Pricing Prediction Analysis and Agent ScoreCard

Project Report submitted in the partial fulfilment

of

B.Tech
In
Artificial Intelligence

by

Ananya Datta (I014)

Under the supervision of

Dr. Hima Deepthi

(Assistant Professor, AI, MPSTME)

SVKM's NMIMS University
(Deemed-to-be University)

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING (MPSTME)

Vile Parle (W), Mumbai-56

(2024-25)

CERTIFICATE

This is to certify that the project entitled **Pricing Prediction Analysis & Agent Scorecard**, has been done by **Ms. Ananya Datta** under my guidance and supervision & has been submitted in partial fulfilment of the degree of B.Tech in Artificial Intelligence of MPSTME, SVKM's NMIMS (Deemed-to-be University), Mumbai, India.

_____ _____

Project mentor (Dr. Hima Deepthi) Examiner (Name and Signature)

(Internal Mentor)

Date: 17th May 2025

Place: Mumbai

_____

HoD (Dr. Vaishali Kulkarni)

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my industry mentor, Mr. Devendra Rajguru, for his invaluable support and guidance throughout the course of internship. His insights and expertise have been critical in shaping my work and driving it to completion. My appreciation extends to my faculty mentor, Dr. Hima Deepthi, as well as the AI department for planning various reviews that guided us to achieve the most. I would also like to extend my special thanks to all my colleagues for helping me explore software such as PostGreSQL, Pentaho and Tableau, as well as for resolving all my doubts regarding corporate processes and specific project data. Finally, I am deeply grateful to my family for their infinite encouragement and love, which have been a constant source of inspiration and motivation throughout this journey.

PARAMOUNT HEALTHCARE MANAGEMENT PVT. LTD.

Paramount Health

502, Sumer Plaza, Marol Maroshi Road, Marol, Andheri (e), Mumbai - 400059
TEL.: (022) 4000 4200 • FAX : (022) 4000 4280 • Website : www.paramount.healthcare • CIN NO: U85100MH1996PTC104772

May 13, 2025

COMPLETION CERTIFICATE

This is to certify that Ms.**Ananya Datta** Roll No..**I014** has completed training & project as a part of Internship in our company as mentioned below and the Report is also submitted.

(i) Project Title: Premium Prediction Analysis & Agent Scorecard

(ii) Date of Joining: 02nd January 2025

(iii) Date of Completion: 02nd July 2025

In partial fulfilment of VIII Semester Internship for B Tech Artificial Intelligence program of Mukesh Patel School of Technology Management & Engineering, Narsee Monjee Institute of Management Studies (NMIMS) (Deemed-to-be University), Mumbai.

Devendra Rajguru
Industry Mentor
Paramount Healthcare Management Pvt. Ltd

ABSTRACT

This report outlines the internship experience at Fairfax Asia's Data Analytics Office, focusing on two key projects – **Premium Pricing Prediction Analysis** and **Agent Scorecard Development**. The first project aims to develop predictive models to forecast motor insurance premiums based on historical data and influencing factors, employing machine learning techniques like CatBoost, LightGBM, and XGBoost. These models were trained and evaluated for accuracy in predicting premiums, with CatBoost emerging as the model of choice due to its robust handling of categorical variables, stability and interpretability. The second project revolves around the creation of an agent scorecard to assess agent performance across multiple business metrics, such as profitability, growth and claims frequency and severity. The goal is to identify opportunities for improvement and optimize agent performance. The report covers data preprocessing, methodology, results and inferences from both projects, providing a comprehensive overview of data analysis and modelling techniques used, as well as the operational impacts of the insights generated.

Table of Contents

List of the figures

Fig 21.	Code Snippet of Scoring logic	

List of tables

Abbreviations

Gwp – Gross Written premium

Gic – Gross Incurred Claim

Nep – Net Earned Premium

Nic – Net Incurred Claim

Noc_total – Total #claims

Cust_post_code – Customer postcode

Cust_gender – Customer gender

Cust_age – Customer age

Cc – Cubic capacity of the vehicle

Inception_date_final – Inception date of the policy

Expiry_date – Expiry date of the policy

Policy_no – Identifier of the Policy

SHAP – Shapely Additive exPlanations

account_yr – Accounting year of the Agent

agent_start_date – Working start date of the Agent

reporting_line – Reporting lines

agent_name – Name of the Agent

Chapter 1 Overview of Internship

The internship at **Fairfax Asia's Data Analytics Office** at Thane focuses on providing hands-on experience in data manipulation, management, analysis and modelling. The company extensively works in Data ETL (Extract, Transform, Load) and helping clients with data-driven insights. My internship is designed to expose myself to real-world applications of Python, Data Analytics and Machine Learning techniques.

1.1 Objectives/Scope of the internship

- To work with data using Python for Analysis and Management
- To assist in creating predictive models like Classification and Regression
- To support client-focused projects and collaborate with teams to deliver insights, and
- Help interpret and present data findings to business teams

1.2 Two Projects

1. **Premium Pricing Prediction Analysis** –

   This involved preprocessing the motor insurance data, then train predictive models (that of classification and regression), to forecast and analyse insurance premiums based on historical data and various influencing factors

2. **Agent Scorecard** –

   Agent-wise analysis of policies of numerous insurance domains based on their performance and various business metrics, helping to identify areas of improvement and opportunities for enhancement.

1.3 Regarding the Company

Fairfax Financial Holdings Limited is a holding company which, through its subsidiaries, is primarily engaged in property and casualty insurance and reinsurance and the associated investment management. Fairfax was founded in 1985 by the present Chairman and Chief Executive Officer, Prem Watsa. Prem refinanced a small Canadian insurance company and changed the name to Fairfax Financial, which was derived from the principles of fair and friendly acquisitions, based on the golden rule of 'treating others as we would like to be treated ourselves'.

Fairfax Asia Limited is an insurance company operating and providing innovative data solutions to insurers in the Asian region and a subsidiary of Fairfax Financial Holdings Limited. It's involved in various insurance products including life, health, marine, and property insurance. Fairfax Asia is headquartered in Hong Kong and has a presence in several countries across Asia, including India

Some companies under it are **Malaysia PACIFIC, Sri Lanka FAIRFIRST, Indonesia AMAG, Thailand FALCON**, and more such like these.

Their vision is to be the insurer's one-stop solution for implementing an intelligent business ecosystem from ideation to adoption. Their Mission is to blend business, data and technology to build intelligent experiences that simplify business execution. Some of the values they encourage are Empathy, Ethics, Speed, Quality & Relevance.

1.4 Key Technical Utilized

- Python programming for data analysis and predictive modelling
- Experience with libraries like Pandas, NumPy and Scikit-Learn for data manipulation and building.
- Knowledge of data ETL processes for managing, transforming data and monitoring prediction.
- Data visualization using tools like Matplotlib and Seaborn to present insights.

1.5 Soft Skills Utilized

- Presenting findings clearly to business teams, and upper management, and making complex data accessible.
- Working effectively with cross-functional teams, including data engineers, data analysts, underwriting teams (in client-side) and data scientists.
- Adaptability: Quickly learned new tools and techniques required for the projects beyond my internship objectives.

Chapter 2 Data Analysis, Exploration and Interpretation

2.1 Objectives of each project

1. **Premium Pricing Prediction Analysis –**
   Based on important features of the policy and its claims, objective is to forecast what the premium should be, in order to improve or aid the pricing process for underwriters. This will help increase both accuracy and competitiveness, by considering the bigger picture. Through this project, we aim to create awareness regarding pricing changes and encourage the pricing team to incorporate percentage changes in premiums charged by the insurance company. This, in turn, will help the company's risk factor over time.

2. **Agent Scorecard –**
   Main objective is to evaluate and analyse agent performance through Key Performance Indicators (KPIs), providing targeted insights or feedbacks for improving efficiency and outcomes. Through this analysis, we aim to identify which agents are the most profitable, which ones can contribute to more growth and which exhibit a lower loss ratio. Additionally, we will identify appropriate actionable findings for low-profitable agents to enhance their performance.

2.2 Dataset Overview

This dataset includes motor private car comprehensive policies from PACIFIC (PIB) Malaysia for the years 2020-2024.

- **Policy data:** Consists of 2,072,235 records and 117 features. Key features include policy details such as policy number, inception and expiry dates, vehicle information (e.g., make, model, engine number), coverage details (e.g., sum insured, policy type), customer details, and premium-related variables.
- **Claim data:** Contains 491,804 records and 123 features, including columns similar to the policy data but with additional claim-specific fields, along with their split variants.

The focus of this paper is on preprocessing the data, performing exploratory analysis, building predictive models, and evaluating their performance. The results and conclusions will be discussed after training and testing the models.

2.3 Data Preprocessing and Cleaning

The initial steps I took before modelling –

Part I

- The **'cc'** feature, which originally ranged from 0 to 9180, was binned into six categories: '0-1000', '1000-1500', '1500-2000', '2000-3000', '3000-4000', and '4000+'. This transformation helped simplify the feature and make it more interpretable while reducing unnecessary variability.

- Next, we refined the dataset by selecting only the most relevant features. Instead of keeping all available fields, we retained a total of 37 features that were common between policy and claim datasets. Additionally, we kept 'GWP' from policy data and 'gic' and 'noc_total' from claim data, as these were essential for our analysis. This step helped eliminate redundant attributes such as customer details, vehicle specifics, and prior policy information, ensuring a cleaner dataset for modeling.

- To further structure the data, we applied a group-by operation based on policy number, inception date, expiry date, and split. For policy records, we aggregated 'GWP', 'xc_GWP_prv', and 'gr_comm_amount', while for claim records, we summed 'gic', 'xc_GWP_prv', and 'noc_total'. This resulted in a dataset with 1,601,765 records for policy data and 60,541 records for claim data.

- After preprocessing, we combined both datasets into a single consolidated file, maintaining the common features while handling policy and claim-specific fields. Policy records had 'gic' and 'noc_total' set to 0, while claim records had 'GWP' set to 0. This allowed us to track premium payments and claims together in one data set.

- Finally, we applied another group-by operation on the consolidated dataset to further sum up 'gic', 'GWP', and 'noc_total' at the policy level. This ensured that each policy's complete history—including total premiums paid and claims filed—was captured in a single record, making the dataset more structured and easier to analyze.

To ensure data accuracy, we first compared the **total year-wise premium, total claims, and the number of claims** against the official dashboard, confirming that the figures aligned correctly. Additionally, we verified that the aggregated values in the merged dataset matched those in the original datasets before merging, ensuring consistency throughout the process.

- Here, we have total number of records = 1,601,845 with number of features = 40 (37 + ‘GWP’, ‘gic’, ‘noc_total’)

Part II

- We checked for missing data in the merged dataset and handled each feature accordingly. Based on the percentage of missing values, we found that cust_post_code, make, and model had minimal missing data, while cust_gender and cust_age had a moderate amount.

Additionally, nearly half of the records lacked data in the prv columns, indicating a significant portion of missing values in those fields.

```
missing percentage in CLAIM DATA: 12.58%
missing percentage in POLICY DATA: 11.15%
```

Fig 1. Total percentage of missing data in the two datasets

```
cust_post_code                  0.0%
cust_gender                    6.16%
cust_age                       6.16%
make                           0.51%
model                          0.74%
xc_policy_no_prv              49.39%
xc_inception_date_final_prv   49.39%
xc_expiry_date_prv            49.39%
xc_contract_type_prv          49.39%
xc_cover_prv                  49.39%
xc_sum_insured_prv            49.39%
xc_loading_prv                49.39%
xc_ncb_perc_prv               49.39%
xc_disc_perc_prv              49.39%
dtype: object
```

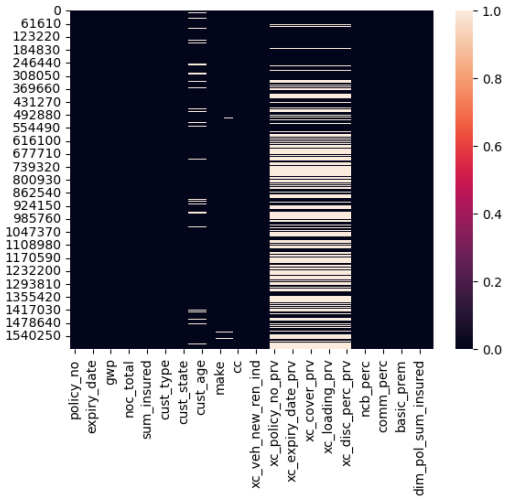


Fig 2. Missing data percentage in each column (also, in heatmap)

- We decided to drop columns where nearly half the records were missing, as synthetic augmentation wouldn't provide reliable data for the model to learn from. There were 7 columns as such and which were 'xc_policy_no_prv','xc_inception_date_final_prv','xc_expiry_date_prv','xc_contract_type_prv','xc_cover_prv','xc_sum_insured_prv','xc_loading_prv','xc_ncb_perc_prv','xc_disc_perc_prv'. Keeping these columns could lead to misrepresentation and negatively impact model performance.
- For **customer age**, we found that there were outliers present going beyond minimum of 20 and maximum of 80, which may be absurd if we consider average age of the person who gets insurance for cars, hence we cap and floor the extreme ends of customer age to be minimum = 20 and maximum = 80.
- Later, we filled the missing values of cust_age by randomly sampling from the existing non-null values, ensuring that the dataset remains complete without introducing bias from more complex imputation methods.
- Next, we fill out the missing values in cust_gender by randomly imputing genders ('M' or 'F') based on their proportions in the dataset, ensuring the probabilities sum to 1. This approach is used to handle missing gender data while maintaining the overall gender distribution in the dataset.
- To handle missing values of model, we first calculated the distribution of the 'model' column, identifying the top 10 most frequent models. Then, we calculated the proportions of these top models and used them to randomly sample values for the missing entries in the 'model'

column, ensuring the distribution of models remained consistent. Finally, we filled the missing values with the sampled models and verified that there were no remaining null values in the column.

- Since the missing values for 'cc', 'cust_post_code', and 'make' of the vehicles were minimal, we opted to remove the corresponding records from the dataframe to maintain data integrity.
- After removing all records and columns with missing values, we proceeded to add a new feature, loss_ratio, which is calculated by dividing each record's gross incurred claim amount by its gross weighted premium. We then classified these loss_ratio values into four categories, each with its corresponding meaning:
    - **Zero**: The customer has not made any claims.
    - **Non-Zero**: The customer has made a claim.
    - **Not Defined**: No premium is available, as the policy date falls outside the scope of the dataset considered here.
    - **Doesn't Exist**: Both the claim amount and premium amount are missing, likely due to incomplete or missing data for certain records.
- We removed those records having loss_ratio class as 'Not Defined' and 'Doesn't Exist'.
- On further discussions, there were 10 more columns which were considered not so important. Hence we dropped - 'xc_veh_new_ren_ind','xc_veh_lost_ren_ind', 'policy_type', 'disc_perc', 'comm_perc', 'loading', 'basic_prem', 'dim_pol_sum_insured', 'dim_pol_veh_sum_insured', 'xc_GWP_prv'.
- Now, we have the total number of records = 1,561,624 with features = 23

Part III

- The step of extracting the year, month and day from inception_date_final was done to create separate features for the year, month, and day of policy inception, which can be useful for time-based analysis or modeling, allowing us to explore seasonal patterns or trends over time.
- The dataset was divided into distinct subsets: the training set (for 2020-2022) was prepared for model development, while the test sets for 2023 and 2024 were created to evaluate the model's performance on unseen data, ensuring an unbiased assessment of the model.
- These columns – policy_no, expiry_date, noc_total were removed because they were deemed unnecessary for the pricing prediction modeling process. This step reduces dimensionality and focuses on the most relevant features that are critical for building predictive models.
- Certain columns, such as 'inception_year', 'inception_month', 'inception_day', and 'cc', had their data types converted to the appropriate formats (e.g., integers for year, month, and day) to ensure they are compatible with modeling algorithms and avoid potential errors during computation.
- The columns were classified as either numerical or categorical to organize the dataset correctly. This classification is crucial for modeling algorithms, like CatBoost, which treat numerical and categorical data differently during model training.
- Lastly, the 'branch' codes were mapped to the corresponding 'reporting_branch' values to standardize them and ensure consistency. Additionally, records with irrelevant 'branch' codes (e.g., 'H5') were removed, improving the quality of the data and ensuring that only meaningful records remain for analysis.
- The final shape of the data that we'd feed the model with is of 1,561,624 records having 21 features.

2.4 Data Exploration

- Using df.info(), we have various datatypes in our data which need to be processed respectively to fit well into the predictive model. Namely, floats (17 continuous numeric values), integer (4 discrete numerical values), objects (17 categorical/text based features) and category (1 probably with a categorical variable with predefined levels).

- Representations

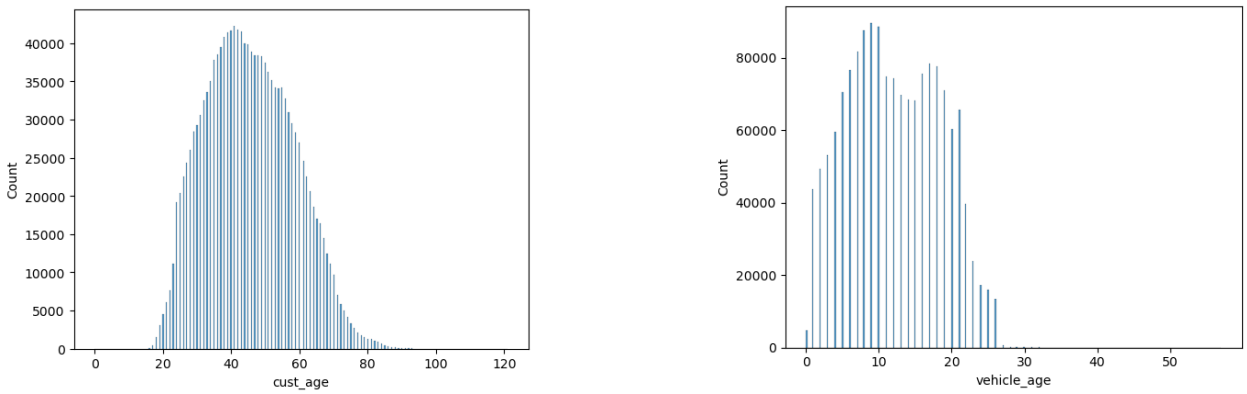


Fig 3. Distribution of Customer Age and Vehicle Age as per their count (in histogram)

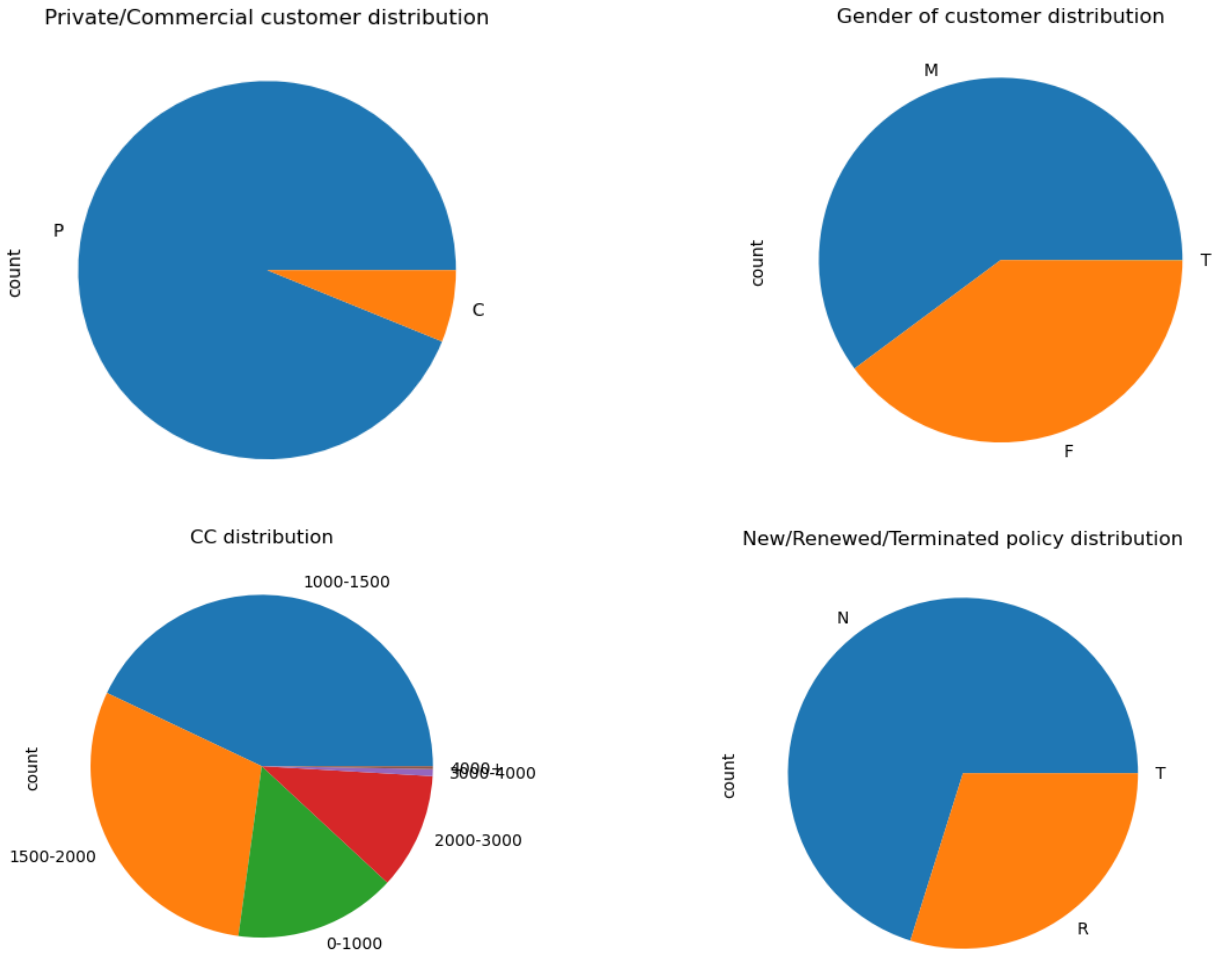
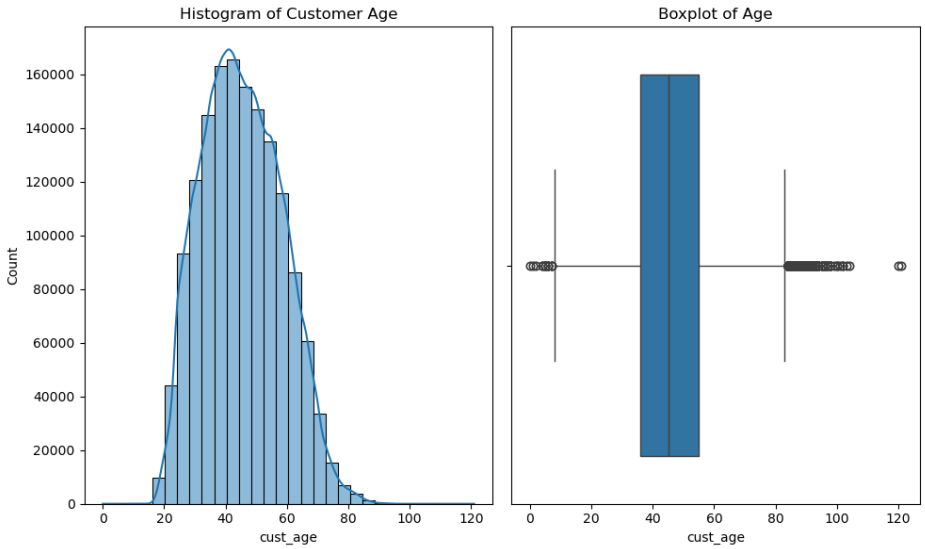


Fig 4. Distribution of Customer Type, Customer Gender, Cublic-Capacity, Policy Type

- Ouliers were detected in cust_age, which we handled by capping and flooring the extreme ages to 20 for minimum and 80 for max.

Before:



After:

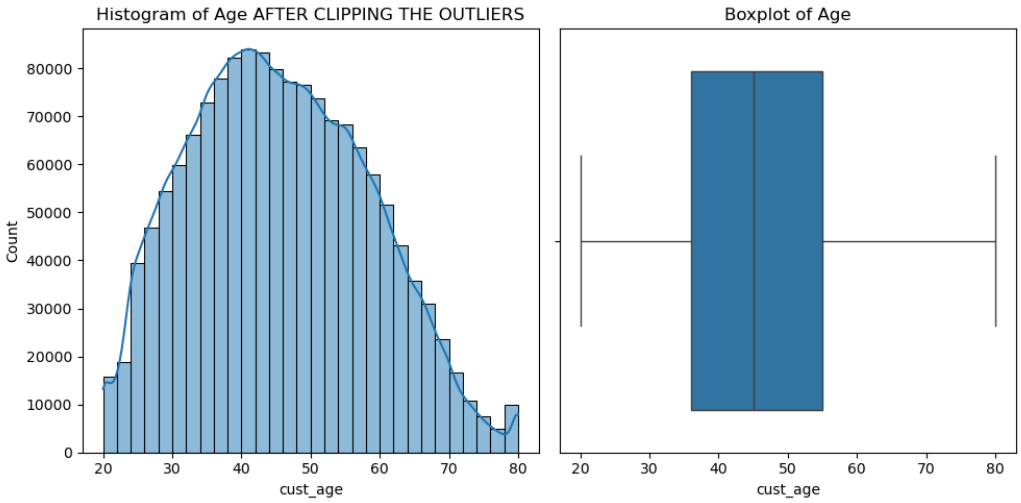


Fig 5. Distribution of customer age Before Vs. After outlier handling

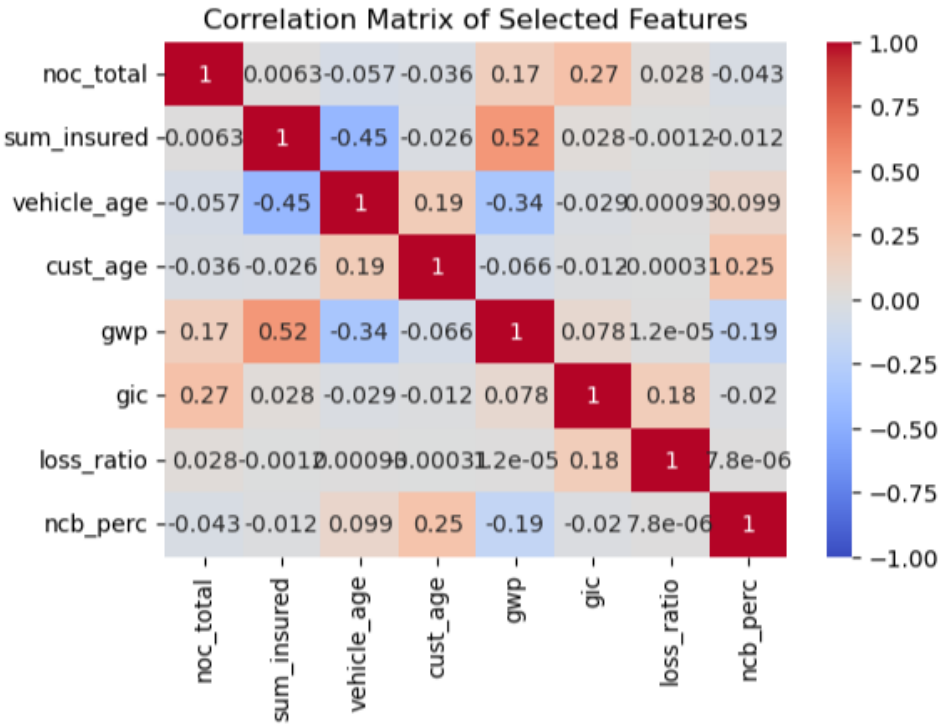


Fig 6. Correlation analysis plot over multiple important columns for 5 years

- We converted the 'inception_date_final' and 'expiry_date' columns to datetime format and calculated policy profitability by subtracting 'gic' from 'GWP'. Then, computed the policy duration and grouped the data by year to plot the average profitability trend over time.

Fig 7. Profitability trend over time

Chapter 3 Methodology and Implementation

3.1 Feature Selection

A filter feature selection approach – **Mutual Information** has been adopted to identify the most relevant features for predicting the target variable GWP (Gross Written Premium). Filter methods evaluate each feature independently of the model, using statistical techniques. Some of its characteristics are model-agnostic, fast and simple, doesn't account for feature interactions or multicollinearity and typically used as a preprocessing step before modelling.

Since Mutual Information requires numerical input, all categorical features were first encoded using Label Encoding and the features were then separated into:

X : All encoded input features (numerical and categorical)

Y: The target variable – GWP

Mutual Information Regression is a non-parametric method that captures both linear and non-linear dependencies between each feature and target variable. Mutual Information quantifies the amount of information obtained about one random variable through another. In the context of regression, it measures how much knowing the value of a feature reduces uncertainty about the target. This method is particularly effective when the relationships between features and the target variable are complex and not well captured by linear models. It assigns a score to each feature on its predictive power, enabling the selection of features with the highest relevance.

Out of 41 deducted features from the original dataset, the following subset was selected by the algorithm as most informative for predicting GWP:

'basic_prem', 'dim_pol_sum_insured', 'dim_pol_veh_sum_insured', 'sum_insured', 'model', 'xc_gwp_prv', 'branch', 'vehicle_age', 'policy_no', 'split', 'make', 'cust_post_code', 'ncb_perc', 'loading', 'cc', 'expiry_date', 'xc_veh_renewal_no', 'cust_age', 'gic', 'xc_veh_new_ren_ind', 'noc_total', 'cust_type', 'pref_segment', 'xc_veh_lost_ren_ind'

The list of features provided by domain expert, who identified the key drivers for pricing as:

'gic', 'sum_insured', 'cust_post_code', 'cust_age', 'vehicle_age', 'xc_veh_renewal_no', 'ncb_perc', 'pref_segment', 'loss_ratio', 'inception_year', 'split', 'branch', 'cust_type', 'cust_state', 'cust_gender', 'make', 'model', 'cc'

Analysis revealed that 14 features were common between the algorithmic and domain expert selections. 4 features were uniquely suggested by the domain expert: 'loss_ratio', 'inception_year', 'cust_state', 'cust_gender'.

We have incorporated insights from both perspectives and trained the model on features provided by the domain knowledge expert to ensure a more comprehensive and balanced model is trained.

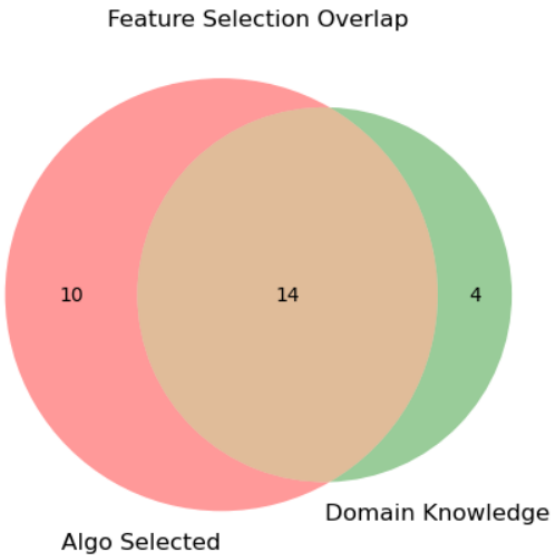


Fig 8. Venn Diagram to show no. of features selected by Algorithm and based on Domain knowledge and its intersection.

3.2 Model Training

I used multiple machine learning models in order to compare each model's given feature importances, metrics (such as R^2, adjusted R^2, RMSE and MAE) and its pros and cons.

3.2.1 CatBoost Model

Firstly, the **CatBoostRegressor** was used because it's particularly effective with categorical data and utilizes gradient boosting techniques. It incorporates methods like ordered boosting to efficiently handle such data, and early stopping ensures that the model does not overfit. The model's performance was thoroughly assessed using multiple evaluation metrics to understand its prediction accuracy, with all operations focused on optimizing the model's ability to generalize well on unseen data.

- The target variable y was set to 'GWP', and the feature set X was created using both numerical and categorical columns. This step defines what the model will predict and the input features used for training.
- **TimeSeriesSplit** was implemented to split the data into 5 folds, ensuring the model was evaluated on different temporal subsets. This approach is critical for time-series data as it avoids data leakage by maintaining the time-order integrity in the training and testing sets.
- The **CatBoostRegressor** model was initialized with specific hyperparameters like 900 iterations, a depth of 8, learning rate of 0.01, L2 regularization (leaf-wise) of 70, and early stopping after 50 rounds with no improvement. These settings were chosen to strike a balance between model complexity, overfitting prevention, and the speed of learning.
- Categorical columns were explicitly set by their indices, ensuring that CatBoost correctly processes them. This is important because CatBoost is optimized to handle categorical features natively, leveraging special algorithms like ordered boosting to improve performance.

- The model underwent **cross-validation** with TimeSeriesSplit, iterating through multiple training and test splits, ensuring that the model was evaluated on different parts of the data over time. This method provides a more robust evaluation, as the model is trained and tested on different subsets.
- Weights were applied to the training data to adjust the importance of different samples. This is useful when some records are more significant than others, such as those with higher loss ratios, ensuring that the model accounts for those differences in its training process.
- The columns were reordered so that the 'loss_ratio' feature came first. This prioritization was done with the assumption that the 'loss_ratio' is a crucial predictor and putting it at the beginning helps the model better learn its impact on the target variable.
- The CatBoost model was trained with early stopping, which helps prevent overfitting by halting training if the model's performance on the validation set does not improve after 100 rounds. This ensures that the model generalizes well and doesn't continue to fit noise in the data.
- After training, the model made predictions on the test set, and various evaluation metrics such as **R^2, MAE, MSE, and RMSE** were calculated to assess the model's performance. These metrics provide a comprehensive understanding of the model's accuracy and error rates. Here's a short description of each metrics mentioned:
    - **R^2 (R-squared)**: A statistical measure that indicates how well the independent variables explain the variability of the dependent variable. An R2 value closer to 1 means a better fit, while a value closer to 0 indicates that the model doesn't explain much of the variance in the data.
    - **MAE (Mean Absolute Error)**: The average of the absolute differences between predicted and actual values. It measures the magnitude of error in predictions, with lower values indicating better accuracy.
    - **MSE (Mean Squared Error)**: The average of the squared differences between predicted and actual values. It gives more weight to larger errors due to the squaring process, making it sensitive to outliers.
    - **RMSE (Root Mean Squared Error)**: The square root of MSE. It provides an error metric in the same unit as the target variable and gives more weight to larger errors. Lower RMSE values indicate a better model fit.

- The fold-wise performance metrics were averaged to get an overall view of the model's effectiveness. This approach provides a stable evaluation of model performance across different subsets of the data, helping mitigate any variance in the results from individual folds.

3.2.2 LightGBM Model

LightGBM developed by Microsoft, designed for fast, distributed, and efficient gradient boosting. Similarly, the dataset was trained on LightGBM, with the following parameters:

- **TimeSeriesSplit** was implemented to split the data into 5 folds, ensuring the model was evaluated on different temporal subsets. This approach is critical for time-series data as it avoids data leakage by maintaining the time-order integrity in the training and testing sets.
- Objective: 'regression' - which specifices that the model is used for a regression task, to predict GWP
- Boosting_type: 'gdbt' - which is a default boosting method in LightGBM. It builds the model in a stage-wise manner and generalizes it by optimizing a loss function, making it effective for tabular data.

- Learning_rate: 0.5 - smaller learning rate helps the model to learn more gradually, leading to better generalization.
- Num_leaves: 31 – controls the maximum number of leaves in each tree. A higher number allows for more complex models that can capture intricate patterns, but can also increase the risk of overfitting.
- Early_stopping_rounds: 100 – this avoids unneccessary computation and mitigates the risk of overfitting.
- Num_boost_round: 1000 – sets the upper limit on the number of boosting iterations(trees). The actual number of trees used may be fewer if early stopping is triggered.
- Feval: [custom metrics: R^2, MAE, MSE, RMSE] - custom evaluation functions were used instead of LightGBM's default metrics.

3.2.3 XGBoost Model

XGBoost is optimized Gradient Boosting framework known for speed and performance. It uses boosted decision for regression and classifications tasks. Similarly, the dataset was trained on XGBoost, with the following parameters:

- Model: XGBoost – an implementation of gradient boosted decision trees specifically designed for performance and speed.
- N_estimators: 1000 – defines the maximum number of trees the model will train.
- Learning_rate: 0.05 - controls how much the model adjusts during each boosting round.
- Booster: 'gbtree' - default and most common booster, which builds decision trees sequentially.
- Categorical variables were encoded using Label Encoding prior to training. It is compatible with XGBoost, though it does not treat them natively as categorical (unlike CatBoost).

Chapter 4 Results and Analysis

Table 1. CatBoost Regressor Model metrics

Metrics	Train (2020-22)	Test (2023)	Test (2024)
R^2 score	0.849	0.842	0.814
MAE	71.06	97.47	156.26
MSE	37,017.17	62,454.19	117,094.56
RMSE	191.88	249.81	342.19

Table 1. displays the performance metrics for the CatBoost Regressor model across 3 datasets: training (2020-22), test (2023) and test (2024). Key metrics to note:

- A high R^2 score of 0.849 on the training dataset suggests a strong model fit. However, there is a slight decline in performance on the test datasets, with the score dropping to 0.842 for 2023 and 0.814 for 2024, indicating a minor reduction in prediction accuracy over time.
- The MAE increases as we move from the training dataset (71.06) to the test datasets, with 2024 showing the highest error (156.26), reflecting a growing deviation from actual values as time progresses.
- The MSE follows a similar pattern as MAE, with an increase from 37,017.17 in the training set to 117,094.56 in the 2024 test set, emphasizing a worsening model performance as the prediction horizon extends.
- RMSE also increases over time, indicating the model's growing prediction error.

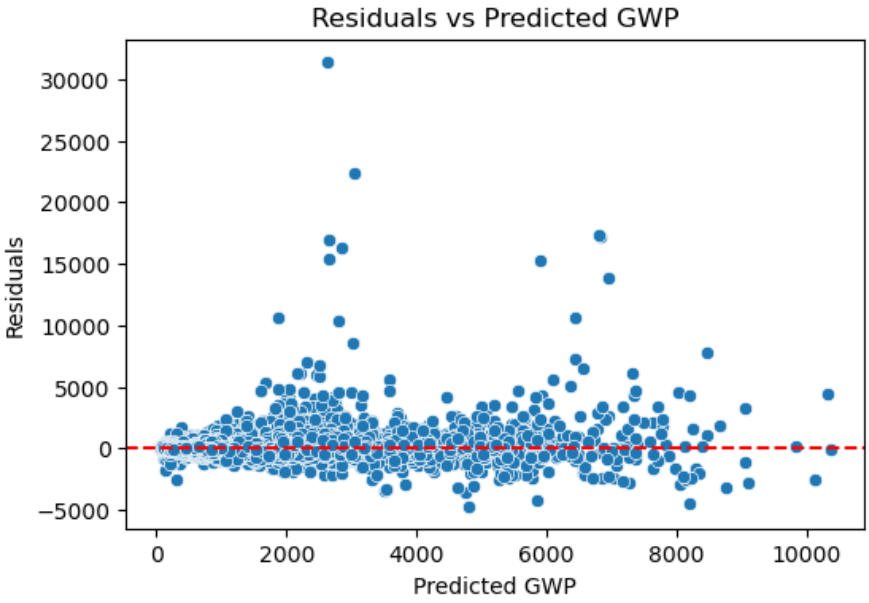


Fig 9. Residuals Vs Predicted GWP plot for Catboost Model

This plot in Fig 9. visualizes the residuals (differences between predicted and actual values) against the predicted GWP. A well-calibrated model would ideally show no clear pattern in the residuals. The plot can be analyzed to determine if the model exhibits any systematic bias, which could indicate areas where the model's predictions are consistently off. If the residuals display randomness and are evenly distributed around zero, it suggests that the model is appropriately capturing the underlying patterns.

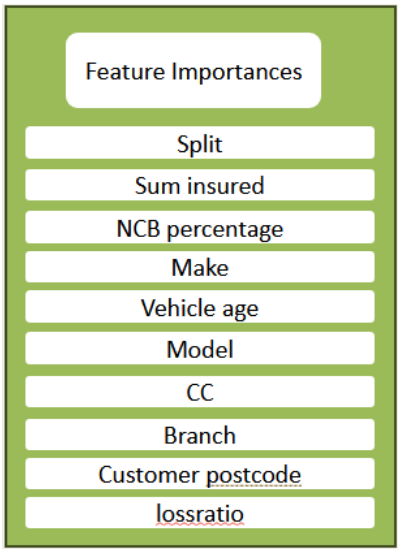


Fig 10. Feature Importances for CatBoost Regressor

Post model training feature analysis using ' **a post-hoc model interpretation tool'**, was also done to understand the contribution of each feature to a model's prediction, based on **SHAP** (SHapely Additive exPlanations) values. Positive SHAP values push predictions to a higher value and negative SHAP values push predictions lower value. Red denotes high feature value, and blue denotes low feature value. Each point represents a SHAP value for one instance of the dataset. SHAP values are derived from a trained model, attributing contribution of each feature to predictions. When used for feature selection, you're leveraging the internal workings (or learned patterns) of the model—making it embedded kind of feature selection model. Some of its characteristics are model-specific, considers feature interactions, requires a trained model and often used to rank features by importance after model training.

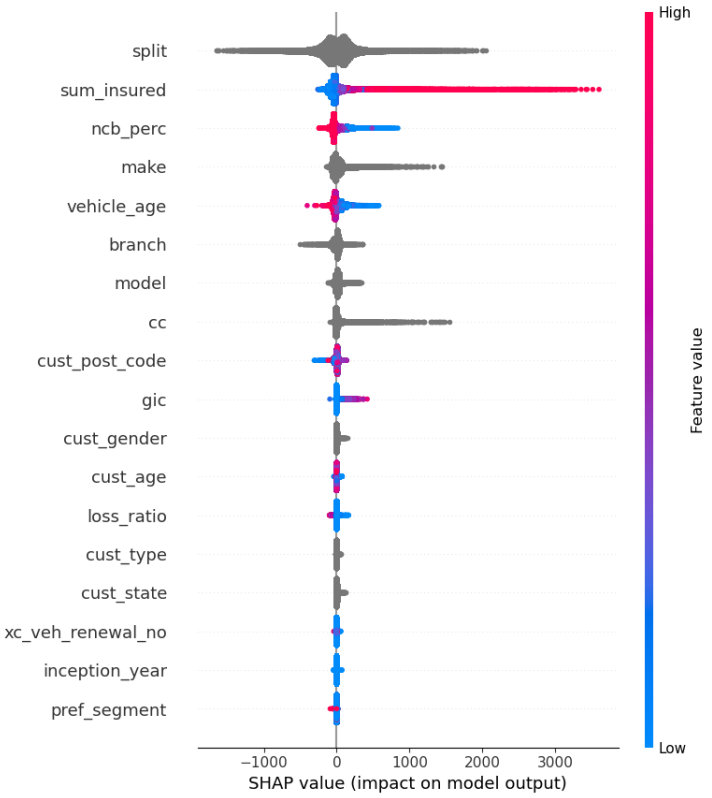


Fig 11. SHAP value (impact on model output)

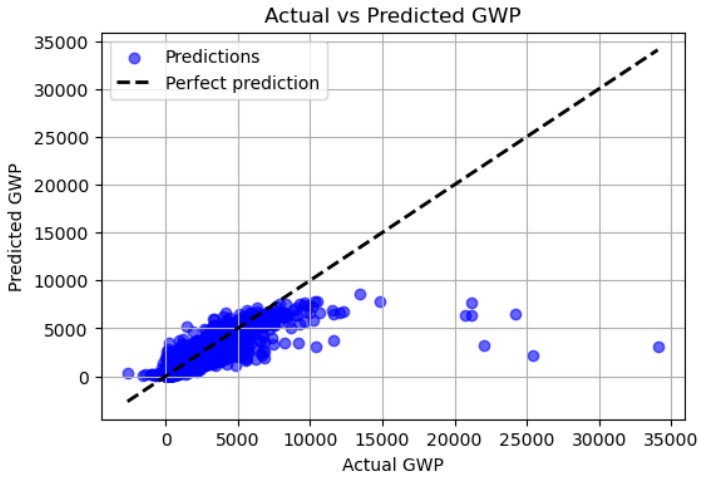


Fig 12. Actual Vs Predicted GWP after training CatBoost Model

This scatter plot Fig 12. compares the actual versus predicted GWP values after training the CatBoost model. A well-performing model would show points closely aligned with the 45-degree line, indicating that the predicted values are very close to the actual values. Deviations from the line reflect areas where the model underperforms, and this plot is useful for visualizing the model's prediction accuracy.

Table 2. LightGBM Model metrics

Metrics	Train (2020-22)	Test (2023)
R^2 score	0.874	0.905
MAE	69.13	85.21
MSE	27,869	37,607
RMSE	158.31	193.92

Table 2 presents the performance of the LightGBM model. Key observations include:

- **R^2 Score**: The model performs exceptionally well on the test data for 2023 (0.905), with a notable improvement over the CatBoost model.
- **MAE, MSE, and RMSE**: Like the CatBoost model, the MAE, MSE, and RMSE show increasing values from training to testing, but the model shows better overall performance on the 2023 dataset compared to CatBoost.

The performance on the test data suggests that LightGBM is highly effective at capturing the patterns in the data, outperforming CatBoost in some aspects.

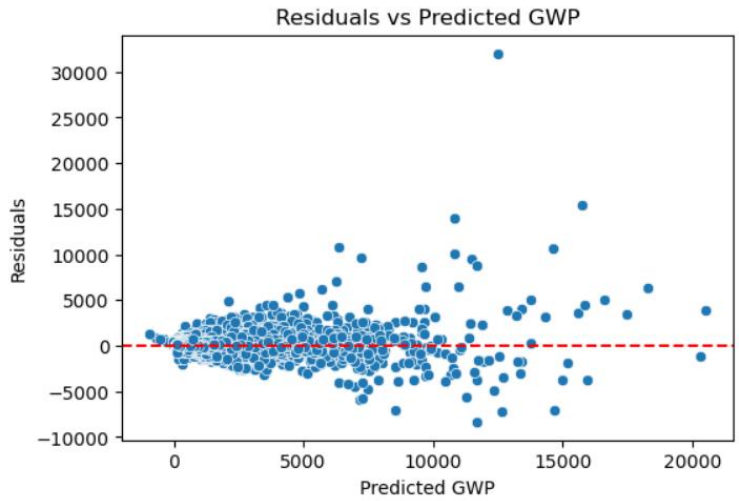


Fig 13. Residuals Vs Predicted GWP plot for Catboost Model

The residuals vs predicted plot Fig 13. for LightGBM provides insights into model behavior. Similar to Fig 9, this plot can be analyzed to evaluate the randomness and any biases in the model's predictions. A well-calibrated model would show a random distribution of residuals with no clear patterns.

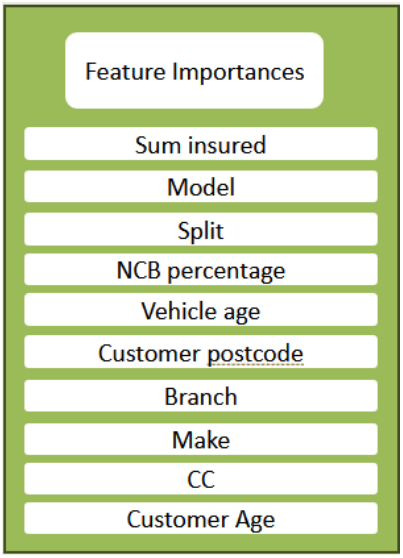


Fig 14. Feature Importances for CatBoost Regressor

Table 3. XGBoost Regressor Model metrics

Metrics	Train (2020-22)	Test (2023)	Test (2024)
R^2 score	0.946	0.898	0.86
MAE	50.21	79.62	146
MSE	16,536.32	40,320.64	88,571.78
RMSE	116.35	200.80	297.61

Table 3 outlines the performance of the XGBoost model. Compared to both CatBoost and LightGBM, XGBoost shows the highest R^2 score during training (0.946), and it maintains strong performance on the test data. However, performance drops slightly on the 2024 test data ($R^2 = 0.86$), indicating a slight overfitting on the training set. The MAE, MSE, and RMSE also show a similar pattern, with XGBoost consistently providing more accurate predictions than CatBoost, but with increasing error in the future.

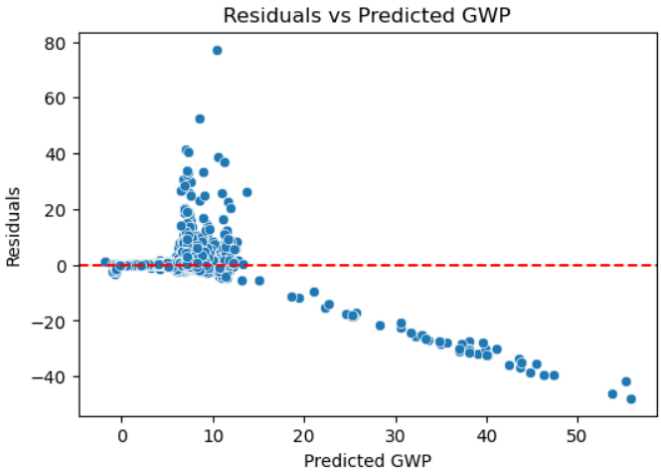


Fig 15. Residuals Vs Predicted GWP plot for XGBoost model

This residuals vs predicted plot Fig 15. for XGBoost helps assess if the model has any systematic errors. A well-behaved model would exhibit randomly distributed residuals, similar to the behavior of the other models.

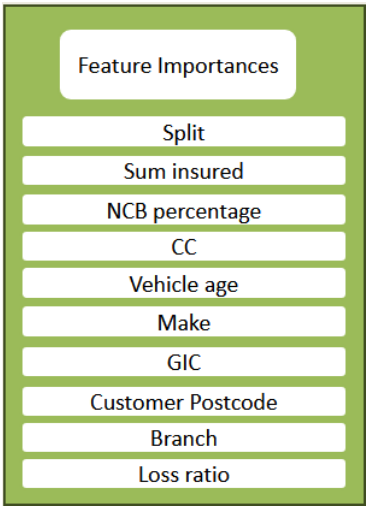


Fig 16. Feature Importances for XGBoost Model

The table 4. compares the three models—CatBoost, LightGBM, and XGBoost—based on their feature importance rankings, metrics (R^2), and the advantages and disadvantages of each. Key takeaways include:

- **CatBoost**: Best for handling categorical features natively and being robust to overfitting. However, it may be slower than LightGBM and less flexible with GPU acceleration.
- **LightGBM**: Known for its speed and memory efficiency, this model is excellent for large datasets but might require more manual preprocessing for categorical variables.
- **XGBoost**: The most accurate model with great flexibility and documentation, though it might require more tuning and preprocessing compared to the others.

Table 4: Comparison Summary of Models (Feature Importance, Metrics (R^2), Advantages & Disadvantages) - CatBoost, LightGBM, XGBoost

CatBoost	LightGBM	XGBoost
Feature importance: 1. Split 2. Sum insured 3. Ncb percentage 4. Make 5. Vehicle age 6. Cc 7. Branch 8. Customer postcode 9. Loss Ratio 10. Customer type 11. Customer age 12. GIC 13. Customer state 14. Customer gender 15. Preferred segment 16. Inception day 17. Xc_veh_renewal_no 18. Inception_year 19. Incepton_month	**Feature importance:** 1. Sum insured 2. Model 3. Split 4. Ncb percentage 5. Vehicle_age 6. Customer post code 7. Branch 8. Make 9. Cc 10. Customer age	**Feature importance:** 1. Split 2. Sum insured 3. Ncb percentage 4. Cc 5. Vehicle age 6. Make 7. GIC 8. Customer post code 9. Branch 10. Loss Ratio 11. Model 12. Customer type 13. Preferred segment 14. Xc_veh_renewal_no 15. Customer age 16. Inception year 17. Customer gender 18. Customer state
Metrics: R^2 Train: 84.9% Test 1: 84.16% Test 2: 81.3%	**Metrics: R^2** Train: 87.3% Test 1: 90.46%	**Metrics: R^2** Train: 94.66% Test 1: 89.88% Test 2: 86%
Advantages - Handles categorical features natively (no encoding!) - Less preprocessing required - Robust to overfitting - Great for imbalanced data - Easy interpretability	**Advantages** - Very fast training - Highly memory efficient - Great for large datasets - Native support for categorical features (though not as seamless as CatBoost) - Excellent accuracy	**Advantages** - Highly accurate and robust - Well-documented and mature - Flexible tuning - Performs great across many domains - Supports missing values

Disadvantages/limitations	**Disadvantages/limitations**	**Disadvantages/limitations**
- Slower training than LightGBM - Less flexible with GPU (though supported) - Can be sensitive to noise or outliers	- Manual encoding of categorical variables often needed - Can overfit on small data - Might be harder to interpret compared to CatBoost	- Slower than LightGBM on very large data - Heavier model size - Needs careful tuning for best performance - More preprocessing if categorical

Chapter 5 Agent Scorecard

In this project, we are supposed to be ranking the insurance agents as per its beneficial factors for the business. Main business questions that would help understand the purpose of this project is –

1. Is the agent profitable (as per sales)?
2. Is the agent showing growth (as per underwriters)?
3. What is the frequency and severity of insurance policies and claims that the agent is getting (as per claims)?

To answer these questions, we will have to look into features like revenue, claims, loss ratios, frequency, severity and average premium per policy sold; deeply and analytically. The main steps undertaken to analyze the agents efficiently are – Data Preprocessing, Descriptive Analysis, Data Visualization and Statistical Inferencing.

Also, the metrics over which an agent can be deemed profitable are mainly –

- **Frequency**: measures how often claim are made. Higher the frequency, negatively profitable.

  F = #claims / #policies

- **Severity**: measures the financial impact of those claims. Higher claim amount, lower profitability.
  S = amount of claims incurred / #claims

- **Loss ratio**: a financial metric that measure the proportion of premiums an insurer earns that are used to pay claims and adjust expenses. Lower loss ratio, higher profit.
  L = amount of claims incurred / premium charged

- **Profitability per policy**: determines whether the policies taken up agent is profitable or not.
  Profitability/policy = premium charged – claim incurred

- **Agent's profit margin**: measures proportion of premium income that remains as profit after claims are paid.
  Profit margin = (premium charged – claim incurred)/ premium charged

- **Average premium per policy**: measures how much premium is generated for each policy
  Avg premium/policy = premium charged / #policies

- **Renewal Rate**: percentage of policies that are renewed after their initial term has ended.
  RR = #renewal / #policies

Steps taken into the analysis:

1. **Understanding the data**

- First, imported required libraries like pandas, numpy and matplotlib.pyplot. Next, loaded the agent view csv file across all line of businesses (LOBs). 14 unique lobs were identified –
  - PAC: Personal Accident
  - FIR: Fire

- LIA: Liability
- ENG: Engineering
- WRK: Workmen Compensation
- ACC: Accident
- MAR: Marine
- BON: Bond
- AVI: Aviation
- HUL: Hull
- MOT: Motor
- UNK: Unknown
- ZNA: Others

- Percentage distribution of unique agents/records in each lob is given in Fig x.

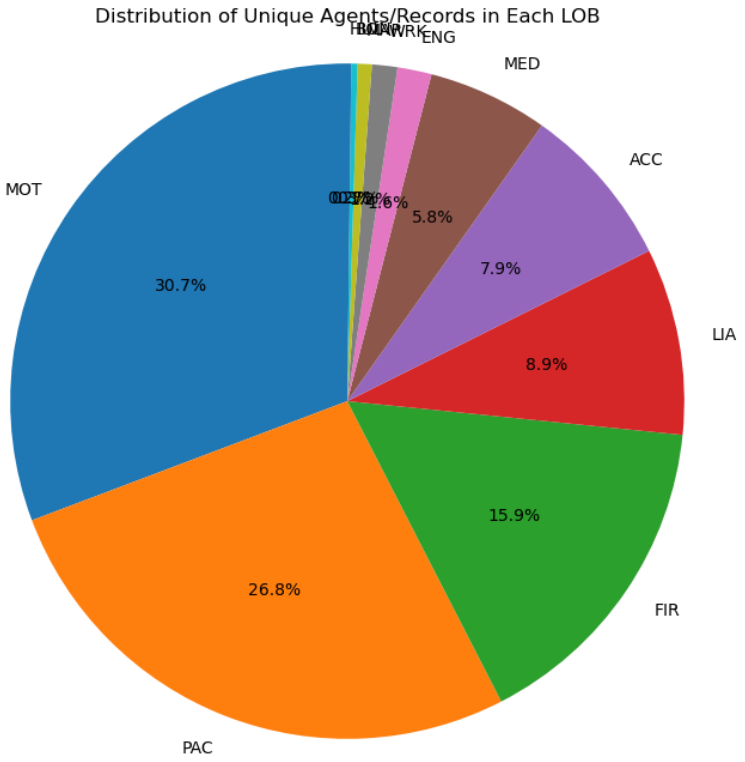


Fig 17. Distribution of Unique agent in each LOB

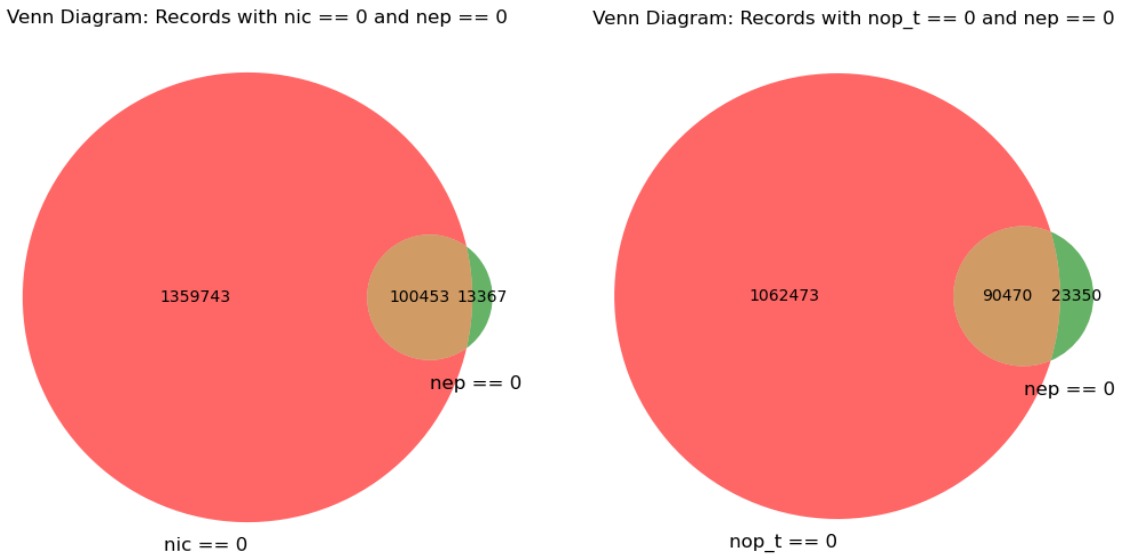


Fig 18. Number of records analysis for zero valued features mainly – nep, nic, nop_t

2. Data Cleaning, Exploration and Manipulation

- Data Cleaning was done by removing missing records in agent_name or nep, as well as dropping columns such as agent_end_date. Later, filtering out records having nep == 0 (no premium), nop_t == 0 (no policies) and negative nic. Final cleaned data size: 363,826 records
- Unique number of agents in each lobs: 3282 MOT, 1985 PAC, 1839 FIR, 1556 LIA, 1061 ACC, 927 MED, 723 WRK, 616 ENG, 245 MAR, 173 BON, 23 HUL
- Created another column – account_yr from account_date to get yearly information and sort with respect to it. Then, aggregated the data based on account_date, agent_name, lob and reporting_line, by adding up nep, nic, noc, nop_t and keeping the first agent_start_date, account_yr. Here the data size was: 173,258 records with 11 features.
- Aggregated again based on account_yr, agent_name, lob and reporting_line. Data size: 35,316 records and 10 features.
- Calculated features – Frequency, Severity and Loss_ratio for each record (based on year and agent_name) and each of these factors' distribution is checked via binning.

```
FACTOR: frequency
(-0.023200000000000002, 2.222]    35293
(2.222, 4.444]                       12
(4.444, 6.667]                        2
(6.667, 8.889]                        4
(8.889, 11.111]                       3
(11.111, 13.333]                      0
(13.333, 15.556]                      0
(15.556, 17.778]                      0
(17.778, 20.0]                        1
(20.0, 22.222]                        1
Name: count, dtype: int64
```

```
FACTOR: severity
(-974.337, 97433.647]       7870
(97433.647, 194867.294]       54
(194867.294, 292300.941]      13
(292300.941, 389734.588]       2
(389734.588, 487168.235]       1
(487168.235, 584601.882]       1
(584601.882, 682035.529]       0
(682035.529, 779469.176]       2
(779469.176, 876902.823]       0
(876902.823, 974336.47]        1
Name: count, dtype: int64
```

```
FACTOR: loss_ratio
(-7676.563, -6834.767]        1
(-6834.767, -6001.307]        0
(-6001.307, -5167.847]        0
(-5167.847, -4334.387]        0
(-4334.387, -3500.926]        0
(-3500.926, -2667.466]        0
(-2667.466, -1834.006]        0
(-1834.006, -1000.546]        0
(-1000.546, -167.086]         0
(-167.086, 666.375]       35314
Name: count, dtype: int64
```

Fig 19. Distribution of Factors based on binning

3. Implement Scoring logic

- Defined custom loss ratio thresholds for each Line of Business (lobs) since each lob operates under different cost structures and risk environments. These thresholds represent the maximum acceptable loss ratio for that lob. For example, the threshold for MOT (motor) is 0.75 (as in 75%), so policies having ratio of claims and premiums going higher than 75% will be considered unprofitable to a business.

```python
# Thresholds for each line of business (lob)
lob_thresholds = {
    'PAC': 0.50,
    'FIR': 0.50,
    'LIA': 0.50,
    'ENG': 0.60,
    'WRK': 0.20,
    'ACC': 0.45,
    'MAR': 0.35,
    'MED': 0.65,
    'BON': 0.10,
    'HUL': 0.65,
    'MOT': 0.75
}
```

Fig 20. Loss ratio thresholds for each LOBs

- To calculate a final performance score for each agent year wise, we have defined a weighted scoring model using 3 key dimensions:
    - Profitability (40%): measures how profitable the agent's portfolio is, relative to the LOB benchmark.
    - Claims Behavior (30%): reflects how frequently and severely claims are made under the agent's book in a year.
    - Revenue Growth (30%): indicates business growth based on year-over-year premium improvement.

  These weights were chosen to reflect the business priority of maintaining profitability while also encouraging growth and efficient claims handling.
- For each agent's record, I calculated the difference between the applicable LOB threshold and the actual loss ratio (i.e., lob_score = threshold – loss_Ratio. This helps capturing how

far above or below an agent's performance was from the acceptable standard. The resulting lob_score was then normalized using min-max scaling to ensure comparability across agents and LOBs, regardless of the raw score magnitude.

- To understand the agent's claims handling efficiency, I calculated two key metrics:
    - **Frequency** = Number of claims / Number of policies
    - **Severity** = Claim amount / Number of claims

  These two metrics were combined additively to reflect total claims exposure and then normalized to generate a single claims_score. This approach penalizes agents with high frequency and/or high severity, highlighting inefficiencies in their business.
- To assess how well agents grew their book of business over time, I calculated the growth rate in Net Earned Premium (NEP) from the previous year. For each agent-LOB combination, I computed: nep_growth = (current_year_nep - previous_year_nep) / previous_year_nep
  Where data was missing or zero in the previous year, I handled the division gracefully to avoid errors and filled missing values with 0. This growth rate was then normalized to produce Nnep_growth, representing the relative growth performance.
- The final performance score for each agent per year per LOB was calculated as a weighted sum of the three normalized components: final_score = (0.4 * profitability_score) + (0.3 * claims_score) + (0.3 * GWP_growth_score)
  This score aggregates performance across profitability, efficiency, and growth, allowing for direct comparison of agents across different contexts.
- To streamline the entire scoring process, I implemented modular functions for each step—profitability, claims, GWP growth, and final score. These functions were executed sequentially through a single processing pipeline (process_agent_score()), which ensured clarity, reusability, and easy adjustments in the future. This design also made it easier to validate intermediate outputs at each stage and ensured consistent application of business logic.

```python
# Weights
WEIGHT_PROFITABILITY = 0.40
WEIGHT_CLAIMS = 0.30
WEIGHT_GWP_GROWTH = 0.30

# Normalize column
def normalize_columns(col):
    min_val, max_val = col.min(), col.max()
    return (col - min_val) / (max_val - min_val) if min_val != max_val else pd.Series([np.nan]*len(col), index=col.index)

# Profitability calculation
def calculate_profitability(data):
    def assign_lob_score(row):
        threshold = lob_thresholds.get(row['lob'], 0.75)
        return threshold - row['loss_ratio']

    data['lob_score'] = data.apply(assign_lob_score, axis=1)
    data['Nlob_score'] = normalize_columns(data['lob_score'])
    return data

# Claims score (no penalty)
def calculate_claims(data):
    data['severity'] = data['severity'].fillna(0)
    data['frequency'] = data['frequency'].fillna(0)
    data['claims_score'] = normalize_columns(data['frequency'] + data['severity'])
    return data

# GWP growth calculation
def calculate_nep_growth(data):
    data = data.sort_values(by=['agent_name', 'lob', 'account_yr'])
    data['prev_nep'] = data.groupby(['agent_name', 'lob'])['nep'].shift(1).fillna(0)
    data['nep_growth'] = ((data['nep'] - data['prev_nep']) / data['prev_nep'].replace(0, np.nan)).fillna(0)
    data['Nnep_growth'] = normalize_columns(data['nep_growth'])
    return data

# Final score
def calculate_final_score(data):
    data['final_score'] = (
        WEIGHT_PROFITABILITY * data['Nlob_score'] +
        WEIGHT_CLAIMS * data['claims_score'] +
        WEIGHT_GWP_GROWTH * data['Nnep_growth']
    )
    return data
```

```python
# Main pipeline
def process_agent_score(data, lob_thresholds):
    data = calculate_profitability(data)
    data = calculate_claims(data)
    data = calculate_nep_growth(data)
    data = calculate_final_score(data)
    return data

# Assuming N is your DataFrame
data = N.copy()
# data['Nfrequency'] = normalize_columns(data['frequency'])
# data['Nseverity'] = normalize_columns(data['severity'])
data = process_agent_score(data, lob_thresholds)

# Show results
data[['account_yr', 'agent_name', 'lob','nep','nic','nop_t','noc', 'lob_score','frequency','severity','nep_growth','final_score']].head()
```

Fig 21. Code Snippet of Scoring logic

- We are going to next check out the agents divided with respect to accounting years which are coming under respective reporting lines.

Chapter 6 Conclusion and Future Scope

In our evaluation of three machine learning models (CatBoost, LightGBM and XGBoost) for premium pricing prediction analysis, we observed varying levels of performance across datasets. Although XGBoost achieved the highest R^2 score during training (0.946), it showed a slight drop in performance when tested on 2024 dataset, with its R^2 score declining to 0.86.

Despite its solid results, including lower MSE, MAE, RMSE values, we chose to move forward with CatBoost model for premium pricing prediction. This decision is mainly based on CatBoost's superior handling of categorical variables, robustness and stability, which are essential in the motor insurance premium pricing context. Furthermore, CatBoost offers an excellent balance of performance and interpretability, making it more accessible for both data engineers and business users alike. Its ability of handle categorical data without extensive preprocessing gives it a significant operational advantage. CatBoost's consistency across validation sets and real-world applications, such as insurance workflows, makes it the more reliable choice for deployment. Currently, we are also working on deploying the prediction analysis for monitoring purposes, to be presented in dashboards.

Regarding the agent scorecard analysis, the project is ongoing, and while progress has been made in developing a comprehensive scoring model based on profitability, claims, and growth metrics, further validation and refinement are still required. Once finalized, the scorecard will be a valuable tool for evaluating agent performance across multiple metrics, helping to drive more informed decisions in the insurance business.

References

1. https://www.fairfax.ca/about-fairfax/
2. https://www.geeksforgeeks.org/feature-selection-techniques-in-machine-learning/
3. https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/
4. https://lightgbm.readthedocs.io/en/stable/
5. https://xgboost.readthedocs.io/en/stable/tutorials/model.html
6. https://catboost.ai/docs/en/concepts/tutorials
7. https://www.xenonstack.com/blog/data-analytics-in-insurance
8. https://earnix.com/blog/determining-the-best-insurance-pricing-strategy/
9. http://shapedthoughts.io/insurance-pricing-101-understanding-the-fundamentals-of-rate-making/
10. https://blog.exactbuyer.com/post/lead-scoring-model-for-insurance-agents?name=Maximizing%20Profits:%20An%20Efficient%20Lead%20Scoring%20Model%20for%20Insurance%20Agents&description=Increase%20profits%20and%20efficiency%20with%20an%20effective%20lead%20scoring%20model%20for%20insurance%20agents.%20Learn%20more%20about%20how%20it%20works%20and%20its%20benefits.