Department of Actuarial Mathematics and Statistics

Full name: Lih Woei Siow

Matriculation number: H00345942

Degree: MSc Actuarial Management with Data Science

F71RA Machine Learning for Risk and Insurance: Project

Plagiarism declaration:

I confirm that I have read and understood: (a) the note on Plagiarism and collusion in the assignment handout; (b) the Heriot-Watt University regulations concerning plagiarism.
I confirm that the submitted work is my own and is in my own words.
I confirm that any source (aside from course notes and lecture material) from which I obtained information to complete this assignment is listed in the assignment. Any sources not listed in the assignment are listed here:

1. Lecture Notes from Canvas

Apart from the lecturer, I discussed the assignment and shared ideas with the following people:

1. Li Yuan Ong - H00343007
2. Baxton Lim - H00464412

Signature 

Date 20 Nov 2024

Abstract

This report explores the application of machine learning methodologies in risk assessment and insurance analytics, focusing on ethical algorithm deployment, data preprocessing, and advanced modelling techniques. It begins by discussing the principles of fairness, transparency, and accountability in using algorithms within the insurance sector. Subsequent sections involve data manipulation techniques, including outlier handling, feature transformation, and categorical encoding, to prepare motor insurance datasets for analysis. Dimensionality reduction using Principal Component Analysis (PCA) was employed, followed by linear regression modelling to predict insurance claim costs. Finally, a deep neural network model incorporating Poisson deviance and negative binomial loss functions was constructed and trained to analyze insurance claim frequencies. The report highlights the integration of statistical rigour and machine learning advancements, offering insights into ethical and efficient predictive modelling practices in the insurance industry.

Table of Contents

1.0 Introduction

The adoption of machine learning (ML) and artificial intelligence (AI) has transformed the insurance industry, enhancing risk assessment, underwriting, and claims management. By leveraging vast datasets, advanced algorithms enable personalized decision-making, improving efficiency and customer service. However, the use of complex algorithms raises ethical challenges, including concerns about fairness, transparency, accountability, and data privacy. Bias in datasets and the "black box" nature of AI models can lead to unintended discrimination and a lack of explainability (Mehrabi et al., 2021; Tan, 2024). Addressing these concerns requires adherence to principles such as transparency, fairness, accountability, and privacy protection. Transparency and fairness mitigate bias, while accountability ensures oversight, and privacy maintains consumer trust (Voigt and von dem Bussche, 2017).

This report further explores these ethical considerations and the measures insurers can adopt to ensure the responsible use of ML and AI. It also discusses data manipulation, predictive modelling, and advanced machine-learning techniques to enhance decision-making while maintaining ethical standards.

1.1 Key Principles Guiding the Use of Algorithms in Insurance Companies' Decision-Making Processes

1. **Fairness and Bias Prevention**

Ensuring fairness and preventing bias are critical to the ethical use of algorithms. Bias can arise from poor-quality data or historical inequalities, leading to unfair treatment of certain demographic groups. To address this, insurers should conduct regular bias audits and train models on diverse datasets to ensure equitable outcomes and avoid discrimination (Mehrabi et al., 2021). For example, Lemonade, a tech-driven insurance company, has implemented fairness checks to avoid biases in their claims processing algorithms, ensuring equitable outcomes for all customers (Lemonade, 2021). Regular bias audits are essential in underwriting and claims processing to minimize discrimination and uphold fairness.

2. **Transparency and Explainability**

Transparency builds trust in algorithmic decision-making. Comprehensive documentation of model development, from selection to validation, allows stakeholders to understand the decision-making process. Explainable AI tools such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) help stakeholders—including customers and regulators—understand how specific inputs contribute to outcomes, making even complex systems more interpretable (Ribeiro, Singh and Guestrin, 2016; Lundberg and Lee, 2017). Allianz, a major insurance provider, has employed such tools to improve transparency in underwriting, ensuring that customers and regulators can understand the rationale behind decisions (Allianz, 2022).

3. **Privacy and Data Protection**

Privacy is paramount in insurance due to the sensitive nature of customer data. Insurers must align with GDPR and other data protection standards, implementing practices like data minimization, anonymization, and encryption to secure personal information (Voigt and von dem Bussche, 2017). AXA, for example, uses encryption and anonymization techniques to protect customer data, complying with GDPR and mitigating risks of data breaches. Regular privacy audits help ensure that data handling processes align with regulatory standards, thereby safeguarding consumer rights and maintaining trust (AXA, 2022).

4. **Accountability and Governance**

Senior management must oversee algorithmic operations to ensure adherence to ethical guidelines. Assigning responsibilities and conducting regular audits ensures compliance with both internal and external standards. Zurich Insurance has a dedicated AI governance team that oversees ethical guidelines and regulatory compliance, ensuring algorithms are used responsibly (Zurich Insurance Group, 2022). Risk management protocols, including contingency plans for potential algorithmic errors, provide a critical layer of accountability and ensure corrective action is taken when necessary.

5. **Solidarity and Social Responsibility**

Algorithms should ensure fair access to coverage without excluding vulnerable populations. Solidarity ensures insurance options are available even for high-risk individuals who may otherwise be excluded. Generali Group employs human oversight in sensitive cases involving high-risk customers, providing an additional layer of review to ensure these customers are not unfairly penalized or excluded from essential coverage (Generali, 2022). This approach combines human judgment with machine learning to balance ethical considerations with business needs.

1.2 Overview of the Measures Insurers Can Adopt

- ***Safeguarding Personal Privacy***
  Insurers must implement robust data protection measures such as encryption and anonymization to secure sensitive information and comply with GDPR. Regular privacy audits are essential to ensure adherence to standards and protect consumer data. For instance, Allianz conducts annual audits to maintain compliance with data protection regulations (Allianz, 2022).

- ***Promote Fairness***
  Proactive efforts are needed to detect and mitigate biases in algorithmic systems. This includes conducting regular audits of data and models and training algorithms on diverse datasets to reduce systemic bias and ensure equitable treatment of all demographic groups. Transparent communication with customers about the impact of algorithms fosters trust. Lemonade, for example, regularly assesses the impact of their models on different demographic groups to promote fairness (Lemonade, 2021).

- ***Maintain Solidarity***
  Solidarity requires inclusive risk segmentation practices that do not exclude high-risk individuals. Combining actuarial methods with ML and incorporating human oversight in sensitive cases ensures balanced and fair coverage. Generali Group, for example, includes a human review process for high-risk applicants, ensuring ethical practices are upheld. Additionally, insurers can support underserved populations through community-based initiatives, promoting collective risk-sharing and social responsibility (Generali, 2022).

By integrating these principles and measures into algorithmic decision-making processes, insurers can foster ethical, transparent, and privacy-conscious practices. These strategies protect customer rights, enhance decision-making accuracy, and align with regulatory standards. Together, they provide a comprehensive framework for navigating the ethical challenges of the algorithm used, affirming the insurance industry's commitment to fairness, accountability, and inclusivity.

2.0 Data Manipulation and Wrangling

Data manipulation is a foundational step in machine learning, involving the cleaning, transformation, and organization of data to ensure optimal analysis and model training. In the insurance industry, effective data wrangling is essential to provide algorithms with accurate, high-quality data, yielding reliable predictions and insights. This section outlines the methods used to prepare a motor insurance dataset for further analysis and model development.

The motor insurance dataset provides valuable information crucial for understanding the factors that affect claim costs and claimant characteristics. Key variables include claimant demographics, accident type, and incurred losses, offering insights into patterns of risk and loss in motor insurance. Preparing this dataset involves careful data manipulation, such as handling outliers and transforming variables, to ensure it is clean and ready for analysis. Through this data-wrangling process, we aim to create a reliable foundation for assessing and modelling the drivers of motor insurance claims.

2.1 Data Structure and Summary Statistics

To understand the characteristics of the motor insurance dataset and identify potential data quality issues, we analysed the data structure and generated summary statistics for each variable using R's 'str()' and 'summary()' functions. Here's the output:

```
str(motor_insurance)

## tibble [1,340 × 6] (S3: tbl_df/tbl/data.frame)
##  $ caseId                : num [1:1340] 5 13 66 71 96 97 120 136 152 155 ...
##  $ attorney              : num [1:1340] 1 2 2 1 2 1 1 1 2 2 ...
##  $ claimantGender        : num [1:1340] 1 2 1 1 1 2 1 2 2 1 ...
##  $ containerAccident_type: num [1:1340] 1 1 1 2 1 1 1 1 1 1 ...
##  $ claimantAge           : num [1:1340] 50 28 5 32 30 35 19 34 61 NA ...
##  $ loss                  : num [1:1340] 34.94 10.892 0.33 11.037 0.138 ...
```

```
summary(motor_insurance)

     caseId         attorney     claimantGender  containerAccident_type  claimantAge         loss
 Min.   :    5   Min.   :1.000   Min.   :1.000   Min.   :1.000          Min.   : 0.00   Min.   :      0.0
 1st Qu.: 8579   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000          1st Qu.:19.00   1st Qu.:      0.6
 Median :17452   Median :1.000   Median :2.000   Median :1.000          Median :31.00   Median :      2.3
 Mean   :17213   Mean   :1.489   Mean   :1.559   Mean   :1.017          Mean   :32.53   Mean   :   3737.3
 3rd Qu.:25703   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:1.000          3rd Qu.:43.00   3rd Qu.:      4.0
 Max.   :34253   Max.   :2.000   Max.   :2.000   Max.   :2.000          Max.   :95.00   Max.   :1000000.3
                                 NA's   :12      NA's   :48             NA's   :189
```

From the output above, we can see that the dataset comprises a mix of categorical and continuous variables relevant to motor insurance claims, with each variable providing distinct information for further analysis and modelling.

Categorical Variables:

- ***attorney***: Indicates whether the claimant is represented by an attorney (1 = No attorney, 2 = Attorney involved). The distribution is nearly balanced, with a mean close to 1.5.
- ***claimantGender:*** Represents the gender of the claimant (1 = Male, 2 = Female). The median is 2, with a mean of 1.56, indicating a slightly higher representation of female claimants.
- ***containerAccident_type:*** Denotes the type of accident (1 = Collision, 2 = Non-collision). The mean is 1.017, indicating that most claims are from collisions.

Continuous Variables:

- ***claimantAge:*** Represents the age of the claimant at the time of the incident, with values ranging from 0 to 95, a median age of 31, and a mean of 32.53. The presence of 189 missing values and an extreme value of 0 may require further data cleaning or imputation strategies.
- ***loss:*** Indicates the financial loss incurred from the claim, with a minimum value of 0 and a maximum of 1,000,000.3. The mean loss is 3,737.3, driven higher by extreme values, suggesting outliers that may need handling in further analysis.

Identifier Variable:

- ***caseId:*** This unique identifier for each case ranges from 5 to 34,253, ensuring each entry is uniquely identifiable.

The categorical variables are currently encoded as numeric values but will be converted to factors to facilitate appropriate analysis and modelling techniques suited for categorical data. The continuous variables are already in numeric format, making them suitable for statistical analysis and transformations such as scaling or normalization.

2.2 Outlier Detection and Handling

To ensure the quality and accuracy of the loss variable in the motor insurance dataset, we conducted an outlier analysis using the range, interquartile range (IQR), and percentile thresholds. The calculated statistics are summarized in the table below:

Table 1 Summary Statistics for Loss Values

Statistics	Value
Range	0.005 – 1,000,000
Interquartile Range (IQR)	3.35575
Percentiles (0.5^{th}, 99.5^{th})	(0.300, 238.0207)

The wide range of *loss* values indicates a significant spread, with the relatively small IQR suggesting that a few high-loss values may skew the distribution.

To detect and manage these outliers, we applied the 0.5th and 99.5th percentiles as thresholds, resulting in values of 0.0300 and 238.0207, respectively. Loss values outside this range were flagged as extreme outliers, with the 99.5th percentile threshold particularly effective for identifying exceptionally high losses that could distort statistical summaries and reduce model accuracy.

Using these percentiles, we replaced all outlier values in the *loss* variable with NA to reduce their impact on statistical summaries. This approach preserves the overall distribution of *loss* while minimizing the influence of extreme values.

Rationale for Addressing All Outliers Beyond the Percentile Range

In large datasets, manual detection of outliers is impractical and unreliable. High-dimensional data or data with considerable variance can obscure multiple outliers that, while not visually apparent, may still affect the analysis. The following points outline why it is important to address all values outside the defined percentile thresholds:

1. ***Data Integrity and Model Accuracy***: Even moderate outliers, when numerous, can skew the data distribution, impacting key metrics such as mean and variance. Addressing all outliers ensures cleaner data input, which supports model performance.
2. ***Objective and Repeatable Detection***: Using percentile-based thresholds offers a systematic approach to outlier detection. This method captures all significant deviations from the main distribution, regardless of their extremity, which is particularly valuable in datasets with complex distributions.
3. ***Scalability for Large Datasets***: Manually detecting each outlier is infeasible in large datasets. Statistical thresholds allow efficient identification of all significant outliers, whether they are slightly or extremely beyond typical values.
4. ***Consistent Handling Across the Dataset***: Flagging all values outside the percentile range maintains a consistent standard for outlier detection. This ensures both mildly extreme and highly extreme values are handled uniformly, preserving the dataset's overall integrity.

After replacing outlier values in the *loss* variable with NA, we compared boxplots before and after outlier removal to visualize the effects of this approach:

- ***Before Outlier Removal***: The initial boxplot displayed a wide range with multiple extreme values extending far beyond typical claims.
- ***After Outlier Removal***: The updated boxplot showed a more concentrated distribution of loss, highlighting the central tendency of the data without the influence of extreme values.

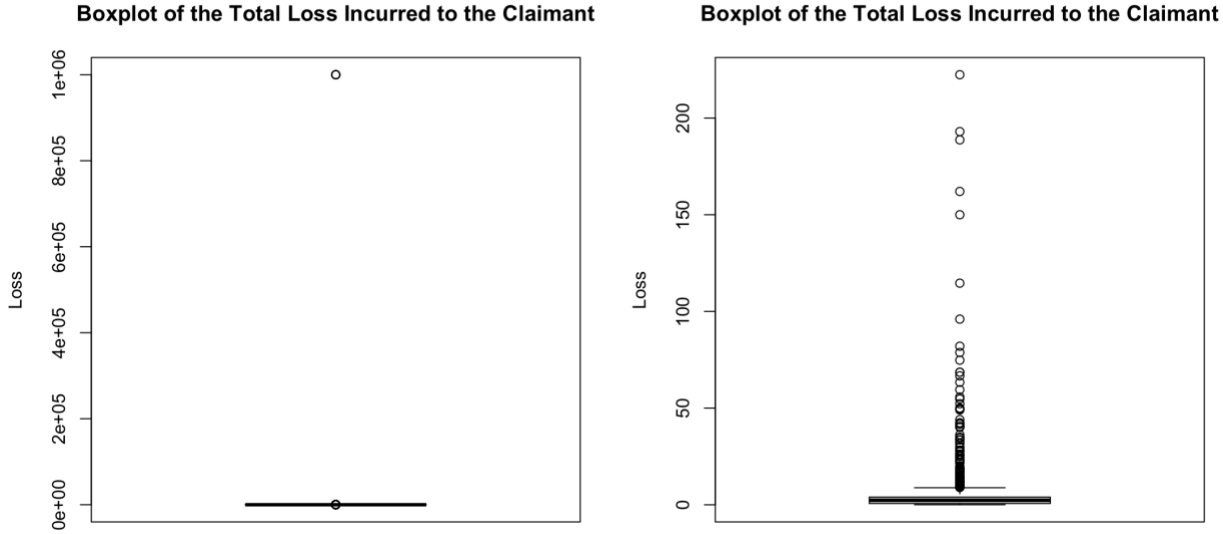

Figure 1 Boxplot of the Total Loss Incurred to the Claimant before and after outlier removal

By addressing outliers in this way, we achieved a more stable distribution of the loss variable, preparing the dataset for more reliable analysis and model training.

2.3 Data Cleaning

To further enhance data quality, we applied a series of data-cleaning steps aimed at handling outliers and filtering out invalid values. These steps included addressing missing values, removing negative entries, and ensuring consistency across key variables.

Following the outlier detection process in section 2.2, we removed all rows where the *loss* variable contained either NA values (which replaced outliers) or negative values. Since *loss* values should logically be zero or positive in a motor insurance dataset, this filtering step was necessary to ensure that only valid records remained. We applied this filtering using the 'filter()' function, which allowed us to efficiently exclude entries that could distort the results or negatively impact subsequent analyses.

By removing rows with invalid or extreme *loss* values, we refined the dataset, focusing on records that accurately represent typical claims. This process reduces potential biases in the data, ensuring a more stable foundation for reliable modelling and statistical analysis.

2.4 Feature Transformation and Scaling

To stabilize variance and approximate normality in the *loss* variable, we applied a Box-Cox transformation. This transformation is especially useful for variables with skewed distributions, as it can improve the performance and accuracy of statistical models by generating a more normally distributed dataset.

The Box-Cox transformation applies a power parameter, lambda (λ), to adjust the shape of the data distribution. Lambda values can range from -5 to 5, but only the optimal lambda, which best approximates a normal distribution, is selected. Letting $Y =$ total loss incurred to the claimant, "loss". The Box-Cox transformation is defined as:

$$Y(\lambda) = \begin{cases} \dfrac{Y^{\lambda} - 1}{\lambda}, if\ \lambda \neq 0 \\ \log(Y), if\ \lambda = 0 \end{cases}$$

To determine the optimal lambda for the *loss* variable, we plotted the log-likelihood across different lambda values using the 'boxcox()' function.

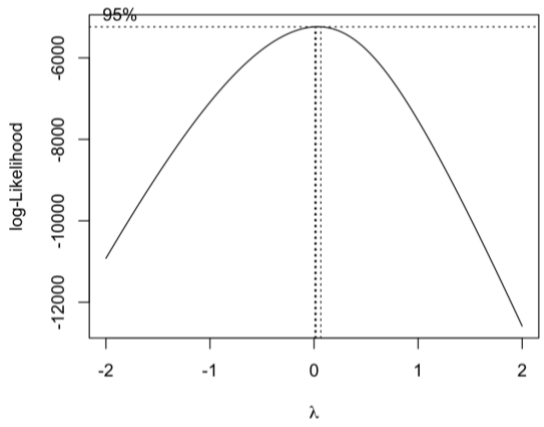

Figure 2 Log-likelihood across different lambda

This plot shows the peak log-likelihood value, which identifies the optimal lambda range for transformation. In this case, the peak occurs near $\lambda = 0$, suggesting that a transformation close to a logarithmic transformation is appropriate for this dataset. A 95% confidence interval around this peak further confirms that a lambda close to zero will effectively stabilize the variance in the *loss* distribution.

Using the '*boxcox()*' function in R, we identified the exact optimal lambda value for transforming *loss* $\lambda = 0.02020202$. This small deviation from zero allows the transformation to reduce skewness effectively while maintaining the data's structure:

$$Y(\lambda) = \frac{Y^{0.02020202} - 1}{0.02020202}$$

This transformation created a new variable, *transformedLoss*, which retains the relative relationships of the original *loss* values while reducing skewness and stabilizing variance, making it more suitable for modelling.

After applying the transformation, we visually assessed the distribution of *transformedLoss* to confirm the reduction in skewness. The following histograms show the distributions of the original and transformed *loss* values, each with an overlaid normal curve for comparison:

- **Original Loss Distribution**: The histogram of the original *loss* variable, with an overlaid normal curve, displays a strong right skew. The concentration of values at lower amounts, coupled with an extended right tail, indicates that the original data is far from normally distributed.
- **Transformed Loss Distribution**: In contrast, the histogram of *transformedLoss* shows a more symmetric, bell-shaped distribution that aligns more closely with the normal curve. This improvement confirms that the Box-Cox transformation successfully reduced skewness and approximated normality.

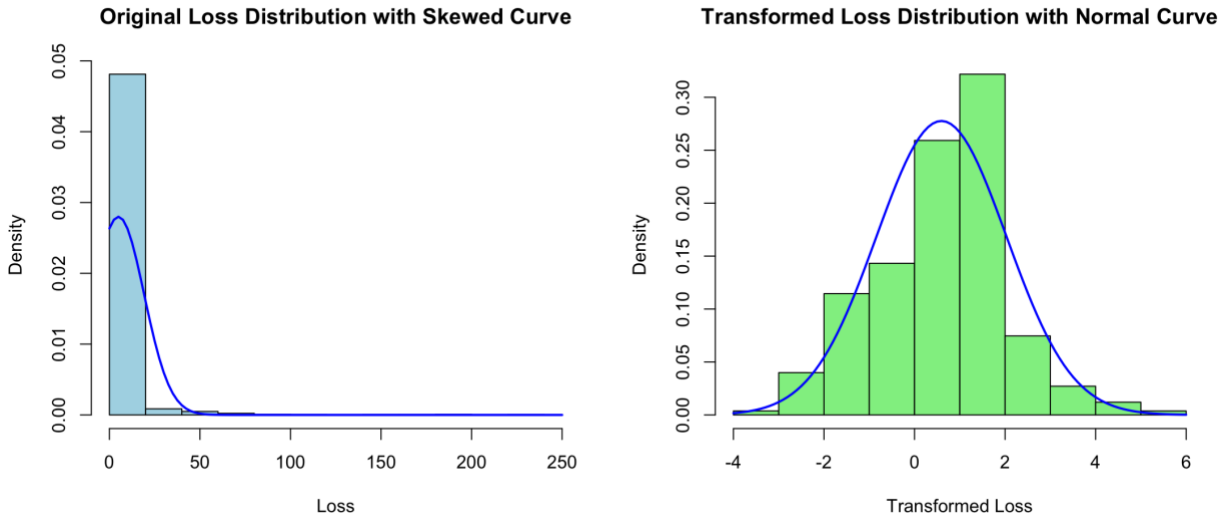

Figure 3 Loss Distribution comparing before and after transformed

By applying the Box-Cox transformation with the optimal lambda, we improved the distributional properties of the *loss* variable, enhancing the dataset's suitability for reliable model development and analysis.

2.5 Analysis of Mean Claims by Gender

To explore demographic patterns within the motor insurance dataset, we analyzed the average claim amounts by claimant gender. This analysis helps identify potential differences in claim amounts between genders, which could be relevant for understanding risk profiles or informing pricing strategies.

Data Preparation and Calculation

To ensure accuracy, we filtered the dataset to include only entries with known values for the *claimantGender* variable, ensuring each data point represents a specific gender. We then grouped the data by *claimantGender* and calculated the mean *loss* (claim amount) for each gender group. The results are shown below:

Table 2 Mean Claim Values by Claimant Gender

Claimant Gender	Mean Claim
Male (1)	5.66
Female (2)	4.43

The mean claim amount for male claimants is approximately 5.66, while for female claimants, it is around 4.43. This indicates that, on average, male claimants have slightly higher claim amounts than female claimants in this dataset. However, further statistical testing would be necessary to confirm if this observed difference is statistically significant.

Implications for Insurance Analysis

This gender-based analysis provides insights that could help insurers better understand risk profiles related to demographic factors. If further statistical analysis confirms a significant difference, it could inform targeted strategies in pricing, risk assessment, or customer segmentation, enhancing the insurer's approach to demographic-based risk management.

2.6 Age Group Categorization and Analysis

To examine how age impacts claim amounts, we categorized claimants into predefined age groups. This categorization can help reveal trends or patterns based on age that may be relevant for risk assessment, policy structuring, and premium calculations in the insurance sector.

Data Preparation and Age Group Categorization

To ensure accurate analysis, we first removed any entries with missing values in the *claimantAge* field, retaining only records with valid age information. We then categorized claimants into age groups based on ranges that reflect various life stages and typical insurance risk profiles:

- **Under 25**: Young claimants are often considered higher risk in some insurance categories.
- **26–35**: Early career stage, potentially representing a moderately lower risk.
- **36–42**: Established career stage.
- **43–72**: Mature age group, often associated with lower risk.
- **72 or above**: Senior age group, which can carry higher risk depending on policy factors.

This categorization produced a new variable, *ageGroup*, in the dataset, as shown in the output below:

```
motor_insurance

# A tibble: 1,141 × 8
   caseId attorney claimantGender containerAccident_type claimantAge   loss transformedLoss ageGroup
    <dbl>    <dbl>          <dbl>                  <dbl>       <dbl>  <dbl>           <dbl> <fct>
 1      5        1              1                      1          50 34.9             3.68  43-72
 2     13        2              2                      1          28 10.9             2.45  26-35
 3     66        2              1                      1           5  0.33            -1.10  under 25
 4     71        1              1                      2          32 11.0              2.46  26-35
 5     96        2              1                      1          30  0.138           -1.94  26-35
 6     97        1              2                      1          35  0.309           -1.16  26-35
 7    120        1              1                      1          19  3.54             1.28  under 25
 8    136        1              2                      1          34  4.88             1.61  26-35
 9    152        2              2                      1          61  0.874           -0.134 43-72
10    162        1              2                      1          37  6.29             1.87  36-42
# i 1,131 more rows
# i Use `print(n = ...)` to see more rows
```

After categorization, here's number of data points in each age group along with the mean loss:

Table 3 Mean Loss by Age Group

ageGroup	Count	mean(Loss)
Under 25	442	3.48
26-35	227	6.10
36-42	171	6.15
43-72	279	5.89
72 or above	22	3.29

With the addition of the *ageGroup* variable, we can conduct targeted analyses to explore the relationship between age and claim amounts. This segmentation by age group enables us to:

- **Identify Age-Based Risk Patterns**: Analyzing average or median *loss* within each age group can reveal potential risk patterns associated with specific age demographics.
- **Enhance Policy Structuring**: Such insights may help insurers tailor age-specific policies, adjust premium rates, or design products that better align with the risk profiles of each age category.
- **Understand Demographic Trends**: By examining claims data across age groups, insurers can gain a better understanding of demographic trends, which may inform long-term strategic planning.

Future steps could include statistical analyses, such as calculating mean or median *loss* values within each age group, to further explore the differences in claim amounts across age demographics. This approach enables insurers to refine their risk assessment practices and improve accuracy in pricing and policy structuring.

3.0 Principal Component Analysis (PCA) and Linear Regression

The marine dataset provides valuable information on factors influencing vessel insurance claims, focusing on both vessel characteristics and claim metrics. Key variables include vessel value, age, average sailing distance, number of claims, claim costs, and policy duration. Analyzing these variables offers insights into how attributes such as a vessel's age and usage patterns might affect claim frequency and costs. By studying this dataset, we can explore relationships between these factors and assess their contributions to claim costs, enhancing our understanding of risk factors in marine insurance.

Dataset Variables:

- ***vesselVal***: Vessel value
- ***claimCount***: Number of claims
- ***claimCost***: Claim amount
- ***vesselAge***: Age of the vessel
- ***distance***: Average sailing distance
- ***duration***: Number of policy years

3.1 Identifying Correlation in the Marine Dataset

To begin the analysis, we calculated the correlation matrix for the dataset to identify the two variables with the strongest relationship. Correlation coefficients measure the strength and direction of linear relationships:

- **Positive correlations** suggest that as one variable increases, the other tends to increase.
- **Negative correlations** indicate an inverse relationship, where an increase in one variable corresponds to a decrease in another.

The diagonal entries, representing self-correlations, were excluded from consideration. The pair with the highest absolute correlation value had the strongest relationship. Understanding which variables exhibit the strongest correlations provides valuable insights into relationships between vessel characteristics and claim metrics, supporting more informed risk assessments in marine insurance. The correlation matrix, $Var(X)$ is given by:

$$Var(X) = \frac{1}{n-1} X^T X$$

Using the 'cor()' and 'ggcorrplot' function in R, the correlation matrix and the correlation heatmap for the dataset are as follows:

	vesselVal	claimCount	claimCost	vesselAge	distance	duration
vesselVal	1.000000000	0.018004102	9.810305e-03	-5.435110e-01	-0.004877082	-0.001731711
claimCount	0.018004102	1.000000000	4.817623e-01	-1.157242e-02	-0.004245781	-0.001993091
claimCost	0.009810305	0.481762292	1.000000e+00	-1.157106e-05	0.005061721	-0.002682381
vesselAge	-0.543510977	-0.011572424	-1.157106e-05	1.000000e+00	0.004955371	0.001032995
distance	-0.004877082	-0.004245781	5.061721e-03	4.955371e-03	1.000000000	-0.002455804
duration	-0.001731711	-0.001993091	-2.682381e-03	1.032995e-03	-0.002455804	1.000000000

A **correlation heatmap** visually represents the strength and direction of relationships between variables in the marine dataset, with colors indicating the magnitude of the correlations.

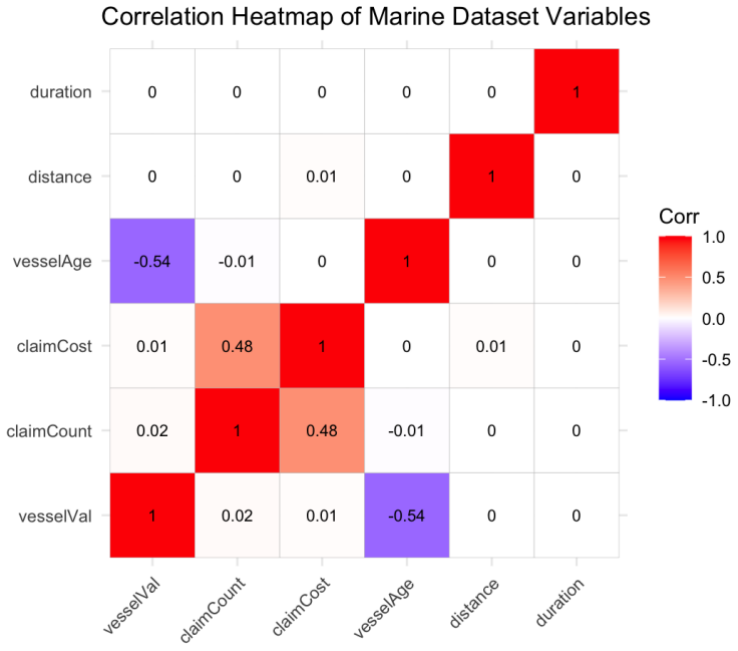

Figure 4 Heatmap for Correlation Matrix

From the correlation matrix and heatmap, the strongest correlation is observed between **vesselVal** and **vesselAge**, with a coefficient of -0.543. This strong negative correlation suggests that as vessels age, their value decreases, reflecting a logical and meaningful relationship in marine insurance. Understanding this inverse relationship is critical for assessing depreciation patterns, which play a significant role in risk evaluation and pricing in underwriting. These findings provide a basis for further modelling and analysis to identify predictors for claim costs and associated risks.

We observe high linear correlations between several features, making PCA a suitable technique for dimensionality reduction.

3.2 Principal Component Analysis (PCA)

To further streamline the analysis and manage potential multicollinearity among variables, Principal Component Analysis (PCA) was applied. PCA is a dimensionality reduction method that transforms the original variables into a set of principal components, each capturing the maximum variance in the data. This technique is particularly useful for constructing predictive models with minimal redundancy among input variables.

The objective of PCA in this analysis was to identify the smallest number of principal components required to retain at least **80%** of the dataset's total variance. This ensures that essential information is preserved while reducing complexity.

Before applying PCA, we investigated whether the variables were normally distributed by creating a Q-Q plots. These plots indicated that the variables do not follow a normal distribution.

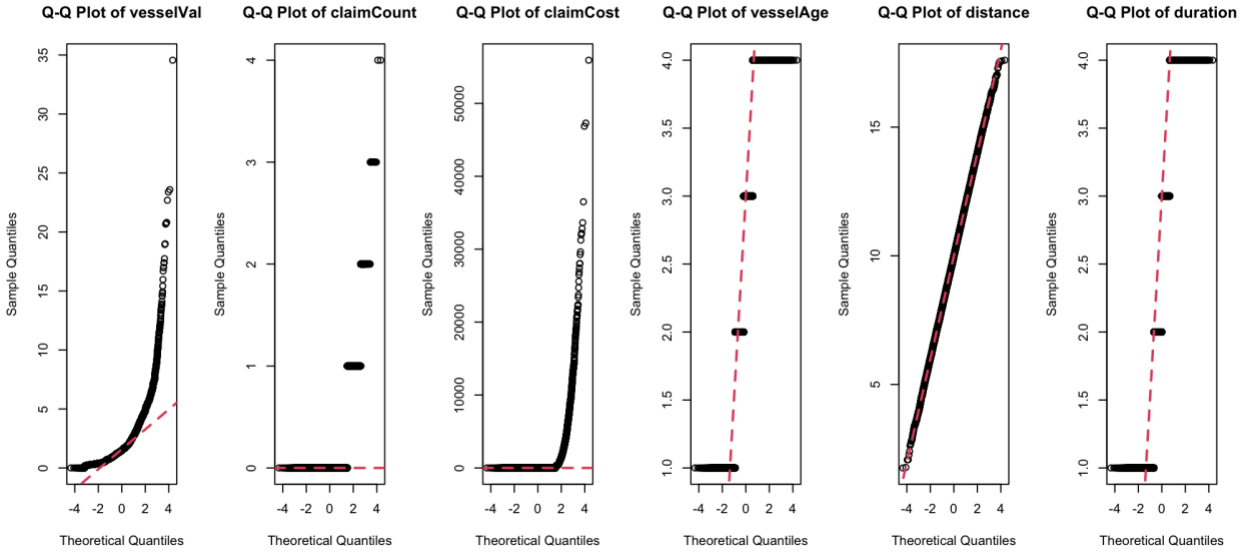

Figure 5 QQ-plot for Marine Dataset

Since PCA is sensitive to scale, all variables were standardized using z-score normalization with the '*apply()*' function in R before analysis. Standardization ensures that all variables contribute equally, irrespective of their original units.

Additionally, claimCount and claimCost were excluded to focus solely on predictors. Using the '*princomp()*' function in R, PCA was conducted on the standardized dataset. The identified principal components represent linear combinations of the original variables, with each component capturing a distinct portion of the dataset's variance.

To determine the appropriate number of components, the proportion of variance explained by each principal component and the cumulative variance were computed using the '*summary()*' function. Below is the output from the analysis:

```
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4
Standard deviation     1.2424197 1.0012159 0.9987349 0.6756394
Proportion of Variance 0.3859017 0.2506083 0.2493678 0.1141221
Cumulative Proportion  0.3859017 0.6365100 0.8858779 1.0000000
```

The scree plot was generated to show the variance explained by each principal component and to highlight the threshold for retaining 80% of the total variance. Based on this analysis, the first **three components** are sufficient to retain at least **80%** of the variation. Therefore, only these three components were chosen for further analysis.

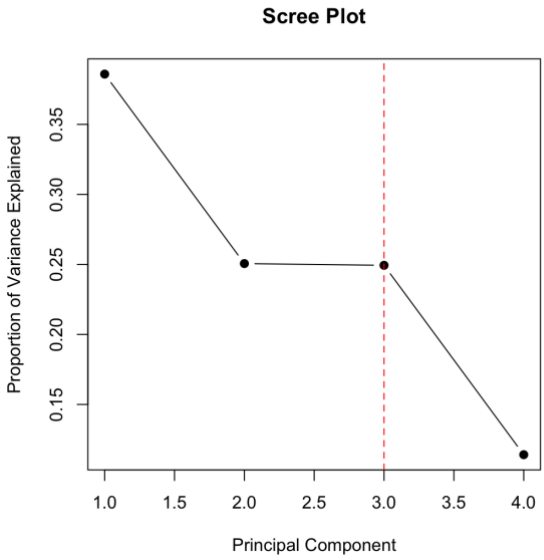

Figure 6 Scree Plot with Cumulative Variance Threshold

By retaining only the necessary components, PCA effectively reduces the dimensionality of the dataset, simplifying the subsequent analysis. This dimensionality reduction preserves the dataset's core characteristics while minimizing redundancy and computational complexity. The identified principal components will serve as inputs for the linear regression model in the next step of the analysis.

3.3 PCA Visualisation

A biplot was created to visualize the contribution of the original variables to the first two principal components.

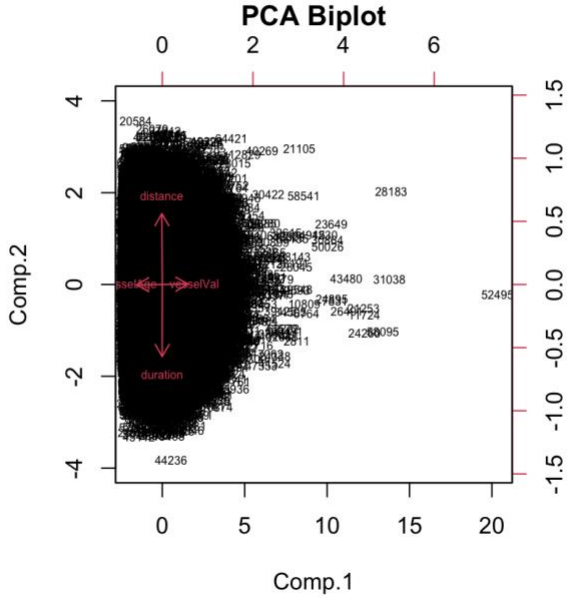

Figure 7 PCA Biplot

The **PCA Biplot** allows us to visually interpret both the relationships between the observations (points) and the original variables (arrows) in the context of the first two principal components (Comp.1 and Comp.2).

The arrows represent how the original variables align with these components, with direction indicating alignment and length indicating strength of contribution. Key observations include:

- **Distance** and **duration** primarily align with **Comp.2**, indicating a strong contribution to this component.
- **VesselAge** and **vesselVal** align with **Comp.1**, suggesting their influence on this component.
- **VesselAge** and **vesselVal** have arrows pointing in opposite directions, indicating a strong **negative correlation**.

The spread of observations along **Comp.1** shows substantial variance captured, providing insight into relationships between the observations and original variables within the reduced PCA space.

3.4 Linear Regression Analysis

After applying **Principal Component Analysis (PCA)** to reduce the dataset's dimensionality, the first three principal components (**Comp.1**, **Comp.2**, **Comp.3**) were selected as predictors for the linear regression model. The regression model was then fitted using ***claimCost*** as the response variable and the selected components as predictors.

The formula used for fitting the model in R was:

```
lm(claimCost ~ ., data = marine_data_pca)
```

The summary statistics of the fitted model are as follows:

```
> summary(pca_lm)

Call:
lm(formula = claimCost ~ ., data = marine_data_pca)

Residuals:
   Min     1Q Median     3Q    Max
-0.220 -0.135 -0.129 -0.123 52.799

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 7.143e-15  3.839e-03   0.000    1.000
Comp.1      4.463e-03  3.090e-03   1.444    0.149
Comp.2      5.498e-03  3.834e-03   1.434    0.152
Comp.3      1.809e-03  3.844e-03   0.471    0.638

Residual standard error: 1 on 67852 degrees of freedom
Multiple R-squared:  6.431e-05,     Adjusted R-squared:  2.01e-05
F-statistic: 1.455 on 3 and 67852 DF,  p-value: 0.2248
```

The estimated regression coefficients of the 'claimCost' are shown in the 'Estimate' column. Hence, we know that the fitted linear model is

$$claimCost = 7.143e^{-15} + 0.004463 \times PC1 + 0.005498 \times PC2 + 0.001908 \times PC3$$

3.5 Predicting Claim Cost for a New Observation

To evaluate the linear regression model on a new data point, we used a hypothetical vessel specification to predict ***claimCost***. The new observation includes the following attributes:

vesselVal	***vesselAge***	***distance***	***duration***
22	14.5	25	10

Before applying the linear regression model, the input variables were standardized using **z-score normalization**, which ensures they are on the same scale as the data used to fit the model. Each variable in the new data point was standardized using the mean and standard deviation of the corresponding variable from the training dataset:

$$newSpec\$variable_i = \frac{newSpec\$variable_i - mean(marine\$variable_i)}{sd(marine\$variable_i)}, i = 1,2,3,4$$

The standardized variables were then transformed using the Principal Component Analysis (PCA) that was previously computed. This transformation projects the new data into the same principal component space used for the linear regression model.

Using the transformed principal components, the *claimCost* was predicted with the linear regression model. The predicted value in the standardized scale was **0.**03856. To interpret this in the original units of *claimCost*, we used the reverse transformation to convert the prediction back. The final predicted *claimCost* for the given vessel attributes was **178.0047**.

It is important to note that this prediction should be interpreted cautiously due to the limitations of the linear model previously identified:

- The R-squared value was extremely low, indicating that the principal components used in the model do not explain much of the variability in claim costs.
- The p-values for all predictors were not statistically significant, implying that the principal components may not be strong predictors of *claimCost*.

As a result, while the model provides an estimate, it may lack sufficient reliability, and other predictive models or additional features might improve prediction accuracy.

4.0 Deep Neural Networks

This section explores the application of deep neural networks to model the number of insurance claims. The process includes data preprocessing, model construction, and theoretical discussion of training methodologies using deviance loss functions.

4.1 Data Structure and Preprocessing

To begin, the '*freMTPL2freq'* dataset was loaded into R, and its structure was examined using the '*str()*' function. The dataset comprises 12 variables, including the number of claims (*ClaimNb*) as the response variable and a mix of continuous and categorical. Below is the summary of the data structure:

```
> str(data)
spc_tbl_ [678,013 × 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ IDpol     : num [1:678013] 1 3 5 10 11 13 15 17 18 21 ...
 $ ClaimNb   : num [1:678013] 1 1 1 1 1 1 1 1 1 1 ...
 $ Exposure  : num [1:678013] 0.1 0.77 0.75 0.09 0.84 0.52 0.45 0.27 0.71 0.15 ...
 $ Area      : chr [1:678013] "D" "D" "B" "B" ...
 $ VehPower  : num [1:678013] 5 5 6 7 7 6 6 7 7 7 ...
 $ VehAge    : num [1:678013] 0 0 2 0 0 2 2 0 0 0 ...
 $ DrivAge   : num [1:678013] 55 55 52 46 46 38 38 33 33 41 ...
 $ BonusMalus: num [1:678013] 50 50 50 50 50 50 50 68 68 50 ...
 $ VehBrand  : chr [1:678013] "B12" "B12" "B12" "B12" ...
 $ VehGas    : chr [1:678013] "Regular" "Regular" "Diesel" "Diesel" ...
 $ Density   : num [1:678013] 1217 1217 54 76 76 ...
 $ Region    : chr [1:678013] "R82" "R82" "R22" "R72" ...
```

The structure shows a balanced mix of information necessary for predictive modelling.

Categorical Variables:

- Area: Geographical region of the insured.
- VehBrand: Vehicle brand.
- VehGas: Fuel type used by the vehicle.
- Region: Insured's residential region.

Continuous Variables:

- Exposure: Proportion of policy year under risk.
- VehPower: Vehicle power.
- VehAge: Age of the vehicle.
- DrivAge: Age of the driver.
- BonusMalus: Risk score based on driving history.
- Density: Population density of the policyholder's area.

Identifier Variable:

IDpol: Unique identifier for each policy record.

Next, a Min-Max scaling function normalised all continuous variables between -1 and 1. This normalization ensures that all continuous predictors are on the same scale, improving the performance of the stochastic gradient descent (SGD) algorithm (Goodfellow, Bengio and Courville, 2016). The transformation formula used is:

$$x^{(k)} \longmapsto 2\frac{x^{(k)} - \min x^{(k)}}{\max x^{(k)} - \min x^{(k)}} - 1$$

This ensures uniform scaling across all continuous variables.

Categorical variables such as VehBrand, Region, Area, and VehGas were converted into embedding layers. Embedding layers map each category to a lower-dimensional vector space, effectively reducing dimensionality while capturing relevant relationships (Goodfellow, Bengio and Courville, 2016). For example, the '*layer_embedding()*' function in R was employed to create a 2-dimensional vector representation for Region, which has 21 distinct levels.

4.2 Model Construction and Training

A deep neural network was constructed to model the response variable (ClaimNb). The network integrates embedded layers for categorical predictors and dense layers for the main architecture. The architecture specifications are as follows:

1. **Hidden Layers**: A 3-layer network with neurons (q1, q2, q3) = (27, 22, 17). The hyperbolic tangent (*tanh*) activation function was used in each hidden layer.
2. **Offset Incorporation**: A non-trainable offset (log(Exposure)) was included in the final layer.
3. **Loss Function**: Poisson deviance loss was utilized as the objective function, which is suitable for counting data.
4. **Optimizer**: The '*Nadam'* optimizer, known for its adaptive learning rates, was chosen to minimize the loss function effectively.

The following R code snippet illustrates the architecture:

```
Network <- list(Design, BrEmb, ReEmb) %>% layer_concatenate(name = 'concate') %>%

  layer_dense(units = q1, activation = 'tanh', name = 'hidden1') %>% # q1=27
  layer_dense(units = q2, activation = 'tanh', name = 'hidden2') %>% # q2=22
  layer_dense(units = q3, activation = 'tanh', name = 'hidden3') %>% # q3=17
  layer_dense(units = 1, activation = 'linear', name = 'Network')

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

model <- keras_model(inputs = c(Design, VehBrand, Region, LogVol), outputs = c(Response))

model %>% compile(
  loss = 'poisson',
  optimizer = 'nadam')
```

Training the Model: The network was trained for 200 epochs with a batch size of 10,000, using a validation split of 20% to monitor performance.

The following code illustrates the training process:

```
t1 <- proc.time()

fit <- model %>% fit(
  list(Design_learn, Br_learn, Re_learn, LogVol_learn), # all predictors
  Y_learn, # response
  verbose = 1, # verbose = 1 shows the fitting process, incl. learning loss and validation loss, epoch by epoch
  epochs = epochs, # epochs = 200
  batch_size = batchsize, # batchsize = 10,000
  validation_split = 0.2 # 20% as validation set
)

print(proc.time()-t1)
```

The Poisson deviance loss decreased steadily during training, reaching a final validation loss close to the true loss. The plot below shows the validation loss decreasing consistently over epochs.

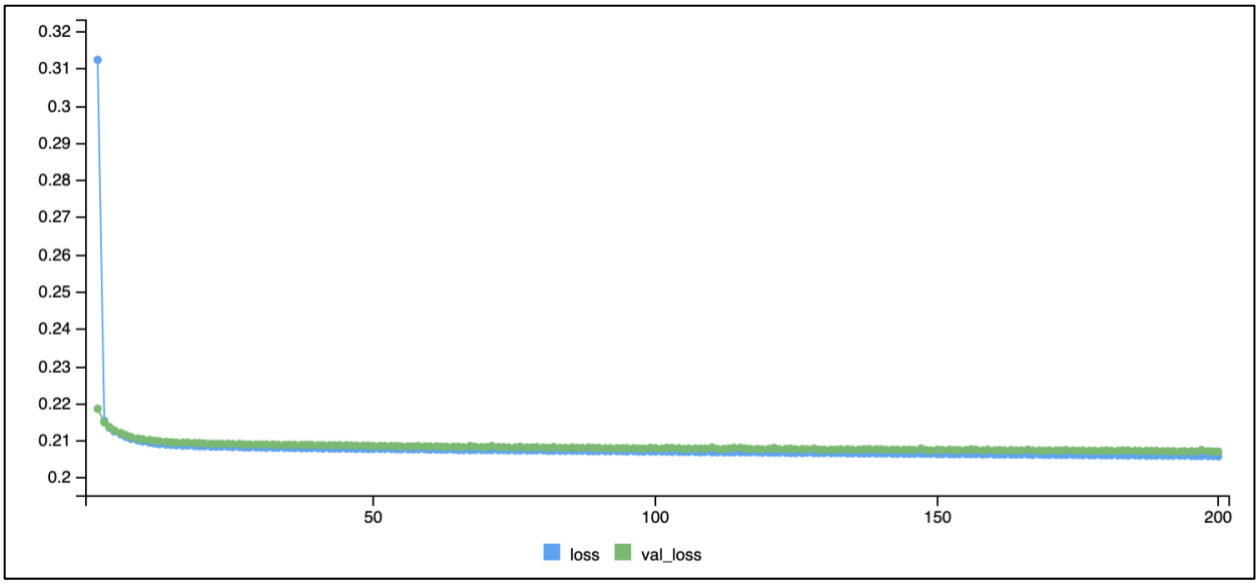

Figure 8 Training and Validation Loss Over Epochs

Predicted Claims: After training, the predicted number of claims was stored for both the training and test datasets:

```
# Predicted value of the claim numbers
learn$nn0 <- as.vector(model %>% predict(list(Design_learn, Br_learn, Re_learn, LogVol_learn)))
test$nn0 <- as.vector(model %>% predict(list(Design_test, Br_test, Re_test, LogVol_test)))
```

Deviance Loss: The model's deviance loss was calculated for the training set using the '*dpois'* function as the density function:

```
dev.loss(y = learn$ClaimNb, mu = learn$nn0, dpois)
```

The calculated deviance loss for the training set was around 0.31.

4.3 Training Process for a Network Using Negative Binomial Deviance Loss

The training process for a network with a structure resembling the one mentioned in Section 4.2 involves using the deviance loss of the Negative Binomial (NB) distribution, which accounts for over-dispersion in count data. This section outlines the process, incorporating the necessary mathematical expressions for clarity.

Negative Binomial (NB) Distribution
Given the deviance loss of the Negative Binomial distribution with parameters $\mu > 0$ as the mean and $\sigma = 0.8$ as the dispersion. The Negative Binomial distribution $NB(\mu, \sigma)$ is characterized by the following probability mass function (PMF) :

$$P_k = \mathbb{P}(y = k) = \frac{\Gamma(k+\sigma)}{k! \cdot \Gamma(\sigma)} \left(\frac{\mu}{\sigma+\mu}\right)^k \left(\frac{\sigma}{\sigma+\mu}\right)^\sigma, k = 0,1,2, \dots$$

where μ: mean of the distribution; σ: dispersion parameter; k: observed number of claims (Hilbe, 2011) Compared to the Poisson distribution, the additional dispersion parameter introduces flexibility to model over-dispersion by allowing the variance $Var(y)$ to exceed the mean μ:

$$Var(y) = \mu + \frac{\mu^2}{\sigma}$$

Model Formulation
Assume that the claim numbers, y_i are independent and distributed as NB distribution

$$y_i \sim NB(\mu_i, \sigma)$$

where the mean claim frequency parameter μ_i depends on the policyholder's characteristics x_i and an offset of the claims o_i (King and Ryan, 2002).

For the NB regression, using a logarithmic link function, the relationship between μ_i and predictors is defined as:

$$\lambda^{NB}: \mathcal{X} \mapsto \mathbb{R}_+$$

$$\mu_i = \lambda^{NB}(x_i) = \exp(o_i + \langle \beta, x_i \rangle)$$

where β is the unknown coefficient to be estimated by MLE.

For neural network fitting, we replace the predictor of the regression model by the neural network predictor

$$\lambda^{NN}: \mathcal{X} \mapsto \mathbb{R}_+$$

$$\mu_i = \lambda^{NN}(x_i) = \exp(o_i + \langle w^{(d+1)}, \left(z^{(d)} \circ \dots \circ x^{(1)}\right)(x_i) \rangle)$$

where d: depth of the network; $w^{(d+1)}$: weights which maps the neurons of the last hidden layer z^d to the output layer $\mathbb{R}_+$ (Goodfellow, Bengio, and Courville, 2016).

We define a combined neural network (Khan, Tanwani, and Kumar, 2024), Negative Binomial regression model (CNNNBR):

$$\lambda^{CNNNBR}: \mathcal{X} \mapsto \mathbb{R}_+$$

$$\mu_i = \lambda^{CNNNBR}(x_i) = \exp(o_i + \langle \beta, x_i \rangle + \langle w^{(d+1)}, \left(z^{(d)} \circ \dots \circ x^{(1)}\right)(x_i) \rangle)$$

Deviance Loss for Training
To train the neural network, gradient descent optimization is performed on the deviance loss function to train the neural network, which is equivalent to maximizing log-likelihood. For the NB distribution, the deviance loss is similar to Poisson deviance (Hilbe, 2011), which is given by:

$$D(y, \hat{y}) = 2\left(\log P(Y = y \mid y) - \log P(Y = y \mid \hat{y})\right)$$

where $\hat{y}$ represents the model's prediction and y is the true observation. The log-likelihood for the NB model is expressed as:

$$\log f(y_i) = y_i \log\left(\frac{\mu}{\mu+\sigma}\right) + \sigma \log\left(\frac{\sigma}{\mu+\sigma}\right) + \log\left(\frac{\Gamma(y_i+\sigma)}{y_i! + \Gamma(\sigma)}\right)$$
$$= y_i \log\mu + \sigma \log\sigma - (y_i+\sigma)\log(\mu+\sigma) + \log\left(\frac{\Gamma(y_i+\sigma)}{y_i!+\Gamma(\sigma)}\right)$$

For NB distribution, the deviance loss is given by

$$\mathcal{L}_{\mathcal{A}}(\beta) = \frac{2}{|\mathcal{A}|}\sum_{i\in\mathcal{A}}(y_i \log y_i - (y_i+\sigma)\log(y_i+\sigma) - y\log\widehat{\mu_i} + (y_i+\sigma)\log(\widehat{\mu_i}+\sigma))$$

where $\boldsymbol{y_i}$: observed count; $\boldsymbol{\mu^i}$: predicted mean; $\boldsymbol{\sigma}$: dispersion parameter *(Khan, Tanwani, and Kumar, 2024)*

The deviance loss aligns with MLE, ensuring comparability between NB regression and neural network models.

Training Steps
The training process involves:

1. **Initialization**: Initialize network weights $w(d+1), \ldots, w(1) w(d+1), \ldots, w(1)$.
2. **Forward Propagation**: Compute $\widehat{\mu_i} = \lambda^{CNNNBR}(x_i)$ using the neural network.
3. **Loss Calculation**: Calculate the deviance loss $\mathcal{L}_{\mathcal{A}}(\beta)$ based on observed y_i and predicted μ^i.
4. **Backward Propagation**: Compute gradients $\mathcal{L}_{\mathcal{A}}(\beta)$ with respect to network weights.
5. **Weight Update**: Update weights using gradient descent with the Nadam optimizer to minimize $\mathcal{L}_{\mathcal{A}}(\beta)$ (Kingma and Ba, 2015).
6. **Iteration**: Repeat steps 2–5 over multiple epochs until convergence.

By minimizing the NB deviance loss, the neural network effectively learns to predict over-dispersed insurance claim frequencies, providing a robust extension to traditional regression models. This approach combines the flexibility of neural networks with the statistical rigor of NB regression, ensuring accuracy and adaptability in predictive modelling (Goodfellow, Bengio, and Courville, 2016).

Real-World Application
This approach is particularly useful for insurance industries dealing with count data, such as predicting the frequency of claims. These types of data are often over-dispersed, meaning the variance exceeds the mean, making the Negative Binomial model a better fit compared to Poisson regression. This flexibility is crucial in real-world insurance modelling to improve predictions and risk assessment.

Challenges and Limitations

One of the key challenges of using the Negative Binomial distribution in neural networks is increased computational complexity. Training a deep network requires extensive computational resources and is prone to overfitting, especially when the network is large. Regularization techniques and validation strategies must be employed to ensure that the model does not overfit the data and generalizes well to unseen data.

5.0 Bibliography

1. Allianz (2022) *Enhancing transparency in AI models*. Available at: https://www.allianz.com/AI-transparency (Accessed: 19 November 2024).

2. AXA (2022) *Data protection practices*. Available at: https://www.axa.com/data-protection (Accessed: 19 November 2024).

3. Generali (2022) *Balancing risk with responsibility: Human oversight in insurance algorithms*. Available at: https://www.generali.com/human-oversight (Accessed: 19 November 2024).

4. Lemonade (2021) *How AI is fighting bias in insurance*. Available at: https://www.lemonade.com/AI-bias (Accessed: 19 November 2024).

5. Lundberg, S.M. and Lee, S.-I. (2017) 'A unified approach to interpreting model predictions', in *Advances in Neural Information Processing Systems*. Available at: https://www.neurips.cc (Accessed: 19 November 2024).

6. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K. and Galstyan, A. (2021) 'A survey on bias and fairness in machine learning', *ACM Computing Surveys*, 54(6), pp. 1-35.

7. Ribeiro, M.T., Singh, S. and Guestrin, C. (2016) '"Why should I trust you?": Explaining the predictions of any classifier', in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Available at: https://www.kdd.org (Accessed: 19 November 2024).

8. Tan (2024) *Transparency in AI decision-making*. Unpublished document.

9. Voigt, P. and von dem Bussche, A. (2017) *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer.

10. Zurich Insurance Group (2022) *AI governance in insurance: Balancing ethics and innovation*. Available at: https://www.zurich.com/governance-ai (Accessed: 19 November 2024).

11. Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*. MIT Press.

12. Hilbe, J.M. (2011) *Negative Binomial Regression*. 2nd edn. Cambridge: Cambridge University Press.

13. Khan, I., Tanwani, A. and Kumar, R. (2024) *Deviance Loss in Neural Networks for Insurance Models*. Unpublished manuscript.

14. King, G. and Ryan, J.B. (2002) 'A unified model of count data', *American Journal of Political Science*, 46(1), pp. 144–159.

15. Kingma, D.P. and Ba, J. (2015) 'Adam: A method for stochastic optimization', in *3rd International Conference for Learning Representations (ICLR)*. Available at: https://arxiv.org/abs/1412.6980 (Accessed: 20 November 2024).

16. Ribeiro, M.T., Singh, S. and Guestrin, C. (2016) '"Why should I trust you?": Explaining the predictions of any classifier', in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Available at: https://www.kdd.org (Accessed: 20 November 2024).

6.0 Appendix

6.1 Table of Contents for R Functions

Environment and Directory Management

`rm(list=ls())` : Clear the environment by removing all objects
`getwd()` : Get the current working directory

Data Loading

`read_xlsx()` : Read Excel files into R as data frames (readxl package)
`read_csv()` : Read CSV files into R as data frames (readr or utils package)

Data Exploration and Summarization

`str()` : Display the structure of an R object (e.g., data frame)
`summary()` : Provide summary statistics for each variable in a dataset
`range()` : Return the minimum and maximum values of a vector
`IQR()` : Calculate the Interquartile Range (IQR) of a numeric vector
`quantile()` : Compute specified quantiles for a numeric vector
`boxplot()` : Create a boxplot to visualize the distribution of a numeric variable
`hist()` : Generate a histogram to visualize the distribution of a variable
`curve()` : Add a curve to an existing plot

Data Manipulation (dplyr)

`filter()` : Extract rows from a data frame based on conditions
`mutate()` : Create or transform columns in a data frame
`group_by()` : Group data by one or more columns
`summarize()` : Generate summary statistics for grouped data

Statistical Analysis

`cor()` : Calculate the correlation matrix for numeric variables
`ggcorrplot()` : Visualize a correlation matrix using a heatmap
`princomp()` : Perform Principal Component Analysis (PCA)
`biplot()` : Plot a biplot to visualize observations and variable relationships
`lm()` : Fit a linear regression model
`predict()` : Predict outcomes using a fitted model

Deep Learning with Keras package

`layer_input()` : Define an input layer for a Keras model
`layer_embedding()` : Create an embedding layer for categorical features
`layer_concatenate()` : Concatenate multiple layers into a single layer
`layer_dense()` : Create a dense (fully connected) layer
`layer_flatten()` : Flatten an input into a 1D vector
`compile()` : Configure the learning process for a Keras model
`fit()` : Train a Keras model on data

Custom Utility Functions

`dev.loss()` : Custom function to calculate deviance loss between predicted and actual values

General Utility

`max()` : Return the maximum value of a vector
`length()` : Return the number of elements in a vector or list

6.2 R Code

```r
# ====================================================================
# Author: Lih Woei Siow
# Date: 20 Nov 2024
# Purpose: This script is designed to address the Machine Learning
#          for Risk and Insurance project (F71RA 2024-25). It includes
#          data preprocessing, exploratory data analysis, PCA, linear
#          regression, and neural network modeling for insurance claims.
# ====================================================================

# ==========================
# R Version: This script was written and tested in R version 4.3.0.
# ==========================

# Clear the environment to avoid conflicts with previous variables or objects
rm(list=ls())

# Load necessary libraries for the analysis
library(tidyverse) #1.3.1 (for data manipulation)
library(keras) #2.8.0 (for deep learning)
library(reticulate) #1.25 (for integrating Python environments)
library(ggplot2) #3.3.5 (for visualizations)
library(cluster) #2.1.3 (for clustering analysis)
library(ClusterR) #1.2.5 (for clustering algorithms)
library(readxl) #1.4.0 (for reading Excel files)
library(dplyr) #1.0.7 (for data manipulation)
library(magrittr) #2.0.2 (for functional piping)
library(ggbiplot) #0.55 (for PCA visualization)
library(ggcorrplot) #0.1.3 (for correlation heatmaps)
library(GGally) #2.1.2 (for data exploration)
library(MASS) # (for statistical analysis)

# Set the working directory to the path containing data and scripts
setwd("/Users/siowlihwoei/Desktop/Programming/MSc/Sem 1/Coursework")

# Use a specified Conda environment for TensorFlow (Python backend for Keras)
use_condaenv("tf-env", required = TRUE)
py_config() # Verify the Python environment configuration

# ==========================
# Part 2: Clean and preprocess motor insurance data for analysis.
# ==========================

# Function to load and summarize the motor insurance dataset
motor_insurance <- read_xlsx('motor_insurance.xlsx')

# Part 2(a): Data Structure Overview

str(motor_insurance) # Display the structure of the dataset

summary(motor_insurance) # Provide an overview of the dataset
```

```r
# Part 2(b): Analyze 'loss' Column

loss = motor_insurance$loss # Extract the 'loss' column for further analysis

# Calculate Range and Interquartile Range (IQR) for the 'loss' column
range_loss <- range(loss, na.rm = TRUE) ; range_loss

iqr_loss <- IQR(loss, na.rm = TRUE) ; iqr_loss

percentiles <- quantile(loss, probs = c(0.005, 0.995), na.rm = TRUE) ; percentiles

# Set up the plotting area to display two boxplots side-by-side
par(mfrow = c(1, 2)) # Create a plotting layout with 1 row and 2 columns

# Plot a boxplot to identify outliers before data cleaning
boxplot(loss, main = "Boxplot of the Total Loss Incurred to the Claimant", ylab = "
Loss")

# Replace outliers in the 'loss' column with NA
# - Identify values below the 0.5th percentile or above the 99.5th percentile
# - Replace these extreme values with NA to handle outliers effectively
motor_insurance = motor_insurance %>% mutate(loss = ifelse(
  loss < percentiles[1] | loss > percentiles[2], NA, loss))

# Update the 'loss' variable to reflect the modified data in the dataset
loss = motor_insurance$loss

# Plot a boxplot again to visualize the 'loss' column after removal of outliers
boxplot(loss, main = "Boxplot of the Total Loss Incurred to the Claimant", ylab = "
Loss")

# Part 2(c): Identify and Replace Outliers in the 'loss' Column

#Filter Out Outliers and Negatives
motor_insurance <- motor_insurance %>% filter(!is.na(loss) & loss >= 0)

# Part 2(d): Box-Cox Transformation to Reduce Skewness

# Apply Box-Cox transformation to identify the optimal lambda for transforming the
'loss' column
# - Fit a linear model with just the intercept (essentially to transform the 'loss'
 distribution)
boxcox_loss <- boxcox(lm(loss ~ 1, data = motor_insurance))

# Extract the optimal lambda value from the Box-Cox transformation results
# The lambda value that maximizes the likelihood
optimal_lambda <- boxcox_loss$x[which.max(boxcox_loss$y)] ; optimal_lambda

# Transform the 'loss' column using the optimal lambda to stabilize variance and re
duce skewness
motor_insurance <- motor_insurance %>% mutate(transformedLoss = (loss^optimal_lambd
a - 1) / optimal_lambda)

# Set up the plotting area to display two histograms side-by-side
par(mfrow = c(1, 2))  #Set plotting area to have 1 row and 2 columns for comparison
```

```r
# Plot histogram of the original 'loss' variable with a normal curve overlay
hist(motor_insurance$loss,
     main = "Original Loss Distribution with Skewed Curve",
     xlab = "Loss",
     col = "lightblue",
     freq = FALSE,
     xlim = c(0, 250))  # Set x-axis limit to focus on skewed part

# Add a normal distribution curve to the original 'loss' histogram
curve(dnorm(x, mean = mean(motor_insurance$loss, na.rm = TRUE),
            sd = sd(motor_insurance$loss, na.rm = TRUE)),
      add = TRUE, col = "blue", lwd = 2)

# Plot histogram of the transformed 'loss' variable with a normal curve overlay
hist(motor_insurance$transformedLoss,
     main = "Transformed Loss Distribution with Normal Curve",
     xlab = "Transformed Loss",
     col = "lightgreen",
     freq = FALSE)

# Add a normal distribution curve to the transformed 'loss' histogram
curve(dnorm(x, mean = mean(motor_insurance$transformedLoss, na.rm = TRUE),
            sd = sd(motor_insurance$transformedLoss, na.rm = TRUE)),
      add = TRUE, col = "blue", lwd = 2)

# Part 2(e): Mean Claims by Gender

# Filter out rows with missing values in 'claimantGender' before performing group a
nalysis
motor_insurance_gender = motor_insurance %>% filter(!is.na(claimantGender))

# Calculate mean claims by gender (1 for Male, 2 for Female)
mean_claims <- motor_insurance_gender %>%
  group_by(motor_insurance_gender$claimantGender) %>%
  summarize(mean_claims = mean(loss, na.rm = TRUE)) ; mean_claims

# Part 2(f): Age Group Categorization

# Filter out rows with missing values in 'claimantAge' before categorizing into age
 groups
motor_insurance = motor_insurance %>% filter(!is.na(claimantAge))

# Create a new categorical variable 'ageGroup' based on 'claimantAge' ranges
motor_insurance <- motor_insurance %>%
  mutate(ageGroup = cut(claimantAge,
                        breaks = c(-Inf, 25, 35, 42, 72, Inf),
                        labels = c("under 25", "26-35", "36-42", "43-72", "72 or a
bove"))) ; motor_insurance

# Calculate the number of observations and mean loss for each age group
motor_insurance %>% group_by(ageGroup) %>% summarise(count=n(),mean(loss))
```

```r
# ==========================
# - Part 3: Perform Principal Component Analysis and linear regression modeling.
# ==========================

#Load the Data
marine <- read_xlsx('Marine.xlsx')  # Load the Marine dataset from an Excel file

# 3(a): Calculate the Correlation Matrix

# Calculate the correlation matrix of the dataset to understand relationships betwe
en variables
correlation_matrix <- cor(marine) ; correlation_matrix

# Convert the correlation matrix to long format and plot using ggplot
# This helps visualize the relationships between different variables in the dataset
ggcorrplot(correlation_matrix,
           lab = TRUE,
           lab_size = 3,
           tl.cex = 9,
           title = "Correlation Heatmap of Marine Dataset Variables")

# Remove duplicate and diagonal entries by setting the lower triangle and diagonal
to NA
correlation_matrix[lower.tri(correlation_matrix, diag = TRUE)] <- NA

# Find the indices of the most correlated pair (excluding NA)
max_corr <- which(abs(correlation_matrix) == max(abs(correlation_matrix), na.rm = T
RUE), arr.ind = TRUE)

# Extract and print the names of the most correlated variables
most_correlated <- c(rownames(correlation_matrix)[max_corr[1]],
                     colnames(correlation_matrix)[max_corr[2]])
cat("The two most correlated variables are:", most_correlated[1], "and", most_corre
lated[2])

# 3(b): Scatter Plot Matrix and Q-Q Plots

# Scatter Plot Matrix using GGally
par(mfrow = c(1,1))
ggpairs(marine, title = "Scatter Plot Matrix of Marine Dataset")

# Generate Q-Q plots for each variable to check normality
par(mfrow = c(1,ncol(marine))) # Set up plotting area for Q-Q plots of each variabl
e
for(i in 1:ncol(marine)) {
  var <- as.numeric(marine[[i]])  # Convert each column to a numeric vector
  qqnorm(var, main = paste0('Q-Q Plot of ', colnames(marine)[i]))
  qqline(var, col = 2, lty = 2, lwd = 2)
}

# Standardize data using Z-score standardization
marine_std <- as.data.frame(scale(marine))

# Perform Principal Component Analysis (PCA)
```

```r
# Select all columns except 'claimCount' and 'claimCost' for PCA
marine_data <- marine_std %>% dplyr::select(!claimCount & !claimCost)

pca_result <- princomp(marine_data, cor = TRUE) # Conduct PCA on the standardized d
ataset
summary(pca_result) # Summary of PCA showing importance of each principal component

pca_result$loadings[,1]  # Show loadings for the first principal component

# Calculate variance explained by each component and plot Scree Plot
variance_explained <- pca_result$sdev^2 # Compute variance explained by each compon
ent
proportion_variance <- variance_explained / sum(variance_explained) # Calculate pro
portion of variance
cumulative_variance <- cumsum(proportion_variance) # Cumulative variance explained

# Scree plot with elbow method visualization
par(mfrow = c(1, 1))
plot(proportion_variance, type = "b", pch = 19, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained", main = "Scree Plot")
abline(v = which.max(cumulative_variance >= 0.80), col = "red", lty = 2)  # Marks 8
0% variance threshold

# Determine the minimum number of components that retain at least 80% of the varian
ce
num_comp <- which(cumulative_variance >= 0.80)[1]
cat("The number of components retaining at least 80% of variation is:", num_comp)

# 3(c): Plot PCA Biplot

#  Create PCA biplot to visualize components
biplot(pca_result, scale = 0, main = "PCA Biplot", cex = 0.5, ylim=c(-4,4))

# 3(d): Multiple Linear Regression with Principal Components

# Select the principal components that explain at least 80% of variance
pca_select <- pca_result$scores[,1:num_comp]

# Combine the selected principal components with the standardized 'claimCost' varia
ble
marine_data_pca <- data.frame(pca_select, claimCost = marine_std$claimCost)

# Fit a multiple linear regression model to predict 'claimCost' using selected prin
cipal components
pca_lm <- lm(claimCost ~ ., data = marine_data_pca)
# Display summary statistics for the linear regression model
summary(pca_lm)

# 3(e): Predicting Claim Costs for New Observations

# Create new data for prediction with standardized features
new_spec = data.frame(vesselVal = 22, vesselAge = 14.5, distance = 25, duration = 1
0)

# Standardize the new data based on the original dataset means and standard deviati
ons
```

```r
new_spec$vesselVal = (new_spec$vesselVal - mean(marine$vesselVal))/sd(marine$vesselVal)
new_spec$vesselAge = (new_spec$vesselAge - mean(marine$vesselAge))/sd(marine$vesselAge)
new_spec$distance = (new_spec$distance - mean(marine$distance))/sd(marine$distance)
new_spec$duration = (new_spec$duration - mean(marine$duration))/sd(marine$duration)

# Apply the PCA transformation to the new standardized data
new_spec_pca = as.data.frame(predict(pca_result, new_spec))

# Predict 'claimCost' for the new data using the fitted regression model
prediction = predict(pca_lm,new_spec_pca); prediction

# Reverse the standardization for the predicted 'claimCost' to obtain an interpretable value
unstand_prediction = prediction * sd(marine$claimCost) + mean(marine$claimCost); unstand_prediction

# ==========================
# - Part 4: Build and evaluate a deep neural network for predicting insurance claims.
# ==========================

# Load your dataset here
data <- read_csv("freMTPL2freq.csv") # Load the dataset from a CSV file

# 4(a): Data Preparation

# Display the structure of the dataset to understand variable types and dimensions
str(data)

# Convert character columns to factors for easier modeling
for(i in seq_along(data)){
  if(is.character(data[[i]])){
    data[[i]] <- factor(data[[i]])
  }
}

# Cap 'ClaimNb' values to a maximum of 4
data$ClaimNb <- pmin(data$ClaimNb, 4) # Prevent extreme values in 'ClaimNb' by capping it at 4

# Cap 'Exposure' values to a maximum of 1
data$Exposure <- pmin(data$Exposure, 1) # Cap 'Exposure' at 1 to normalize this feature

# Split Data into Learning and Test Sets
learn_idx <- sample(1:nrow(data), round(0.9*nrow(data)), replace = FALSE) # Randomly select 90% of data for training
learn <- data[learn_idx, ]  # Training data
test <- data[-learn_idx, ]  # Test data
n_l <- nrow(learn) # Number of rows in training set
n_t <- nrow(test) # Number of rows in test set

# Define function to perform Min-Max scaling (normalization to range [-1, 1])
```

```r
MM_scaling <- function(x){ 2*(x-min(x))/(max(x)-min(x)) - 1}

# Create a scaled dataset specifically for neural network modeling
data_NN <- data.frame(ClaimNb = data$ClaimNb) # Initialize data frame with 'ClaimNb
' column
data_NN$DriveAge <- MM_scaling(data$DrivAge) # Scale 'DrivAge'
data_NN$BonusMalus <- MM_scaling(data$BonusMalus) # Scale 'BonusMalus'
data_NN$Area <- MM_scaling(as.integer(data$Area)) # Scale 'Area' (convert to intege
r first)
data_NN$VehPower <- MM_scaling(as.numeric(data$VehPower)) # Scale 'VehPower'
data_NN$VehAge <- MM_scaling(as.numeric(data$VehAge)) # Scale 'VehAge'
data_NN$Density <- MM_scaling(data$Density) # Scale 'Density'
data_NN$VehGas <- MM_scaling(as.integer(data$VehGas)) # Scale 'VehGas'

# Split the scaled data into learning and test sets
learn_NN <- data_NN[learn_idx,] # Training data for neural network
test_NN <- data_NN[-learn_idx,] # Test data for neural network

# Create matrices for model input (excluding 'ClaimNb' which is the target variable
)
Design_learn <- as.matrix(learn_NN[ , -1])
Design_test <- as.matrix(test_NN[ ,-1])

# Extract categorical variables for embedding layers
Br_learn <- as.matrix(as.integer(learn$VehBrand)) - 1 # Vehicle brand for training
set (zero-indexed)
Br_test <- as.matrix(as.integer(test$VehBrand)) - 1 # Vehicle brand for test set

Re_learn <- as.matrix(as.integer(learn$Region)) - 1 # Region for training set (zero
-indexed)
Re_test <- as.matrix(as.integer(test$Region)) - 1 # Region for test set

# Extract Exposure data and apply log transformation for use as an offset
Vol_learn <- as.matrix(learn$Exposure)
Vol_test <- as.matrix(test$Exposure)
LogVol_learn <- log(Vol_learn) # Log transformation of 'Exposure' for training
LogVol_test <- log(Vol_test)  # Log transformation of 'Exposure' for test

# Set the target variable for training and test sets
Y_learn <- as.matrix(learn_NN$ClaimNb)
Y_test <- as.matrix(test_NN$ClaimNb)

# 4(b): Define the Neural Network Architecture

# Define layer sizes and embedding dimensions
q1 <- 27 # the dimension of the 1st hidden layer
q2 <- 22 # the dimension of the 2nd hidden layer
q3 <- 17 # the dimension of the 3rd hidden layer
qEmb <- 2 # the dimension of the embedded layer for "VehBrand" and "Region"

epochs <- 200 # number of epochs to train the model
batchsize <- 10000 # number of samples per gradient update

# Input layer for continuous features
# DesignShape <- ncol(learn_NN) - 1 # the number of the continuous predictors
```

```
Design <- layer_input(shape = ncol(learn_NN) - 1, dtype = 'float32', name = 'Design
')

# Input layers for categorical features
Br_ndistinct <- length(unique(learn$VehBrand)) # Number of unique vehicle brands =
11
Re_ndistinct <- length(unique(learn$Region)) # Number of unique regions = 21

VehBrand <- layer_input(shape = 1, dtype = 'int32', name = 'VehBrand') # Input laye
r for 'VehBrand'
Region <- layer_input(shape = 1, dtype = 'int32', name = 'Region')  # Input layer f
or 'Region'

# Input layer for Exposure (as the offset)
LogVol <- layer_input(shape = 1, dtype = 'float32', name = 'LogVol')
Vol <- layer_input(shape = 1, dtype = 'float32', name = 'Vol')

# Step 2.3: Embedding Layers for Categorical Features

# Embedding layer for 'VehBrand'
BrEmb = VehBrand %>%
  layer_embedding(input_dim = Br_ndistinct, output_dim = qEmb, input_length = 1, na
me = 'BrEmb') %>%
  layer_flatten(name = 'Br_flat')

# Embedding layer for 'Region'
ReEmb = Region %>%
  layer_embedding(input_dim = Re_ndistinct, output_dim = qEmb, input_length = 1, na
me = 'ReEmb') %>%
  layer_flatten(name = 'Re_flat')


# Step 2.4: Main Neural Network Architecture and Output Layer

# Concatenate the input layers and build the hidden layers
Network <- list(Design, BrEmb, ReEmb) %>% layer_concatenate(name = 'concate') %>%
  layer_dense(units = q1, activation = 'tanh', name = 'hidden1') %>%  # 1st hidden
layer
  layer_dense(units = q2, activation = 'tanh', name = 'hidden2') %>%  # 2nd hidden
layer
  layer_dense(units = q3, activation = 'tanh', name = 'hidden3') %>%  # 3rd hidden
layer
  layer_dense(units = 1, activation = 'linear', name = 'Network')  # provide one ne
uron in the output layer

# Define the output layer for the response variable using the offset layer
Response = (Network + LogVol) %>%
  # give the response
  layer_dense(units = 1,
              activation = 'exponential',
              name = 'Response',
              trainable = FALSE,
              weights = list(array(1, dim = c(1,1)), array(0, dim = c(1))))
```

```r
# Step 2.5: Model Configuration and Fitting

# Assemble the model with input and output layers
model <- keras_model(inputs = c(Design, VehBrand, Region, LogVol), outputs = c(Resp
onse))
summary(model) # Display model summary to show architecture details

# Compile the model with loss function and optimizer
model %>% compile(
  loss = 'poisson', # set poisson deviance loss function as the objective loss func
tion
  optimizer = 'nadam' # Optimizer used for training
)

# Model fitting by running gradient descent method to minimize the objective loss f
unction
{
  t1 <- proc.time() # Start timer to measure training time

  fit <- model %>% fit(
    list(Design_learn, Br_learn, Re_learn, LogVol_learn), # all predictors
    Y_learn, # response variable
    verbose = 1, # Verbose = 1 for detailed training progress
    epochs = epochs, # Number of training epochs = 200
    batch_size = batchsize, # Batch size for each iteration = 10,000
    validation_split = 0.2 # Use 20% of data as validation set
  )

  print(proc.time()-t1) # Print elapsed training time
}

# Predict 'ClaimNb' for both training and test datasets
learn$nn0 <- as.vector(model %>% predict(list(Design_learn, Br_learn, Re_learn, Log
Vol_learn)))
test$nn0 <- as.vector(model %>% predict(list(Design_test, Br_test, Re_test, LogVol_
test)))

# Define custom function to compute deviance loss
dev.loss <- function(y, mu, density.func) {
  logL.tilde <- log(density.func(y, y))
  logL.hat <- log(density.func(y, mu))
  2 * mean(logL.tilde - logL.hat)
}

# Compute deviance loss for the training data predictions
dev.loss(y = learn$ClaimNb, mu = learn$nn0, dpois)
```