means of linear regression are plotted in fig. 6.

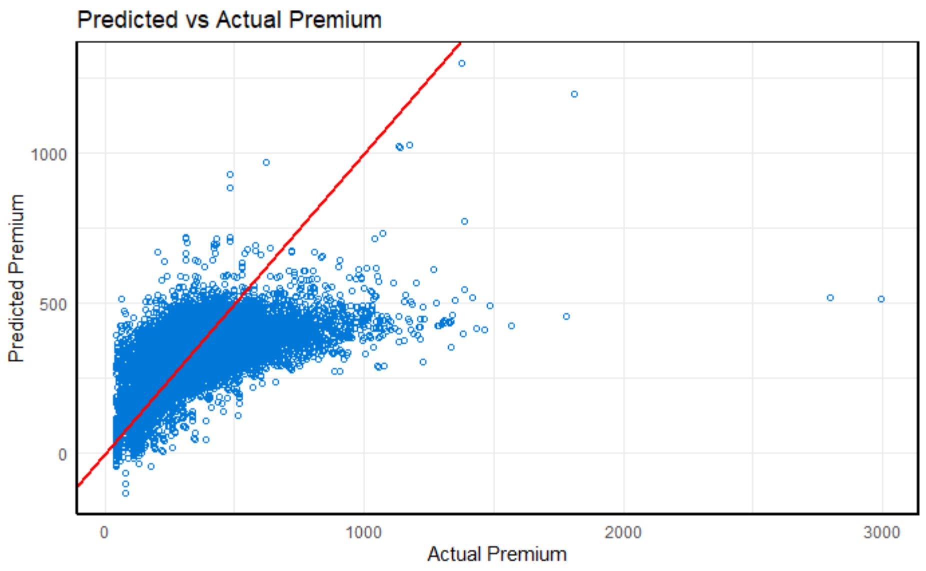
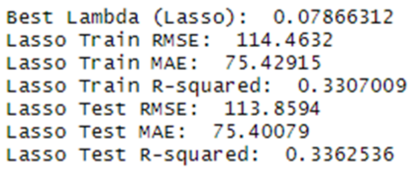


Figure 6: Simple linear regression for predicting Premium

4.2.3 Lasso and Ridge Regression

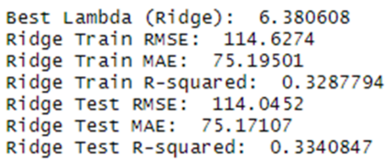
The simple linear model showed no signs of overfitting since both the error and R^2 in the Train and Test sets were similar, hinting that the issue lies in the complexity of the model. Shrinkage/regularisation methods featured in Lasso and Ridge were unlikely to deliver any major improvements but could still yield some improvement. Lasso regression performs feature selection by shrinking some coefficients to 0 namely: `Diesel_mod`, `Multiple_drivers_mod`, `Passenger_vehicle_mod`, `Doors_3_mod`, `Semi_annual_payment_mod`.

To determine the optimal value of lambda, the rate of penalising number of predictors, 10-fold cross validation was utilised, splitting the training set into 10 equal folds. For each iteration 9 folds were used to train the model and 1 fold to validate it. The `cv.glmnet()` library searches through a grid of lambda values, and for each value of lambda, it undergoes Lasso/Ridge Regression on the 9 folds and predicts on the last remaining fold. This is then repeated for all 10 folds and thus for each lambda value it computes the mean cross validation error (MSE), and picks the lambda which gives the lowest mean cross validated error. However, there was minimal change in the RMSE, MAE and R^2 shown in fig. 7 & 8.



```
Best Lambda (Lasso):  0.07866312
Lasso Train RMSE:  114.4632
Lasso Train MAE:  75.42915
Lasso Train R-squared:  0.3307009
Lasso Test RMSE:  113.8594
Lasso Test MAE:  75.40079
Lasso Test R-squared:  0.3362536
```

Figure 7: Lasso RMSE, MAE, R^2



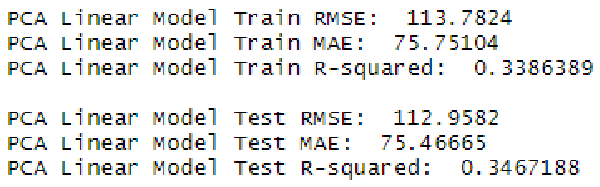
```
Best Lambda (Ridge):  6.380608
Ridge Train RMSE:  114.6274
Ridge Train MAE:  75.19501
Ridge Train R-squared:  0.3287794
Ridge Test RMSE:  114.0452
Ridge Test MAE:  75.17107
Ridge Test R-squared:  0.3340847
```

Figure 8: Ridge RMSE, MAE R^2

Principal component analysis was then applied to the basic linear model to see if there was any improvement in predictive power.

4.2.4 Principle Component Regression (PCR)

The models created from the basic linear, Ridge and Lasso regression were poor in explaining the variance in `Premium`; Principle Component Analysis (PCA) was used to see if executing the basic linear regression only on a subset of principle components (PC), more formally PCR, yielded better results. The PC directions that capture most of the variance are kept in the hopes of reducing dimensionality, eliminating multicollinearity and noise in the data.



```
PCA Linear Model Train RMSE:  113.7824
PCA Linear Model Train MAE:  75.75104
PCA Linear Model Train R-squared:  0.3386389

PCA Linear Model Test RMSE:  112.9582
PCA Linear Model Test MAE:  75.46665
PCA Linear Model Test R-squared:  0.3467188
```

Figure 9: PCA RMSE, MAE and R^2

Using the PC that cumulatively captures 99% of the variance (PC1 - PC24 Table A2), a linear model was constructed with results shown in fig 9. This approach was unsuccessful as there was no significant improvement made to the RMSE, MAE or the R^2 values (explaining about 34.67% of the variance in Premium. Moving away from linear models Gradient Boosting was considered next.

4.3 GBM (Gradient Boosting):

Instead of aggregating multiple uncorrelated trees seen in the Random Forest approach (see Section 4.1), Gradient Boosting builds trees sequentially Burkov (2020), where each new tree is trained to correct the errors (residuals) made by the previous trees; over many iterations the models are combined in a weighted manner to form an overall predictor.

Hyperparameters

`n.trees:` – Number of trees (boosting iterations). Each tree corrects the residuals from the previous tree. More trees will generally improve performance but also can result in overfitting to the training dataset.

`interaction.depth:` – Maximum depth of the individual trees: this is a measure each trees' complexity; higher depth allows the trees to capture more complex patterns.

`shrinkage:` – refers to the learning rate, how much each tree contributes to the final prediction. Smaller values make the model less prone to overfitting.

`n.minobsinnode:` Minimum number of observations in the terminal node. This controls the regularisation. Larger values make the trees simpler and reduces overfitting.

Tuning the Hyperparameters:

For each combination of hyperparameters, the model was trained on 4 folds and validated on the 5th; this was repeated 5 times allowing each fold to be the validation set once. The model's performance was then averaged across the 5 folds and the hyperparameters which delivered the smallest RMSE value were selected. Initially the below values for the hyperparameters were used yielding 36 different combinations. To tune the hyperparameters, Grid search and 5-fold cross-validation were utilised:

Parameter	**Values**
`n.trees`	100, 200, 300
`interaction.depth`	1, 3, 5
`shrinkage`	0.01, 0.1
`n.minobsinnode`	10, 20

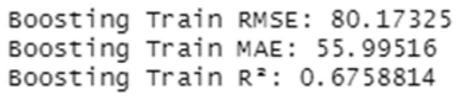
Table 4: Grid of tuning parameters

The combination that delivered the smallest RMSE was: `n.trees` = 300, `interaction.depth` = 5, `shrinkage` = 0.1 and `n.minobsinnode` = 10.

The model is then re-trained on the entire training set using the best tuned hyperparameters before being applied on the test set. Notice that in the above grid (Table 4), the limits for all the hyperparameters have been reached: the optimal model consists of 300 trees, a tree depth of 5, shrinkage of 0.1 and minimum number of observations in the terminal node of 10. It is likely that the optimal hyperparameters lie out of range. Running the model with the above listed hyperparameters delivers a significant improvement relative to the linear model in terms of RMSE, MAE and R^2 (fig. A1). The grid was expanded and the best model continued to reach the borders of the grid until the use of the one standard error rule (model selection with the smallest number of trees that has a validation error within one standard error of the 'best' model with the lowest validation error) due to only marginal improvements in the validation RMSE.

After tuning [1] and considering computational limitations, the following were selected (fig. A2): `n.trees` = 700, `interaction.depth` = 15, `shrinkage` = 0.1 and `n.minobsinnode` = 10.

Assessing the Training Error (of the final tuned model):

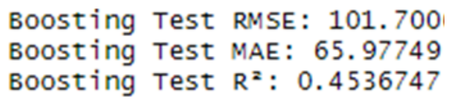


```
Boosting Train RMSE: 80.17325
Boosting Train MAE: 55.99516
Boosting Train R²: 0.6758814
```

Figure 10: Train RMSE, MAE, R^2 for the 'Best' Boosting model

Direct comparison will be made to the linear model since decision trees were used to predict the premium category and both Lasso and Ridge regression yielded very similar results. Compared to the basic linear model, gradient boosting yielded a noticeable improvement when fitting to the training data. The training RMSE had decreased from 114.46 euros to 80.17 euros, the MAE decreased from 75.46 euros to 56 euros and the R^2 value increased from 33.07% to 67.59%; gradient boosting resulted in both lower error and increased explanation of variation of `Premium` over the training data.

Assessing the Test Error:



```
Boosting Test RMSE: 101.700
Boosting Test MAE: 65.97749
Boosting Test R²: 0.4536747
```

Figure 11: Test RMSE, MAE, R^2 for the 'Best' Boosting model

[1]Further improvements were attempted by increasing the grid but the limiting factor of hardware prevented this; results still deliver an improvement to previous linear models in predicting `Premium`.

The test RMSE of 101.70 euros is much higher than the training RMSE of 80.17 euros, indicating the model is overfitting to the training data; further parameter tuning is required. The R^2 value also decreased to 45.37% , much lower than the Tr 67.59% but still higher than the basic linear model 33.63%. The boosting model explains more of the variance in the `Premium` and provides an improved model over that of basic linear regression (and Lasso and Ridge regression) in predicting `Premium`.

Feature Importance:

For regression, Gradient Boosting calculates feature importance Kuhn (2025) by measuring how much each feature reduces the squared error when it is used in splits across all trees. Each time a feature is used to split a node in a decision tree, the algorithm calculates the decrease in the residual sum of squares (RSS) caused by the split. These reductions then get accumulated over all the trees in the ensemble. The more a variable contributes to reducing the error across the ensemble the higher it's importance. This is done via the `caret` library. These total contributions are then summed across all the boosting iterations (`n.trees`), and then scaled relative to the most important feature shown in fig 12.

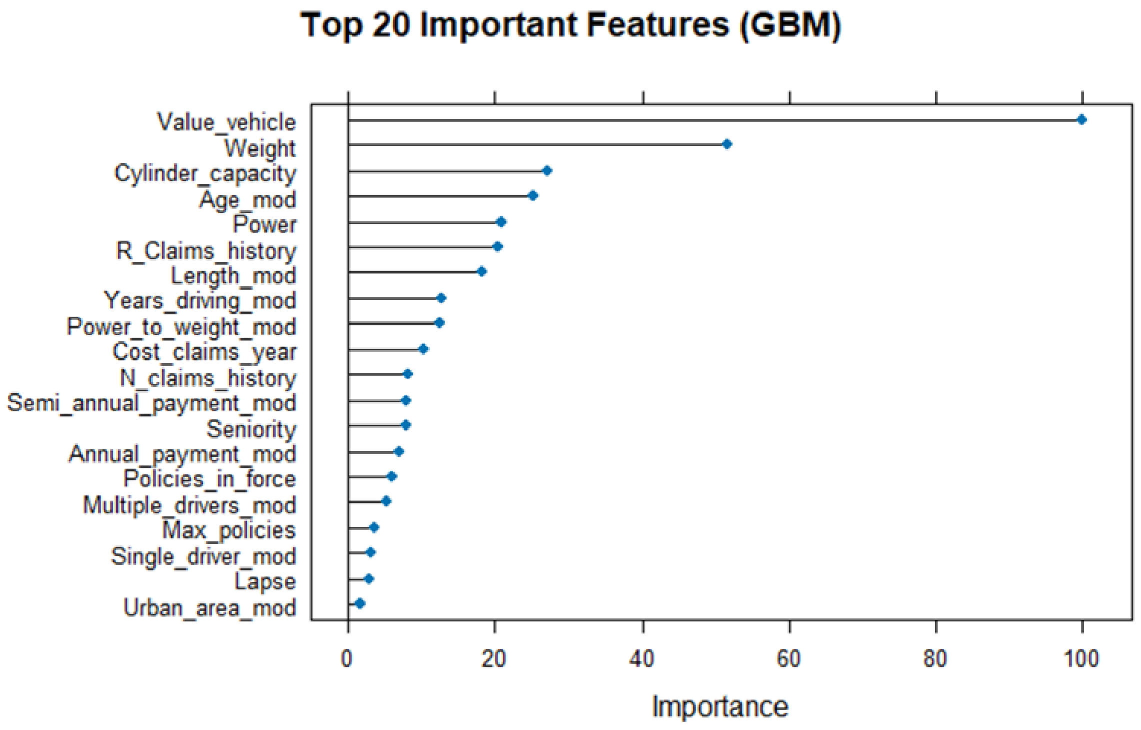


Figure 12: Variable Importance in Boosting model with 700 trees

4.4 Neural Network for Predicting Premium

4.4.1 Neural Network Justification

Linear correlation of features with premium was performed however results were considered poor. Below shows all features with independent correlation to `Premium` greater than equal to 0.4 or less than equal to -0.4.

```
  Premium Value_vehicle         Power        Weight    Length_mod
1.0000000     0.4556598     0.4291511     0.4198945     0.4071668
```

Notwithstanding the above, an initial neural network was attempted for predicting `Premium`. The methodology was similar to that reported below to predict `Log_premium_mod`. This yielded no significant improvement to previous above models and as such this work is not reported in full for brevity's sake.

Individual relationships between `Premium` and other features (fig. A3 & A4) were investigated by means of graphical visualisation; a type of exponential relationship was observed; exponential growth (e.g. `Year_matriculation`) or decay (e.g. `Power_to_weight_mod`) with respect to `Premium`. The presence of a non-linear relationship was also supported by fig. 4 in section 4.2.2. The prediction of the logarithm of `Premium` denoted `Log_premium_mod` was considered in the hope of generating a model with a higher R^2, lower MAE and lower MSE compared to previous discussed models. This resulted in multiple features with an increased correlation at approximately 0.5:

```
 Log_premium_mod            Weight        Length_mod             Power
       1.0000000         0.5520587         0.5509437         0.5060035
   Value_vehicle Cylinder_capacity     Motorbike_mod       Doors_0_mod
       0.4923010         0.4803679        -0.5535944        -0.5535944
```

Feature engineering:

New features were incorporated to incentivise the model to find more complex relationships with `Premium` and ultimately a model that had more predictive power for `Premium`.

`Log_premium_mod` was created directly from taking the logarithm of `Premium`.

$\texttt{Depreciated\_value\_mod} = \texttt{Value\_vehicle} \times 0.8^{\texttt{Vehicle\_age\_mod}}$

$\texttt{Average\_cost\_mod} = \texttt{Depreciated\_value\_mod} \times \texttt{R\_claims\_history}$

$\texttt{Year\_cost\_mod} = \texttt{Depreciated\_value\_mod} \times \texttt{N\_claims\_year}$

$\texttt{Total\_cost\_mod} = \texttt{Depreciated\_value\_mod} \times \texttt{N\_claims\_history}$

Note there seems to be a mistake in the dataset claiming that `Value_vehicle` is the market value of the vehicle on 31/12/2019. This is unlikely as the contracts in the dataset end before 2019 and many observations with very similar/same driver and vehicle characteristics including `Value_vehicle` have drastically differing values for both `Premium` and `Vehicle_age_mod` (Fig. A5 A6). This is discussed later in section 5.

For comparison a heat map (fig.13) can be used to view the correlations of `Premium` and `Log_premium_mod` with respect to all other features - notice that `Log_premium_mod` seems to have some stronger correlations with other features compared to that of `Premium`.

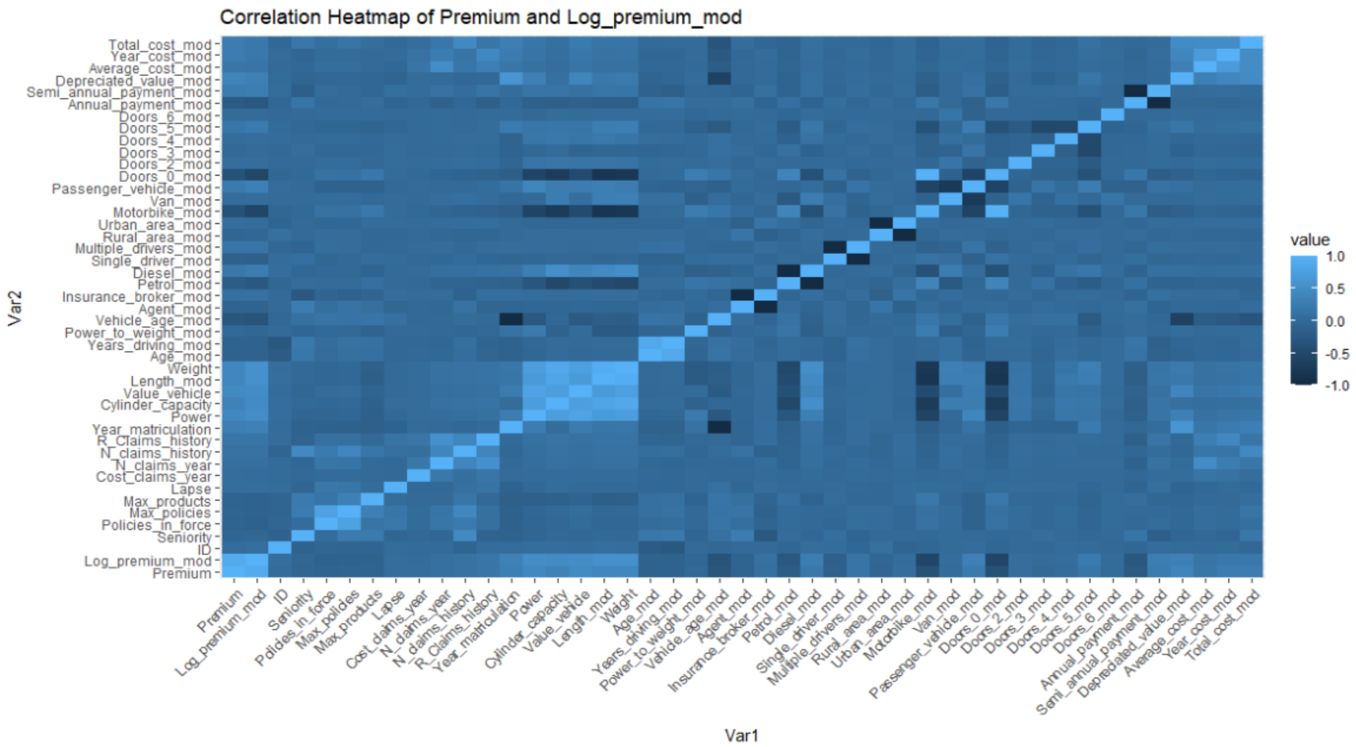


Figure 13: Heat map showcasing the independent correlations of standardised features

Due to complexity, a neural network was then created to capture relationships to best correlate with standardised `Log_premium_mod` and from this `Premium` itself.

4.4.2 Neural Network Preparation

A series of preparations was considered to satisfy the requirements for a successful neural network model:

Data standardisation: As per section 4.2.1 the dataset was standardised excluding categorical variables, that instead were one-hot encoded and `ID` was also omitted. The mean and standard deviation used was from the training data separate from the test to avoid data leakage.

Consideration of data trimming and feature selection: removing extreme outliers (bottom and top 0.01%) of `Premium`/`Log_premium_mod` observations was considered on the basis that these skew the final results leading to a poorer fit. This was not the case and actually resulted in a worse model possibly due to the complex nature of neural networks – data trimming was therefore omitted. Choosing features with the highest independent correlation to `Log_premium_mod` or features chosen by other models was also considered. This too was not performed as it also led to worse results likely for the same reason as above.

Train-Test split: The dataset was split into 90% training and 10% testing to ensure the model generalises well to unseen data; with over 100,000 observations 10% test data is still significant and gives the model additional data to train on compared to e.g. a 70% 30% split. All observations with the same `ID` (driver) were kept together to avoid data leakage. The training data also includes 10% validation data such that training alone is 81% and validation of the training is 9% (total 90%).
Removal of ID and premium category columns: The `ID` feature was dropped after splitting the data into training and testing sets since it does not contribute predictive value. The features `Premium_category_low_mod`, `Premium_category_medium_mod` and `Premium_category_high_mod` were

removed to avoid data leakage since these columns were created from premium.

Tensor conversion: Training data denoted x_train and y_train were converted to tensors for compatibility with Google TensorFlow.

4.4.3 Model Architecture

The model is built sequentially with the following parameters applied at each stage and collectively summarised in table 5:

Neuron units: Initially the number of neuron units starts high at 512 and decreases step wise to the 12th and final output layer (1 unit only).

Activation function selection: For the first layer, the leaky-RELU (Leaky Rectified Error Linear Unit) activation function was selected post benchmarking over the options of RELU (Rectified) and GELU (Gaussian). Of note, Leaky-RELU does allow the early-stage iterative progression of a small quantity of negative neurons (which may become positive later on aiding correlation). Intermediate layers utilised ReLU. The second to last layer used the GeLU activation function for its smoothing properties – the reasoning was slight variations in input (e.g. `Age_mod`, `Value_vehicle`) generally do not cause disproportionate shifts in `Premium` as Insurance premiums should not have abrupt jumps; increasing or decreasing continuously based on factors like risk, vehicle type, or claims history. The final activation was linear for the regression problem.

Batch normalisation: Performing batch normalisation normalises activations to stabilise training and improve convergence with features centred (mean ≈ 0) and scaled (variance ≈ 1), preventing extreme values and unstable learning early on and affecting subsequent layers. This is applied to the early layers and not necessary later on.

Dropout: To prevent overfitting with a large batch number some activations are randomly set to 0. During training, e.g. 30% of neurons in a given layer are randomly dropped. This discourages the model from relying on specific neurons and avoids overfitting. As per normalisation, this is only applied in the first half of the layers. In later levels, lower dropout rates (set to 0) helps retain more information.

Output layer: This layer has only one unit because this is a regression task and uses linear activation to ensure the model outputs are continuous values.

Layer no.	Type	no. of neurons	Act. func.	Batch Norm.	Dropout
1	First	512	L-ReLU	YES	0.3
2	Intermediate	384	ReLU	YES	0.3
3	Intermediate	256	ReLU	YES	0.2
4	Intermediate	192	ReLU	YES	0.2
5	Intermediate	128	ReLU	YES	0.2
6	Intermediate	96	ReLU	NO	0
7	Intermediate	64	ReLU	NO	0
8	Intermediate	48	ReLU	NO	0
9	Intermediate	32	ReLU	NO	0
10	Intermediate	16	GeLU	NO	0
11	Output	1	Linear	NO	0

Table 5: Parameters set for neural network layer computations.

4.4.4 Neural Network Set-up

The loss function MSE (Mean square error) was used as it penalises larger errors more heavily improving the accuracy of the model. For training the neural network an early stopping of 8 was introduced. This stops training when validation loss no longer improves after 8 epochs, preventing unnecessary computations. An attempt to prevent the case of overfitting when error increases on the validation set whilst training error is still decreasing. A large batch size of 2048 aided in smooth gradient updates and stability in training - accompanied with dropout overfitting was prevented. A sufficient 50 epochs allowed enough time for the model to learn complex relationships in the data though both training and validation loss plateaued before this – perhaps a sign of further model hyperparameter tuning/feature engineering.

Adam Optimiser According to Kingma and Ba (2014) Adam is an option available within TensorFlow that can be used instead of the classical stochastic gradient descent procedure to update network weights iteratively based in the training data. Adam realises the benefits of both AdaGrad and RMSProp:

Adaptive Gradient Algorithm (AdaGrad): This maintains a per-parameter learning rate that improves performance on problems with sparse gradients. Root Mean Square Propagation (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing) working well on noisy data.

Adam parameters employed: Gradient clipping of 0.75 was implemented to avoid any case of exploding gradients which can occur when gradient values become excessively large during backpropagation. This instability can lead to chaotic updates, making training inefficient or even causing the model to diverge. Clipping by Norm was also utilised meaning any gradients in a gradient vector exceeding magnitude 0.75 are each scaled down proportionally.

4.4.5 Neural Network Training Results

After training the neural network fig. 14 shows both the mean squared error loss on both the training set and the validation set with respect to number of epochs (separate to the testing set - the validation set was 10% of the training set per iteration). This shows that the validation loss plateaued at around 10 epochs suggesting potential room for improvement mentioned in section 4.3.4.

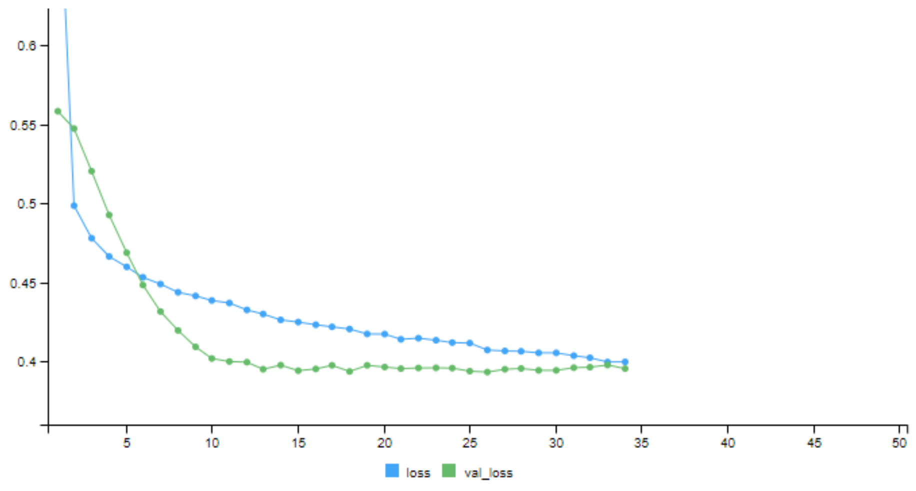


Figure 14: Neural Network loss vs. epochs for predicting standardised log(Premium) - blue is training loss and green is validation loss.

Completion of the neural network training resulted in the following figure for predicted standardised log(Premium) vs. actual standardised log(Premium) in the test data (fig. 15).

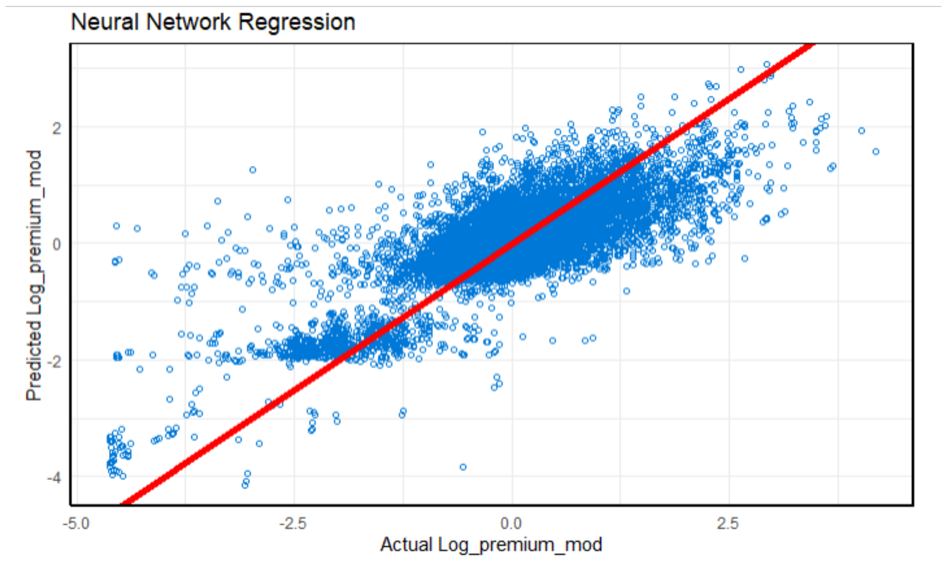


Figure 15: Neural Network results for predicting standardised log(Premium)

The data for predicted standardised log(Premium) was then converted back to Premium by unstandardising then taking the exponent. This was then comapred to the original unaltered Premium data with the same corresponding ID's as the ID's in the test data only.

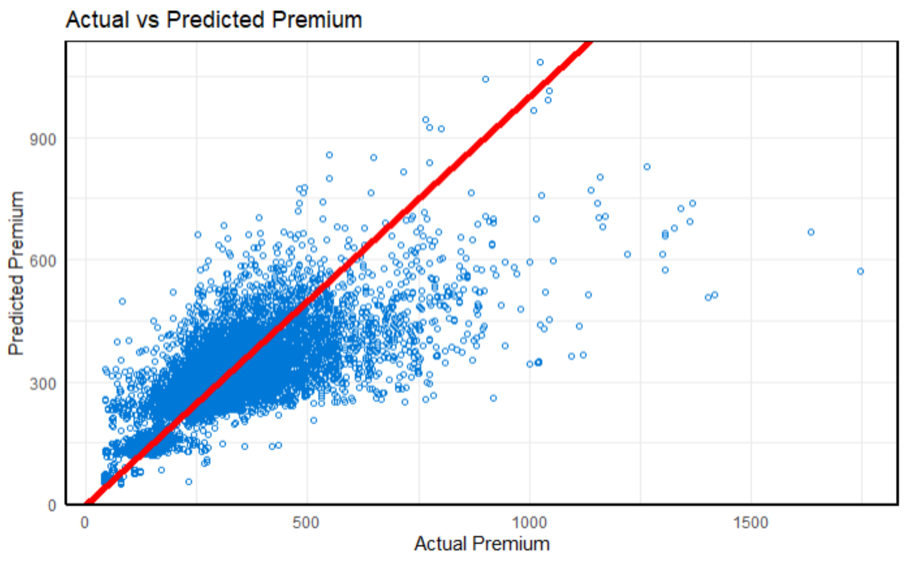
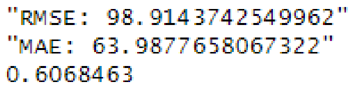


Figure 16: Neural Network results for predicting Premium

4.4.6 Model Evaluation

The neural network model attempted to predict the standardised log(Premium). After much tuning with hyperparameters and experimenting with feature engineering a sufficient model was created with error metrics and R^2 shown in fig. 18.



```
"RMSE: 98.9143742549962"
"MAE: 63.9877658067322"
0.6068463
```

Figure 17: Neural Network RMSE, MAE and R^2

The neural netowrk model had a MAE of 64 euros, RMS of 99 euros and an R^2 value of 0.61 suggesting the model explains around 60% of the variance in the model. On average an error of 64 euros with a linear penalty is considered good meaning the prediction for Premium is not, on average, drastically different to the actual premium – the premium prediction does not have to be exact but for many consumers an estimate can prove quite useful to see whether insurance companies are grossly overcharging their clients or evaluating a range of vehicle options.

The above is considered Moderate predictive power – model has capturing some patterns in the data, but there's still room for improvement. Better than random since $R^2 = 0$ would mean no predictive ability, 0.61 indicates the model is doing significantly better than chance. Some variance is unaccounted for as either additional predictors are needed, or the model might need adjusting.

At this juncture the data set itself may benefit from rigorous real-world assessment. Although outside of the scope of this work, certain anomalies were noted. A significant example is that of 'Vehicle_value' which is stated as the value of the vehicles at some date in 2019. In analysing the data, many instances of very old vehicles were noted with humble characteristics yet a value which appeared to be new vehicle list price when first registered – i.e. not depreciated. It is reasoned that if this is the case, the underwriting premium would be unreasonably skewed unless the policy was a replacement with new exact model as opposed to a discounted cash settlement based on depreciated value as is the industry norm.

5 Conclusion

This project aimed to develop a predictive model for insurance premiums using data from a Spanish insurance firm. Various models were employed: random forest (classification), linear regression, gradient boosting (regression), and neural networks (regression) to assess their performance in predicting premiums. For classification of `Premium` the random forest model achieved a modest accuracy of correctly predicting `Premium_category` of 58%. For predicting the value of `Premium` neural networks showed particular strength when predicting `log(Premium)` explaining 61% of the variation in `Premium` (R^2), a MAE of of 64 euros and RMSE of 99 euros outperforming linear/Lasso/Ridge regression and also gradient boosting; gradient boosting explained 45% of the variation in predicting `Premium` directly, MAE of 66 euros and RMSE of 102 euros.

The key features identified by most models were the vehicle & driver age and years of driving experience aligning with common industry expectations, indicating that insurers place significant weight on these factors as indicators of risk. One unexpected finding was the importance of the feature `Payment`, particularly in the random forest model and linear regression models. Despite being a categorical variable, which informs the frequency of how often an individual pays their insurance bill, it is a strong predictor of premiums and may indicate a from of credit scoring of the policy holder. Although the models provided useful insights, none achieved a level of prediction for exacting real-world applications, with the neural network model explaining some 61% of the variance in premiums.

A key point to raise is the suspected quality of the dataset. It had only been noticed through later analysis in section 4.3.1 that the feature `Value_vehicle` (labelled as the market value of the vehicle on 31/12/2019) is inconsistent as a large quantity of observations in the dataset have

Date_next_renewal which pre-dates this. Likewise (Fig. A5 A6) there is obviously many instances supporting the inconsistencies in Value_vehicle. A crude attempt was made in feature engineering to overcome this with an annual deprciation rate of 20% but poor quality data could be the reason for less than stellar results. This is a worrying sign as it is unclear what else is incorrect in the dataset. There is no known assurance of the quality of this data e.g. no evidence of approval from a regulatory body and one has to question why this dataset was even made publicly available? Why did the work in Segura-Gisbert, J. Lledó, J. Pavía, J.M. (2025) not even attempt anything past 'shallow' analysis?

The failure of the dataset to account for actual depreciation associated with the vehicle's age is a potentially serious shortcoming. This oversight may not only compromise the model's predictive accuracy (along with other potential errors), but also misrepresent the true significance of the value of a vehicle as a pricing determinant. Whilst this issue was crudely addressed in the section 4.3.1 with a constant depreciation rate for all vehicles over time, future research could improve upon this by modelling depreciation more dynamically, perhaps using industry-standard depreciation curves, age-adjusted valuation models or recorded actual market value of each vehicle per year. Incorporating such refinements would allow a more faithful reflection of real-world vehicle values and, consequently, lead to more robust premium pricing predictions.

A notable limitation of the model is its reliance on data from a single insurance firm in Spain. This raises concerns about the model's ability to generalise to other regions or firms. If the model were to be applied in practice, it would reflect the pricing strategy of the Spanish firm rather than providing a potentially universally applicable prediction; other insurance firms, particularly those in different countries, may prioritise different factors or weight the same factors differently. Future work could benefit from expanding the dataset to include data from multiple insurers across various countries, which would improve the model's robustness and applicability.

Notwithstanding likely dataset errors, the use of various techniques has resulted in a moderate to good correlation for Premium which satisfies the goal of passenger vehicle premium estimator albeit more in relative than absolute terms.

References

Burkov, A. (2020). Understanding gradient boosting machines. `https://medium.com/neuranest/understanding-gradient-boosting-machines-5fb37a235845`.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). An introduction to statistical learning.

Institute of Highway Engineers (2005). Guidelines for motorcycling. `https://motorcycleguidelines.org.uk/the-guidelines/6-0-motorcycle-parking/6-5-motorcycle-parking-resources/`.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Kuhn, M. (2025). Variable importance. `https://topepo.github.io/caret/variable-importance.html`.

Liaw, A. Wiener, M. (2002). Classification and regression by randomforest. `https://cran.r-project.org/web/packages/randomForest/randomForest.pdf`.

Newton, I. (1687). Philosophiae naturalis principia mathematica. *Royal Society.*

Segura-Gisbert, J. Lledó, J. Pavía, J.M. (2025). Dataset of an actual motor vehicle insurance portfolio. *Eur. Actuar. J.*, 15:241–253.

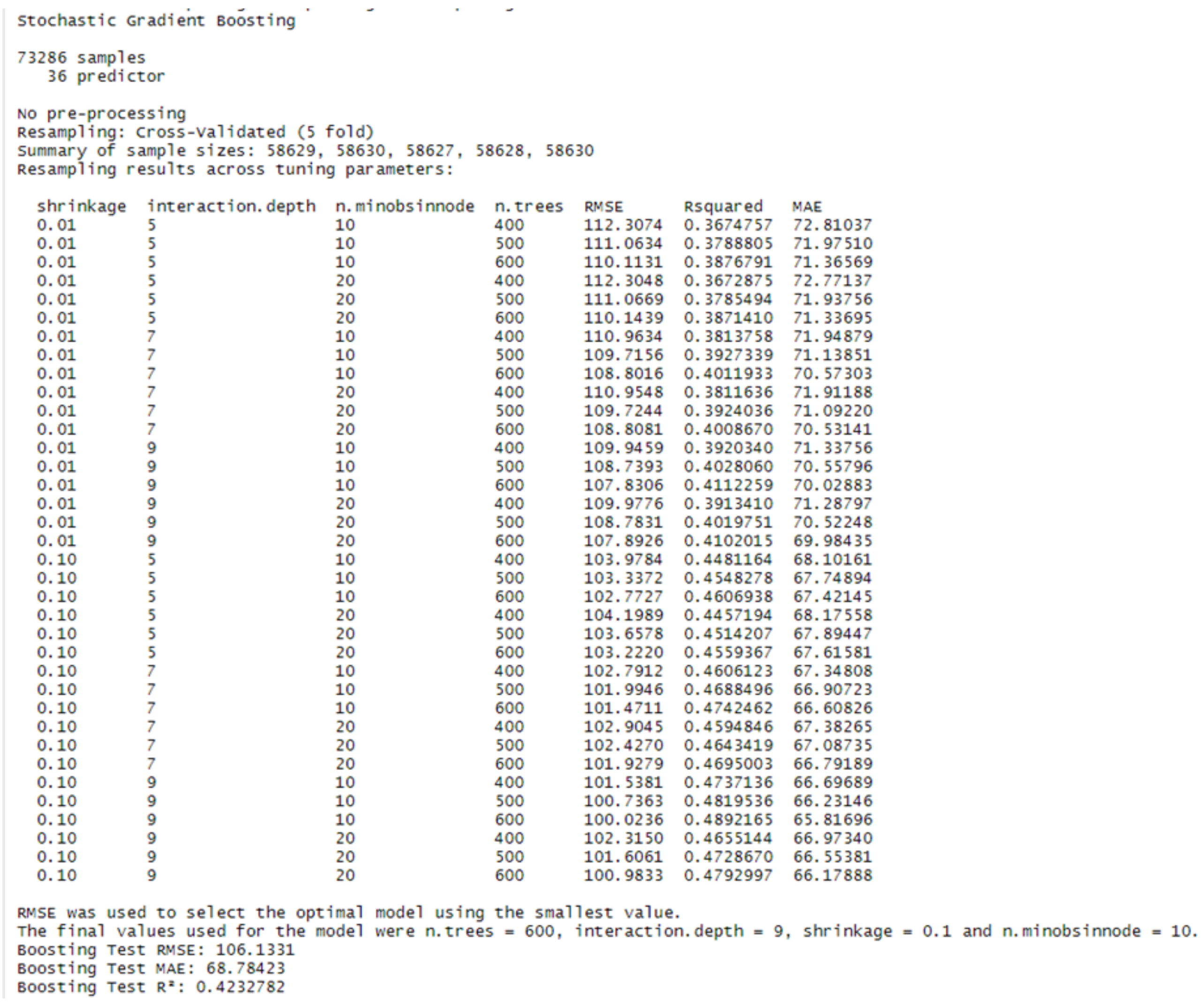
A Appendix

Variable Number	Variable Name
1	Seniority
2	Policies_in_force
3	Max_policies
4	Max_products
5	Lapse
6	Cost_claims_year
7	N_claims_year
8	N_claims_history
9	R_Claims_history
10	Power
11	Cylinder_capacity
12	Value_vehicle
13	Length_mod
14	Weight
15	Age_mod
16	Years_driving_mod
17	Power_to_weight_mod
18	Vehicle_age_mod

Table A1: List of Standardised Variables

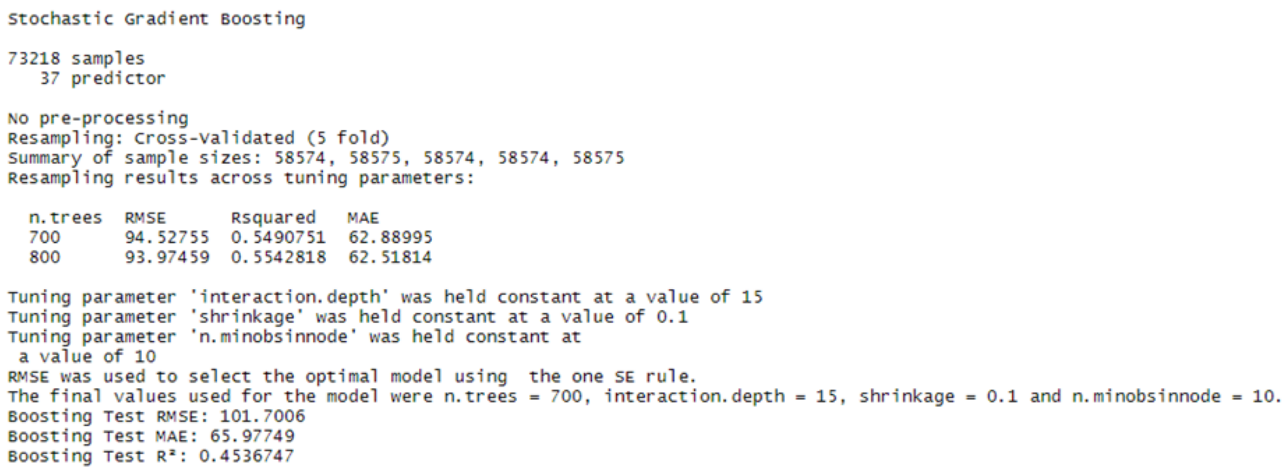
Prin. Component	Std. Dev.	Prop. of Var.	Cumulative
PC1	2.1366	0.2178	0.2178
PC2	1.6315	0.1270	0.3448
PC3	1.4423	0.0993	0.4440
PC4	1.2148	0.0740	0.5145
PC5	1.1267	0.0606	0.5751
...	...	...	...
PC24	0.3419	0.0056	0.9895
PC25	0.3255	0.0050	0.9946
PC26	0.2081	0.0027	0.9968
PC27	0.1956	0.0019	0.9985
PC28	0.1584	0.0012	0.9997
PC29	0.0783	0.0003	1.0000
PC30–PC37	very small	≈ 0	1.0000

Table A2: Summary of PCA Components



```
Stochastic Gradient Boosting

73286 samples
   36 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 58629, 58630, 58627, 58628, 58630
Resampling results across tuning parameters:

  shrinkage  interaction.depth  n.minobsinnode  n.trees  RMSE      Rsquared   MAE
  0.01       5                  10              400      112.3074  0.3674757  72.81037
  0.01       5                  10              500      111.0634  0.3788805  71.97510
  0.01       5                  10              600      110.1131  0.3876791  71.36569
  0.01       5                  20              400      112.3048  0.3672875  72.77137
  0.01       5                  20              500      111.0669  0.3785494  71.93756
  0.01       5                  20              600      110.1439  0.3871410  71.33695
  0.01       7                  10              400      110.9634  0.3813758  71.94879
  0.01       7                  10              500      109.7156  0.3927339  71.13851
  0.01       7                  10              600      108.8016  0.4011933  70.57303
  0.01       7                  20              400      110.9548  0.3811636  71.91188
  0.01       7                  20              500      109.7244  0.3924036  71.09220
  0.01       7                  20              600      108.8081  0.4008670  70.53141
  0.01       9                  10              400      109.9459  0.3920340  71.33756
  0.01       9                  10              500      108.7393  0.4028060  70.55796
  0.01       9                  10              600      107.8306  0.4112259  70.02883
  0.01       9                  20              400      109.9776  0.3913410  71.28797
  0.01       9                  20              500      108.7831  0.4019751  70.52248
  0.01       9                  20              600      107.8926  0.4102015  69.98435
  0.10       5                  10              400      103.9784  0.4481164  68.10161
  0.10       5                  10              500      103.3372  0.4548278  67.74894
  0.10       5                  10              600      102.7727  0.4606938  67.42145
  0.10       5                  20              400      104.1989  0.4457194  68.17558
  0.10       5                  20              500      103.6578  0.4514207  67.89447
  0.10       5                  20              600      103.2220  0.4559367  67.61581
  0.10       7                  10              400      102.7912  0.4606123  67.34808
  0.10       7                  10              500      101.9946  0.4688496  66.90723
  0.10       7                  10              600      101.4711  0.4742462  66.60826
  0.10       7                  20              400      102.9045  0.4594846  67.38265
  0.10       7                  20              500      102.4270  0.4643419  67.08735
  0.10       7                  20              600      101.9279  0.4695003  66.79189
  0.10       9                  10              400      101.5381  0.4737136  66.69689
  0.10       9                  10              500      100.7363  0.4819536  66.23146
  0.10       9                  10              600      100.0236  0.4892165  65.81696
  0.10       9                  20              400      102.3150  0.4655144  66.97340
  0.10       9                  20              500      101.6061  0.4728670  66.55381
  0.10       9                  20              600      100.9833  0.4792997  66.17888

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 600, interaction.depth = 9, shrinkage = 0.1 and n.minobsinnode = 10.
Boosting Test RMSE: 106.1331
Boosting Test MAE: 68.78423
Boosting Test R²: 0.4232782
```

Figure A1: Hyperparameter tuning: restricted to 300 trees.



```
Stochastic Gradient Boosting

73218 samples
   37 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 58574, 58575, 58574, 58574, 58575
Resampling results across tuning parameters:

  n.trees  RMSE      Rsquared   MAE
  700      94.52755  0.5490751  62.88995
  800      93.97459  0.5542818  62.51814

Tuning parameter 'interaction.depth' was held constant at a value of 15
Tuning parameter 'shrinkage' was held constant at a value of 0.1
Tuning parameter 'n.minobsinnode' was held constant at
 a value of 10
RMSE was used to select the optimal model using  the one SE rule.
The final values used for the model were n.trees = 700, interaction.depth = 15, shrinkage = 0.1 and n.minobsinnode = 10.
Boosting Test RMSE: 101.7006
Boosting Test MAE: 65.97749
Boosting Test R²: 0.4536747
```

Figure A2: Boosting RMSE, MAE and R^2

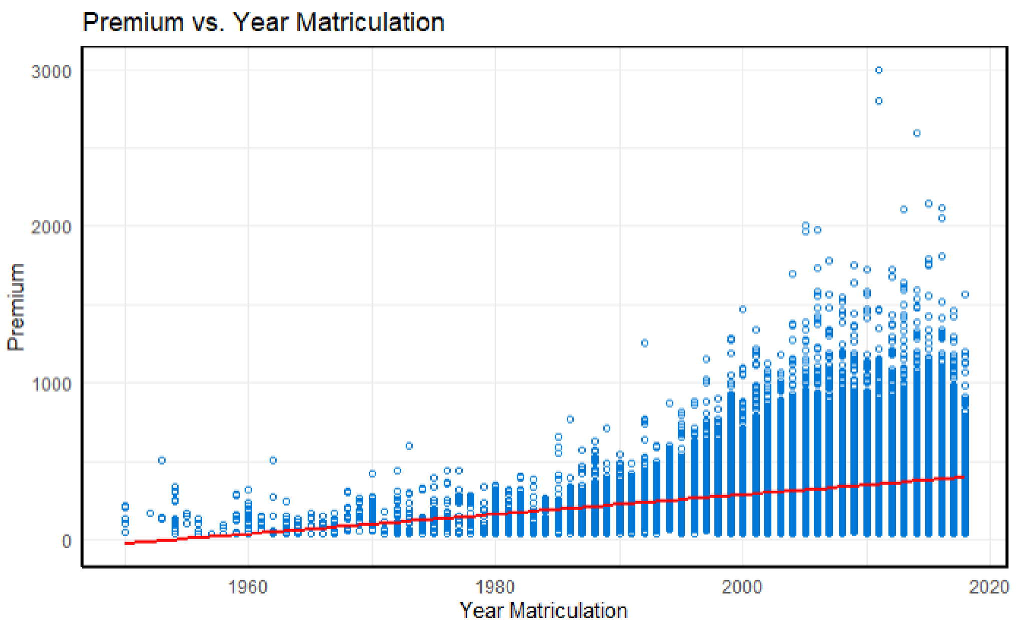


Figure A3: Year matriculation vs Premium

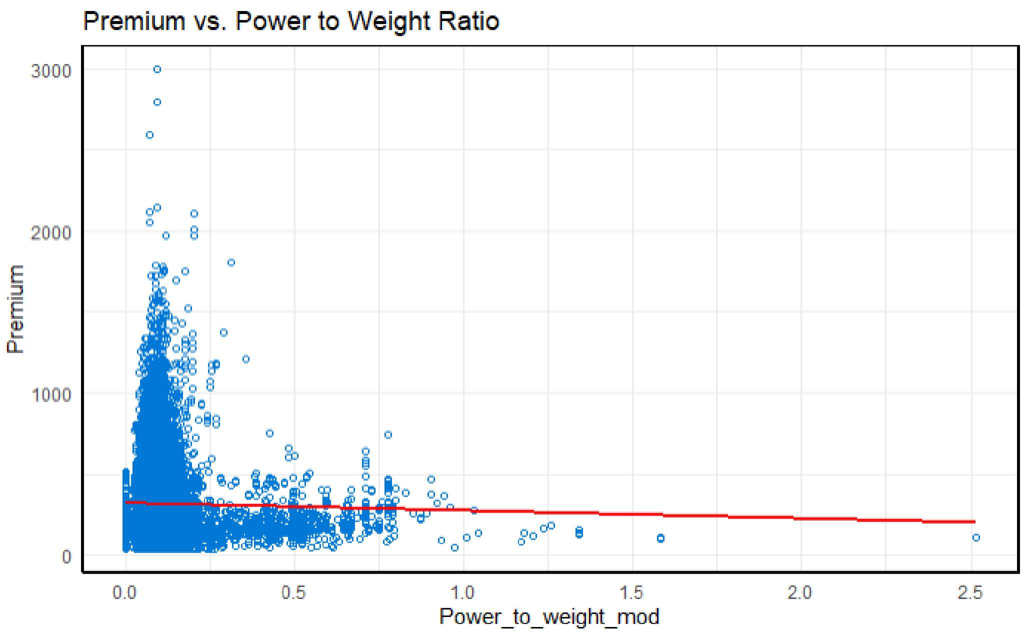
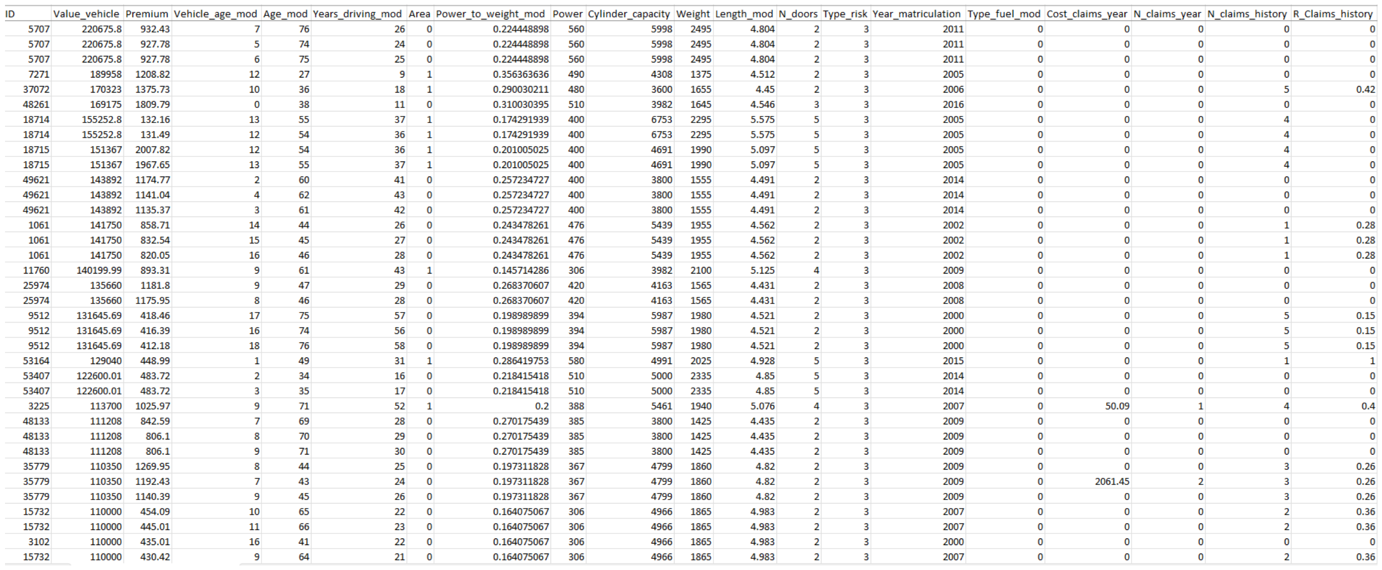
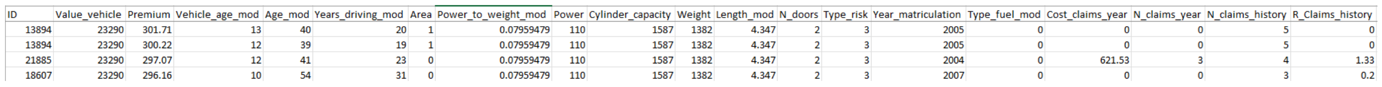


Figure A4: Power to weight vs Premium



ID	Value_vehicle	Premium	Vehicle_age_mod	Age_mod	Years_driving_mod	Area	Power_to_weight_mod	Power	Cylinder_capacity	Weight	Length_mod	N_doors	Type_risk	Year_matriculation	Type_fuel_mod	Cost_claims_year	N_claims_year	N_claims_history	R_Claims_history
5707	220675.8	932.43	7	76	26	0	0.224448898	560	5998	2495	4.804	2	3	2011	0	0	0	0	0
5707	220675.8	927.78	5	74	24	0	0.224448898	560	5998	2495	4.804	2	3	2011	0	0	0	0	0
5707	220675.8	927.78	6	75	25	0	0.224448898	560	5998	2495	4.804	2	3	2011	0	0	0	0	0
7271	189958	1208.82	12	27	9	1	0.356363636	490	4308	1375	4.512	2	3	2005	0	0	0	0	0
37072	170323	1375.73	10	36	18	1	0.290030211	480	3600	1655	4.45	2	3	2006	0	0	0	5	0.42
48261	169175	1809.79	0	38	11	0	0.310030395	510	3982	1645	4.546	3	3	2016	0	0	0	0	0
18714	155252.8	132.16	13	55	37	1	0.174291939	400	6753	2295	5.575	5	3	2005	0	0	0	4	0
18714	155252.8	131.49	12	54	36	1	0.174291939	400	6753	2295	5.575	5	3	2005	0	0	0	4	0
18715	151367	2007.82	12	54	36	1	0.201005025	400	4691	1990	5.097	5	3	2005	0	0	0	4	0
18715	151367	1967.65	13	55	37	1	0.201005025	400	4691	1990	5.097	5	3	2005	0	0	0	4	0
49621	143892	1174.77	2	60	41	0	0.257234727	400	3800	1555	4.491	2	3	2014	0	0	0	0	0
49621	143892	1141.04	4	62	43	0	0.257234727	400	3800	1555	4.491	2	3	2014	0	0	0	0	0
49621	143892	1135.37	3	61	42	0	0.257234727	400	3800	1555	4.491	2	3	2014	0	0	0	0	0
1061	141750	858.71	14	44	26	0	0.243478261	476	5439	1955	4.562	2	3	2002	0	0	0	1	0.28
1061	141750	832.54	15	45	27	0	0.243478261	476	5439	1955	4.562	2	3	2002	0	0	0	1	0.28
1061	141750	820.05	16	46	28	0	0.243478261	476	5439	1955	4.562	2	3	2002	0	0	0	1	0.28
11760	140199.99	893.31	9	61	43	1	0.145714286	306	3982	2100	5.125	4	3	2009	0	0	0	0	0
25974	135660	1181.8	9	47	29	0	0.268370607	420	4163	1565	4.431	2	3	2008	0	0	0	0	0
25974	135660	1175.95	8	46	28	0	0.268370607	420	4163	1565	4.431	2	3	2008	0	0	0	0	0
9512	131645.69	418.46	17	75	57	0	0.198989899	394	5987	1980	4.521	2	3	2000	0	0	0	5	0.15
9512	131645.69	416.39	16	74	56	0	0.198989899	394	5987	1980	4.521	2	3	2000	0	0	0	5	0.15
9512	131645.69	412.18	18	76	58	0	0.198989899	394	5987	1980	4.521	2	3	2000	0	0	0	5	0.15
53164	129040	448.99	1	49	31	1	0.286419753	580	4991	2025	4.928	5	3	2015	0	0	0	1	1
53407	122600.01	483.72	2	34	16	0	0.218415418	510	5000	2335	4.85	5	3	2014	0	0	0	0	0
53407	122600.01	483.72	3	35	17	0	0.218415418	510	5000	2335	4.85	5	3	2014	0	0	0	0	0
3225	113700	1025.97	9	71	52	1	0.2	388	5461	1940	5.076	4	3	2007	0	50.09	1	4	0.4
48133	111208	842.59	7	69	28	0	0.270175439	385	3800	1425	4.435	2	3	2009	0	0	0	0	0
48133	111208	806.1	8	70	29	0	0.270175439	385	3800	1425	4.435	2	3	2009	0	0	0	0	0
48133	111208	806.1	9	71	30	0	0.270175439	385	3800	1425	4.435	2	3	2009	0	0	0	0	0
35779	110350	1269.95	8	44	25	0	0.197311828	367	4799	1860	4.82	2	3	2009	0	0	0	3	0.26
35779	110350	1192.43	7	43	24	0	0.197311828	367	4799	1860	4.82	2	3	2009	0	2061.45	2	3	0.26
35779	110350	1140.39	9	45	26	0	0.197311828	367	4799	1860	4.82	2	3	2009	0	0	0	3	0.26
15732	110000	454.09	10	65	22	0	0.164075067	306	4966	1865	4.983	2	3	2007	0	0	0	2	0.36
15732	110000	445.01	11	66	23	0	0.164075067	306	4966	1865	4.983	2	3	2007	0	0	0	2	0.36
3102	110000	435.01	16	41	22	0	0.164075067	306	4966	1865	4.983	2	3	2000	0	0	0	0	0
15732	110000	430.42	9	64	21	0	0.164075067	306	4966	1865	4.983	2	3	2007	0	0	0	2	0.36

Figure A5: Example of inconsistent observations of `Premium` with respect to `Value_vehicle` across similar vehicles and policy holders



ID	Value_vehicle	Premium	Vehicle_age_mod	Age_mod	Years_driving_mod	Area	Power_to_weight_mod	Power	Cylinder_capacity	Weight	Length_mod	N_doors	Type_risk	Year_matriculation	Type_fuel_mod	Cost_claims_year	N_claims_year	N_claims_history	R_Claims_history
13894	23290	301.71	13	40	20	1	0.07959479	110	1587	1382	4.347	2	3	2005	0	0	0	5	0
13894	23290	300.22	12	39	19	1	0.07959479	110	1587	1382	4.347	2	3	2005	0	0	0	5	0
21885	23290	297.07	12	41	23	0	0.07959479	110	1587	1382	4.347	2	3	2004	0	621.53	3	4	1.33
18607	23290	296.16	10	54	31	0	0.07959479	110	1587	1382	4.347	2	3	2007	0	0	0	3	0.2

Figure A6: Example of observations of the same car with different vehicle age from same/different policy holders yet all have the same value for `Value_vehicle`