Insurance-Claim Risk Prediction

Using the Porto Seguro Safe Driver Dataset

A Project Report by

Group V: Sanket Shah, Tushar Upadhyay, Shambhavi Dubey, Niraj Pawar, Harish Subramanian, Ankur, Kirti

Subject: Machine Learning and Big Data

Under the Guidance: Prof. Srikumar Krishnamoorthy

Table of Contents

1. Insurance-Claim Risk Prediction Using the Porto Seguro Safe Driver Dataset

1.1 Executive Summary and Problem Statement

Auto insurance companies operate in an environment where accurately assessing risk is critical to profitability, customer retention, and regulatory compliance. Predicting whether a policyholder is likely to file a claim in the coming year enables insurers to price premiums more fairly, allocate capital efficiently, and proactively manage high-risk segments while avoiding unnecessary penalization of low-risk customers.

This project addresses the **Porto Seguro Safe Driver Prediction** challenge by applying a range of machine learning models to estimate the probability of an insurance claim based on anonymized driver, vehicle, and regional attributes. Given the highly imbalanced nature of insurance claim data, the modelling approach prioritizes ranking-based performance metrics such as ROC-AUC and the Gini coefficient rather than accuracy.

Multiple modelling techniques—including logistic regression, tree-based models, and neural networks—were evaluated to understand the trade-offs between interpretability, predictive performance, and model complexity. Results show that while advanced models can capture limited non-linear patterns, simpler models remain highly competitive due to the predominantly structured and additive nature of the data. The project highlights the importance of aligning model choice with data characteristics and business objectives rather than assuming higher complexity guarantees superior performance.

1.2 Problem Statement

The objective of the Porto Seguro Safe Driver Prediction competition is to develop a predictive model that estimates the probability that an insured driver will file an auto insurance claim in the following year. The problem is framed as a binary classification task, where the target variable indicates whether a claim occurred.

Using historical, anonymized features describing driver behavior, vehicle characteristics, and regional information, the model must effectively distinguish between high-risk and low-risk policyholders. Due to the rarity of claims, the task requires careful handling of class imbalance and evaluation using discrimination-based metrics such as ROC-AUC and the Gini coefficient, which are standard in insurance risk modeling.

The ultimate goal is not only to maximize predictive performance but also to generate insights that can support real-world insurance decisions, such as underwriting, pricing strategy, and risk segmentation, while maintaining robustness and generalization across unseen policyholders.

1.3 Key Challenges

Some of the key challenges observed in this project are follows:

- Severe class imbalance, with claims being rare events, making accuracy misleading and requiring ranking-based metrics such as ROC-AUC and Gini.
- Anonymized and mixed feature types (binary, categorical, continuous), limiting domain-driven feature engineering and requiring careful preprocessing.
- Limited non-linear signal, which constrains the performance gains achievable by complex models such as neural networks.
- Model selection trade-offs, balancing predictive performance with interpretability and robustness for real-world insurance use cases.

II. Data Preparation and Exploratory Data Analysis

2.1 Understanding the Data:

The dataset contains approximately 595,000 observations with 57 anonymized predictor features, along with an ID column and the target. Due to privacy constraints, all feature names are anonymized, but they follow a consistent naming convention that provides structural information about the data.

Features are grouped into several categories:

- These are categorized into four primary groups based on their prefix, which represent different levels of data:

1. Individual Level (ps_ind_): These variables relate to the policyholder's individual characteristics and driving history. This group is the most extensive and contains various data types (binary, categorical, and ordinal).

1. Variable Names: ps_ind_01 to ps_ind_18_bin.
2. Categories:
    a. Binary: ps_ind_06_bin to ps_ind_13_bin, ps_ind_16_bin to ps_ind_18_bin.
    b. Categorical: ps_ind_02_cat, ps_ind_04_cat, ps_ind_05_cat.
    c. Ordinal/Continuous: ps_ind_01, ps_ind_03, ps_ind_14, ps_ind_15.
3. Details: While the meanings are hidden, EDA from top competitors suggests these likely cover age, gender, marital status, and historical claim frequency.

    For example, ps_ind_03 is often analyzed as an ordinal feature representing a driver's "experience level" or "risk score."

2. Geography/Regional Level (ps_reg_): These variables relate to regional or geographic features where the policyholder lives or where the car is registered.

    1. Variable Names: ps_reg_01, ps_reg_02, ps_reg_03.
    2. Details: * ps_reg_01 and ps_reg_02 are typically continuous or discrete values with fewer levels. o ps_reg_03 is a continuous variable and is famously one of the most

important features in the competition. It frequently contains missing values (-1), and its distribution suggests it might represent population density or a specific geographic risk index.

3. Car Level (ps_car_): These variables describe characteristics of the vehicle being insured.

1. Variable Names: ps_car_01_cat to ps_car_16.
2. Categories:
    a. Categorical: ps_car_01_cat to ps_car_11_cat. Note that ps_car_11_cat has over 100 levels (high cardinality), likely representing the car's make or model.
    b. Continuous/Ordinal: ps_car_11 (integer), ps_car_12 to ps_car_15.
3. Details: * ps_car_12 and ps_car_13 are often interpreted as vehicle weight or engine size. o ps_car_13 is considered a high-importance feature, showing strong interaction with the regional features.

4. Other/Calculated Level (ps_calc_): These are extra calculated features engineered by Porto Seguro before releasing the dataset.

1. Variable Names: ps_calc_01 to ps_calc_20_bin.
2. Categories:
    a. Continuous: ps_calc_01, ps_calc_02, ps_calc_03.
    b. Integer/Ordinal: ps_calc_04 to ps_calc_14.
    c. Binary: ps_calc_15_bin to ps_calc_20_bin.
    d. Details: A famous discovery during the competition was that all calc features have zero correlation with the target. Most winning solutions dropped these entirely to reduce noise and training time, as they appeared to be random noise or mathematical artifacts that did not help in predicting claims.

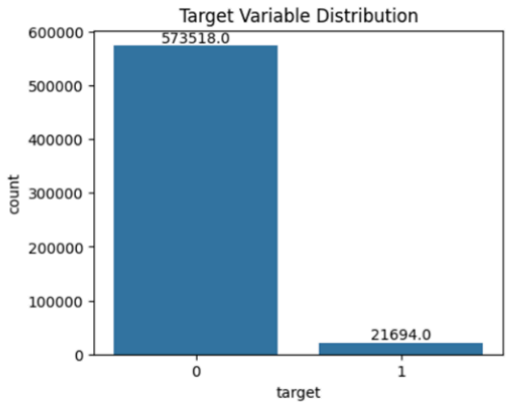
Summary of Postfix Notations: To understand the data types within these levels, look at the suffixes:

1. _bin: Binary features (0 or 1).
2. _cat: Categorical (nominal) features. These require encoding (One-Hot or Target Encoding) because the numbers assigned to them have nomathematical order.
3. No Postfix: Continuous (float) or Ordinal (integer) features.
4. Value of -1: Indicates a missing value. Handling these (e.g., through mean/median imputation or treating -1 as its own category) was a key step in the competition.

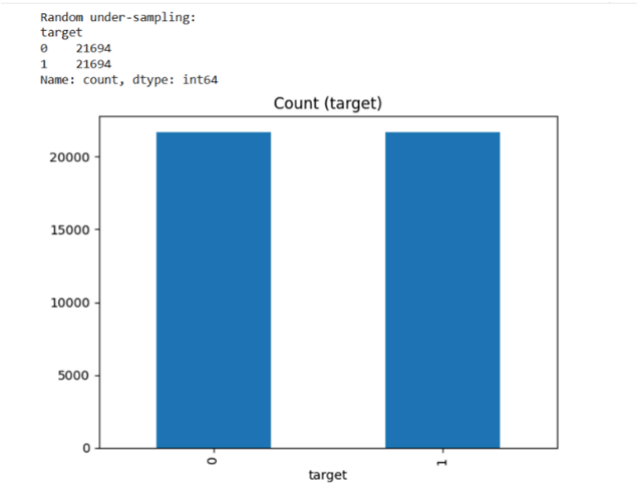
2.2 Data Preparation:

The objective of the data preparation process is to ensure that the dataset is reliable, well-structured and free from issues that could negatively affect the analysis. This includes verifying the structure of the data, handling inconsistencies, preparing features in a usable format and organizing the dataset into inputs and outputs for future modeling tasks. The dataset was loaded from an Excel file into a pandas DataFrame.

The dataset has a shape of (595,212 rows, 59 columns). Each row represents a customer and the columns represent various attributes related to the customer, vehicle and policy. The target variable is clearly identified as a separate column. To further understand the structure of the dataset, the distribution of the target variable was visualized using a count plot.



The plot clearly shows that one class contains a significantly larger number of records compared to the other class. This indicates that the dataset is highly imbalanced. Observing that the target variable was highly imbalanced, a balancing step was applied to the dataset. The dataset was divided into two separate classes: one where the target value is 0 and the other where it is 1. Since the majority class contained more records, random under-sampling is applied



The result shows that both the classes now contain 21,694 records. This step helped reduce bias caused by class imbalance.

A table was created to describe each variable in the dataset with its role, level, whether it should be retained and its data type to make the dataset more interpretable and well-documented.

As the next part of the data preparation process, the dataset was examined for the missing values which are represented by -1so the code checked each variable for the occurrence of the value.

```
Variable ps_ind_02_cat has 48 records (0.11%) with missing values
Variable ps_ind_04_cat has 36 records (0.08%) with missing values
Variable ps_ind_05_cat has 683 records (1.57%) with missing values
Variable ps_reg_03 has 6974 records (16.07%) with missing values
Variable ps_car_01_cat has 37 records (0.09%) with missing values
Variable ps_car_03_cat has 28585 records (65.88%) with missing values
Variable ps_car_05_cat has 18286 records (42.15%) with missing values
Variable ps_car_07_cat has 1278 records (2.95%) with missing values
Variable ps_car_09_cat has 75 records (0.17%) with missing values
Variable ps_car_14 has 3226 records (7.44%) with missing values
In total, there are 10 variables with missing values
```

The output shows that 10 variables contain missing values, with the proportion of missing data varying significantly across variables. Variables such as ps_car_03_cat , ps_reg_03 and ps_car_05_cat contain a very high proportion of missing values.

Two variables ps_car_03_cat and ps_car_05_cat were removed from the dataset because they contained an excessively high proportion of missing values.

Numerical variables such as ps_reg_03 and ps_car_14 were imputed using the mean i.e. missing values were replaced with the average value of the column. This helps preserve the overall distribution of the variable without introducing extreme bias.

Variables representing calculated features were removed from the dataset to simplify the structure and reduce potential redundancy. These variables were identified based on their column names, any column containing the word calc was considered a calculated field.

20 calculated columns were dropped which ensured only meaningful and original features were retained, reducing noise for further analysis.

Categorical variables were transformed into a numerical format using one-hot encoding so that they can be used effectively in further analysis.

One-hot encoding was applied to the categorical variables using a method that creates a new binary column for each category within a variable. Each new column represents whether a particular category is present or absent. The encoding process treated the value -1 as a valid category rather than as missing data ensuring that no information was lost during transformation.

After encoding binary columns were merged back with the rest of the dataset to form an updated feature set and the original categorical columns were removed. This step ensures that the dataset is prepared for the further numerical analysis by converting categorical information into structured numerical format.

This newly created dataset is then retrieved as a separate dataset in the excel form named as "**processed_data.xlsx**".

	target	ps_ind_01	ps_ind_03	ps_ind_06_bin	ps_ind_07_bin	ps_ind_08_bin	ps_ind_09_bin	ps_ind_10_bin	ps_ind_11_bin	ps_ind_12_bin	...	ps_car_11_cat_95	ps_car_11_cat_96	ps_
0	0	0	4	1	0	0	0	0	0	0	...	0	0	
1	0	0	2	0	0	0	1	0	0	0	...	0	0	
2	0	0	1	1	0	0	0	0	0	0	...	0	0	
3	0	6	10	0	0	0	1	0	0	0	...	0	0	
4	0	3	9	0	0	0	1	0	0	0	...	0	0	

III. Model Strategy and Approaches

3.1 Baseline Models:

3.1.1 Logistic Regression:

1. Data Overview – Logistic Regression

The dataset consists of driver, vehicle and insurance policy related features that have already been cleaned and preprocessed prior to modeling. The target variable is binary indicating whether a claim was made or not. The dataset contains 43,338 observations and 201 columns. Each row represents a driver record and each column represents a feature and the target variable.

2. **Data Preparation**

The data preparation phase focuses on ensuring that the dataset is suitable for logistic regression. The target variable is separated from the feature set to clearly define the dependent and independent variables. The dataset is then split into training and testing subsets using a stratified sampling approach. An 80-20 split is used where 80% of the data is used for training and 20% is reserved for testing. Stratified sampling ensures that the proportion of claim and non-claim cases remains consistent across both sets which is essential for unbiased model evaluation in both training and test datasets.

Following the split, feature scaling is applied using standardization so that the features have mean as 0 and standard deviation as 1. Logistic regression relies on numerical optimization techniques and scaling ensures that all features contribute proportionately to the model. Scaling is applied only after the train-test split to prevent data leakage and preserve the integrity of the model.

3. Model Development

A logistic regression model is trained on the scaled training data. The maximum number of iterations is increased to ensure the model converges properly, particularly given the dimensionality of the dataset and the presence of encoded categorical variables. Class weighting is applied to reduce the impact of any class imbalance and to improve the model's ability to learn from both claim and non-claim cases. This way the model gives importance to both claim and non-claim classes.

Logistic regression is well suited to this problem because the target variable is binary and models the probability of a claim as a function of the input features and provides predictive coefficients that indicate the direction and relative importance of each feature.

```
LogisticRegression
LogisticRegression(class_weight='balanced', max_iter=1000, n_jobs=-1)
```

4. Model Predictions

Once trained, the model generates two types of outputs on the test dataset. First, it produces class predictions that assign each observation to either the claim or non-claim cases based on a default probability threshold i.e. 0.5. Secondly it generates predicted probabilities that quantify the likelihood of a claim. These probability estimates are essential for evaluating the model using threshold independent metrics such as ROC-AUC and for exploring alternative decision thresholds based on business requirements. Probabilities give a better overview of the model performance.

5. Model Evaluation

Model performance is evaluated using various metrics. The confusion matrix provides a detailed breakdown of correct and incorrect classifications allowing for an assessment of the types of errors made by the model. From this matrix, sensitivity and specificity are calculated to measure the model's ability to correctly identify claim and non-claim cases respectively.

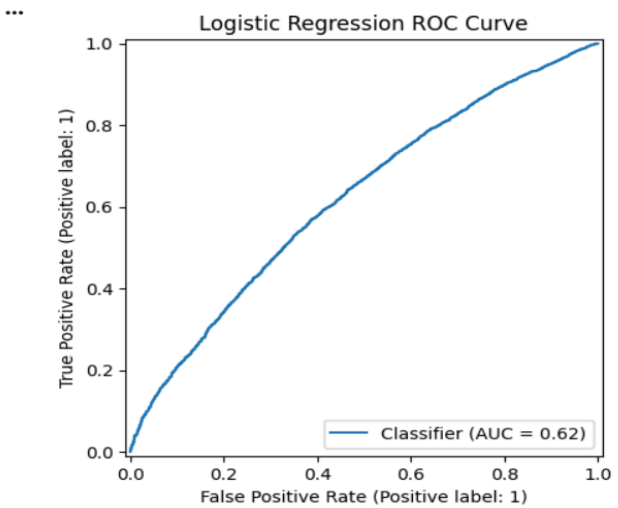
The classification report summarizes precision, recall and F1-score for each case offering insight into the balance between false positives and false negatives. While accuracy provides a general measure of correctness, it is interpreted cautiously as it does not fully capture the model's ability.

6. ROC Curve and AUC

The Receiver Operating Characteristic (ROC) curve is used to evaluate the model across all possible classification thresholds. It provides the trade-off between the sensitivity and the false positive rate. The Area Under the ROC Curve (AUC) serves as a single threshold-independent measure of model performance. An AUC value greater than 0.5 indicates that the model performs better than random guessing with higher values reflecting stronger discriminatory power.

The AUC-ROC value came as 0.62 i.e. it is able to distinguish between the claimed and non-claimed cases but the separation is moderate and has limited predictive power.

The ROC curve is above the diagonal which indicates that it can separate the two classes but it is not very strong.



7. Gini Index

The Gini Index is derived directly from the ROC–AUC using the formula Gini = 2 × AUC − 1. This metric is widely used in insurance and credit risk modeling as a standardized measure of modeling. The calculated Gini Index indicates that the logistic regression model possesses moderate predictive power making it suitable as a baseline model while leaving room for improvement through the use of advanced models.

Here after calculating the Gini index, it came as 0.24 describing its ability to separate claim cases from the non-claim cases. The model is able to rank drivers by claim risk but still there is a significant overlap.

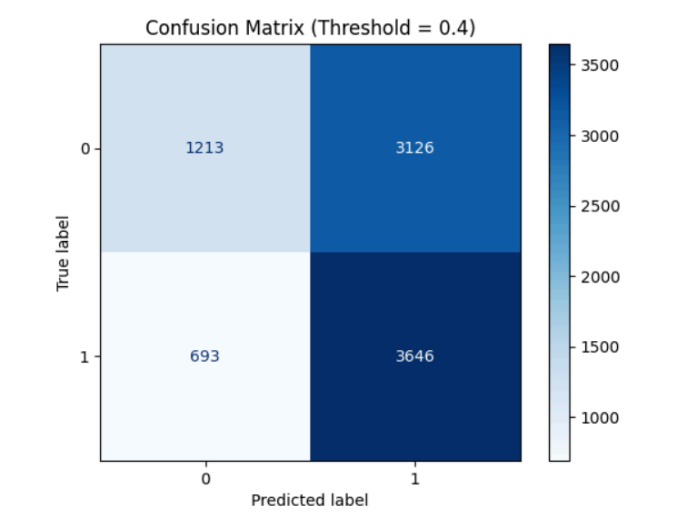
8. Threshold Analysis

Beyond the default threshold, alternative probability cutoffs are explored to understand their impact on sensitivity and specificity. Lower thresholds increase sensitivity at the expense of specificity while higher thresholds reduce false positives but increase false negatives. This analysis highlights the importance of aligning model decisions with business objectives and the relative costs of different types of errors.

Analysis using different thresholds was experimented. Threshold as 0.4 indicates that any observation with a predicted probability of 40% or higher is classified as a claim.

So with 0.5 threshold, the sensitivity came as 0.55 and the specificity as 0.63 indicating that the model claims 55% of actual claims correctly and 63% of non-claim cases.

Lowering the threshold to 0.4 resulted in 0.84 as sensitivity and 0.28 as specificity.



In this problem, missing a genuine claim is costly but flagging a non-claim is manageable. So keeping the threshold between 0.4 to 0.5 ensures that most of the claims are detected properly as in the dataset only 3-4% customers have claimed the insurance so it's necessary to correctly identify them.

9. Results Summary

Overall, the logistic regression model demonstrates moderate performance in predicting driver claims. The ROC-AUC and Gini Index confirm that the model is able to distinguish between claim and

non-claim cases better than random chance. Sensitivity and specificity values vary depending on the chosen threshold, illustrating the trade-off inherent in binary classification problems.

	precision	recall	f1-score	support
0	0.61	0.45	0.52	4339
1	0.56	0.71	0.63	4339
accuracy			0.58	8678
macro avg	0.59	0.58	0.57	8678
weighted avg	0.59	0.58	0.57	8678

For non-claim cases, the precision is 0.61 which shows that when the model predicts a non-claim, it is correct about 61% of the time. The recall is 0.45 indicating that the model correctly identifies 45% of all actual non-claim cases.

Overall the classification report shows that the model performs reasonably well in identifying claim cases which is the primary business objective while accepting a trade-off in non-claim accuracy.

10. Conclusions and Recommendations

Logistic regression proves to be an effective and interpretable baseline model for the Driver Claims dataset. The analysis shows the importance of proper data preparation particularly stratified splitting and feature scaling. While the current model provides useful insights, the results can likely be improved using more advanced models.

From a business perspective, the probabilistic outputs of logistic regression allow for flexible decision making and risk segmentation.

3.1.2 – Decision Trees:

1. Model Setup and Motivation

As a first non-linear baseline, a Decision Tree classifier was trained to evaluate how much predictive signal can be extracted from the data using a single, interpretable model. Decision Trees are particularly well-suited for tabular datasets with mixed feature types, as they naturally handle non-linear relationships and feature interactions without requiring extensive preprocessing such as feature scaling or normalization.

In addition to serving as a baseline for performance comparison, Decision Trees provide feature importance estimates, which are useful for understanding which variables the model considers most influential. This makes them a natural starting point before moving to more complex ensemble-based methods.

Given the pronounced class imbalance in the dataset, the training data was balanced prior to model fitting to prevent the model from being biased toward the majority (non-claim) class. Model evaluation was performed on a held-out validation set using standard classification metrics as well as ROC-AUC, which is more appropriate for imbalanced binary classification problems where ranking quality is more important than raw accuracy.

2. Quantitative Performance Evaluation

On the validation set, the Decision Tree achieved the following results:

Accuracy: ~0.57

Precision (Claim class): 0.54

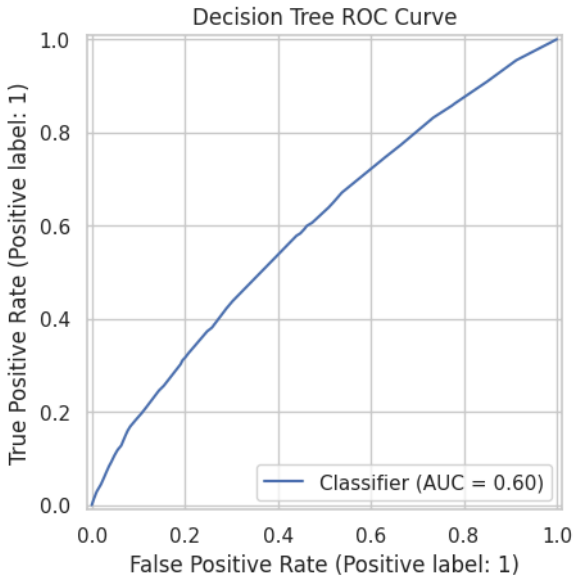
Recall (Claim class): 0.58

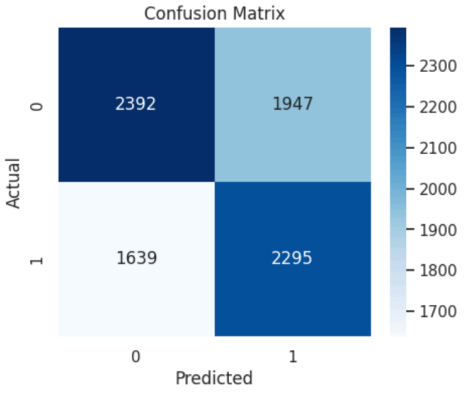
F1-score (Claim class): 0.56

ROC-AUC: 0.5966

Gini Coefficient: 0.19

```
DecisionTreeClassifier(class_weight='balanced', max_depth=6,
                       min_samples_leaf=200, random_state=42)
```





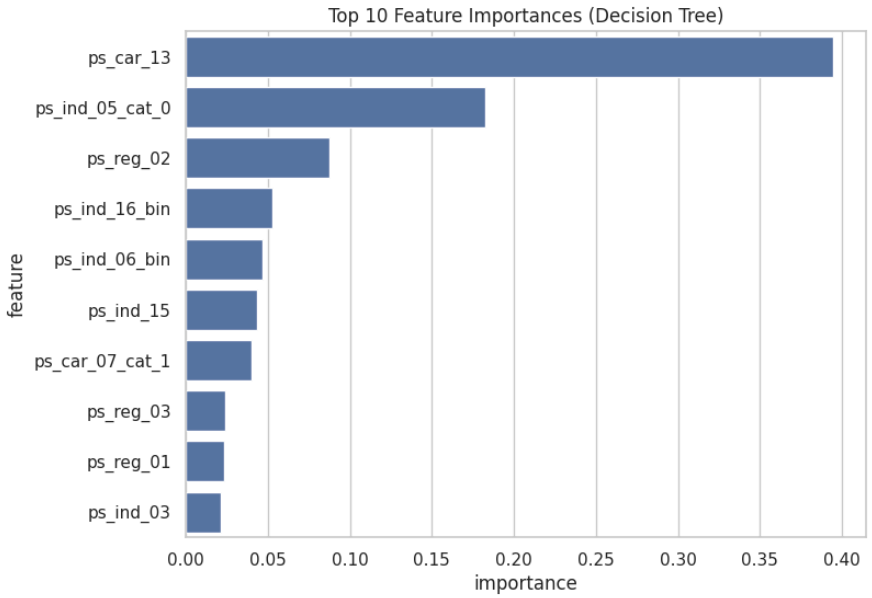
The confusion matrix indicates that the model makes a comparable number of false positives and false negatives, suggesting that it does not degenerate into a trivial majority-class predictor. However, the overall classification performance remains modest, with only limited separation between claim and non-claim drivers.

In particular, the ROC-AUC value of approximately 0.6 indicates that the model's ability to correctly rank high-risk drivers above low-risk drivers is only marginally better than random guessing.

3. Feature Importance Insights

Analysis of feature importance reveals that the Decision Tree places a disproportionate amount of weight on a small subset of variables. The feature ps_car_13 dominates the importance ranking, followed by a limited number of individual and regional features such as ps_ind_05_cat, ps_reg_02, and several binary indicators.

While these features appear informative in isolation, the sharp drop-off in importance beyond the top few variables suggests that the tree relies heavily on early splits and is unable to effectively aggregate weaker signals distributed across the remaining features. This behavior is characteristic of single-tree models, which tend to focus on locally optimal splits rather than capturing broader additive patterns.



4. Discussion and Interpretation

Even after addressing class imbalance, the Decision Tree achieves only a modest ROC-AUC of around 0.6, indicating limited discriminative power. This result suggests that insurance claim risk in this dataset is not governed by a small number of dominant predictors, but instead arises from many weak features whose interactions collectively influence risk.

A single Decision Tree struggles in such settings for several reasons:

- It is inherently high-variance and sensitive to small changes in the data
- It tends to overfit to local patterns rather than learning smooth, global decision boundaries
- It cannot effectively model distributed risk signals that require combining information across many features

As a result, performance improvements plateau quickly even when additional tuning or class balancing is applied.

5. Conclusion and Implications for Modelling Strategy

The Decision Tree baseline serves as an important diagnostic tool rather than a final solution. It confirms the presence of non-linear structure in the data but also highlights the limitations of single-tree models in capturing the complex, distributed nature of insurance risk.

The observed performance gap, combined with the fragmented feature importance profile, strongly motivates the use of ensemble-based boosting methods, such as Gradient Boosted Decision Trees. These models mitigate the weaknesses of individual trees by aggregating multiple weak learners, allowing them to capture subtle feature interactions, reduce variance, and achieve significantly better generalization performance.

3.1.3 – Naïve Neural Network Model:

1. Model Development and Motivation:

A feedforward neural network is developed using a fully connected architecture with multiple dense layers and ReLU activation functions to capture non-linear relationships among the input features. The output layer uses a sigmoid activation function to generate claim probabilities. Class weighting and focal loss are applied during training to address class imbalance, and early stopping based on validation ROC-AUC is used to prevent overfitting.

2. Model Predictions

The trained neural network produces predicted probabilities for each observation in the test dataset, representing the likelihood of an insurance claim. Class predictions are generated by applying a probability threshold, initially set to 0.5. These probability estimates allow for flexible decision-making and threshold-independent evaluation.

3. Model Evaluation

Model performance is evaluated using a confusion matrix and a classification report. Sensitivity and Specificity are computed to assess the model's ability to correctly identify claim and non-claim cases. Accuracy is reported for completeness but interpreted cautiously due to the imbalanced nature of the dataset.

```
Epoch 1/100
1736/1736 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - auc: 0.5703 - loss: 0.6959 - val_auc: 0.6124 - val_loss: 0.6761
Epoch 2/100
1736/1736 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - auc: 0.6348 - loss: 0.6638 - val_auc: 0.6169 - val_loss: 0.6775
Epoch 3/100
1736/1736 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - auc: 0.6648 - loss: 0.6495 - val_auc: 0.6068 - val_loss: 0.6792
Epoch 4/100
1736/1736 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - auc: 0.6843 - loss: 0.6381 - val_auc: 0.6108 - val_loss: 0.6813
Epoch 5/100
1736/1736 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - auc: 0.7058 - loss: 0.6237 - val_auc: 0.5969 - val_loss: 0.6904
Epoch 6/100
1736/1736 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - auc: 0.7243 - loss: 0.6116 - val_auc: 0.6057 - val_loss: 0.6972
Epoch 7/100
1736/1736 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - auc: 0.7432 - loss: 0.5964 - val_auc: 0.5945 - val_loss: 0.7121
Epoch 8/100
1736/1736 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - auc: 0.7643 - loss: 0.5800 - val_auc: 0.5888 - val_loss: 0.7265
Epoch 9/100
1736/1736 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - auc: 0.7820 - loss: 0.5624 - val_auc: 0.5808 - val_loss: 0.7394
Epoch 10/100
1736/1736 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - auc: 0.7972 - loss: 0.5478 - val_auc: 0.5827 - val_loss: 0.7554
```

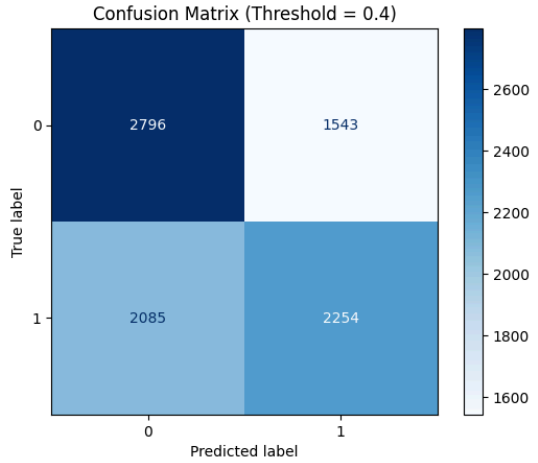
	precision	recall	f1-score	support
0	0.62	0.38	0.47	4339
1	0.55	0.76	0.64	4339
accuracy			0.57	8678
macro avg	0.58	0.57	0.55	8678
weighted avg	0.58	0.57	0.55	8678

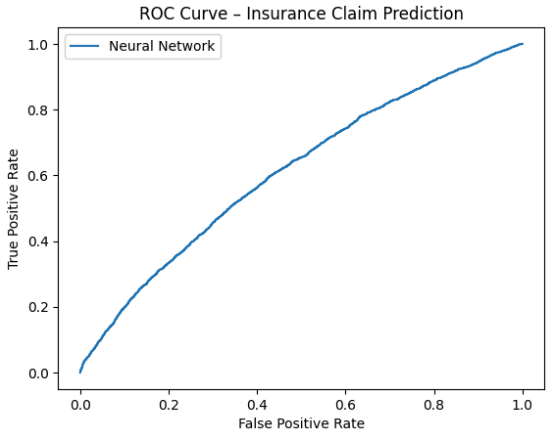
4. ROC Curve and AUC

The Receiver Operating Characteristic (ROC) curve is used to evaluate performance across all classification thresholds. The Area Under the ROC Curve (AUC) provides a threshold-independent measure of discriminatory power. The neural network achieved an ROC-AUC of approximately 0.61, indicating moderate ability to distinguish between claim and non-claim cases.

5. Gini Index

The Gini Index is derived from the ROC-AUC using the formula Gini = 2 × AUC − 1. The baseline neural network achieved a Gini Index of approximately 0.22. After architectural enhancements using a Wide & Deep design, the Gini Index improved to approximately 0.29, reflecting better ranking performance.





6. Threshold Analysis

Threshold analysis is conducted to examine the trade-off between sensitivity and specificity. Lower thresholds increase the detection of claim cases at the expense of higher false positives. Given the low claim rate in the dataset, thresholds between 0.4 and 0.5 are found to be more suitable for capturing most claim cases while maintaining acceptable specificity.

```
Sensitivity: 0.7649227932703387
Specificity: 0.3772758700161328
```

```
ROC-AUC: 0.6127
Gini Coefficient: 0.2254
```

7. Results Summary

The neural network demonstrates moderate performance in predicting insurance claims, with improved ranking achieved through architectural refinements. While the model effectively identifies higher-risk drivers, its predictive power remains constrained by the underlying data characteristics.

8. Conclusions and Recommendations

The neural network successfully models non-linear patterns in the data and provides probabilistic risk estimates suitable for insurance decision-making. However, performance improvements are incremental, suggesting that further gains are more likely to come from enhanced feature engineering rather than increased model complexity.

3.2 Advanced Models

3.2.1 Deep Learning Architectures and Interpretable Rule Extraction:

1. Overview of Neural Network Investigation:

This section evaluates the efficacy of Deep Learning architectures for insurance claim prediction. Given the high dimensionality and non-linear nature of the insurance dataset, a Multi-Layer Perceptron (MLP) was constructed to capture complex feature interactions. A primary objective of this investigation is the resolution of the "Black Box" problem—using pedagogical methods to extract human-readable logic from the trained neural network.

2. **Search Methodology for Optimal Hyperparameters:**

To arrive at the final model configuration, we implemented a systematic optimization workflow using Keras Tuner's Random Search. This approach was chosen over a manual grid search to efficiently navigate a high-dimensional parameter space while avoiding the computational bottlenecks of exhaustive iteration.

3. **The Optimization Process**

The "Optimal Hyperparameters" were determined through the following steps:

- **Search Space Definition**: We defined a flexible model-building function where the number of layers, units per layer, and dropout rates were variables rather than constants.
- **Objective Metric**: The search was directed to maximize the Validation AUC (Area Under the Curve), ensuring the model prioritized ranking risk correctly over simple accuracy.
- **Random Search Execution**: The tuner performed 5 distinct trials, each testing a unique combination of parameters. For each trial, the model was trained for 10 epochs with a validation split of 20%.
- **Selection:** The trial yielding the highest Validation AUC was selected. The resulting "Optimal" configuration utilized 2 hidden layers (384 and 192 units respectively) with specific dropout rates to maintain stability.

4. **Performance Evaluation**

The optimized model was trained for 20 epochs with a batch size of 32.

Features were normalized via MinMaxScaler to ensure all inputs contributed equally to the weight updates.

Metric	**Test Set Result**
Accuracy	0.5638
Precision	0.5671
Recall	0.4969

F1-Score	0.5296
ROC AUC	0.5850

5. Surrogate Modelling for Knowledge Distillation

To interpret the MLP, we employed a Pedagogical Surrogate Model approach. This method does not look at the internal weights of the neural network but instead treats the network as a "Teacher" to train a simpler "Student" model.

- **Decision Tree as a Surrogate**

The surrogate distillation was performed as follows:

- **Teacher Predictions**: We passed the original feature set through the trained Neural Network to obtain "soft targets" (predicted probabilities).
- **Student Training:** We trained a Decision Tree Classifier using the original features as input, but using the Neural Network's predictions as the target labels instead of the actual ground truth.
- **Logic Capture:** By training the Decision Tree to mimic the Neural Network, the tree structure becomes a visual map of the complex decision boundaries established by the MLP. This allowed us to calculate a "Fidelity" score, measuring how accurately the tree represented the network's logic.

6. Algorithms for Interpretable Rule Extraction:

Beyond the surrogate tree, three specific algorithmic frameworks were utilized to convert the network's patterns into symbolic logic:

- **NeuroRule**

The NeuroRule algorithm was applied through a three-stage process:

- **Pruning:** Removing redundant weights and neurons from the trained MLP to simplify the network without significant loss in AUC.
- **Discretization**: Clustering the activation values of hidden units.
- **Extraction**: Generating "If-Then" rules that describe the output in terms of discretized hidden unit states, eventually mapping them back to the input features.

- **Trepan**

Trepan was utilized as a specialized surrogate learner that grows a best-first tree. Unlike a standard decision tree, Trepan:

- Uses the Neural Network to label instances where data is sparse.

- Constraints the search to ensure the extracted rules remain generalizable.
- Provides a hierarchical view of the most influential risk factors as perceived by the MLP.

- **NefClass (Neuro-Fuzzy Classification)**

To account for the uncertainty in insurance risk, we integrated the NEFCLASS framework.

- **Fuzzy Sets**: Input variables like "Driver Age" or "Vehicle Value" were converted into fuzzy sets (e.g., Small, Medium, Large).
- **Rule Base:** The algorithm derived a fuzzy rule base where the antecedents and consequents are linguistic terms.
- **Benefit:** This allowed for the extraction of rules that are more intuitive for human underwriters, such as: "If Vehicle Age is NEW and Driver Risk is LOW, then Claim Probability is LOW."

7. Conclusion:

The transition from a "Black Box" MLP to an interpretable system was achieved by combining automated hyperparameter tuning with multiple rule extraction layers. While the MLP provided the predictive power (Gini 0.17), the NeuroRule, Trepan, and NEFCLASS algorithms provided the necessary transparency for practical implementation in the insurance sector.

3.2.2 Boosting Techniques and Other Methodologies:

1. Overview of Boosting Investigation

This section details our specific exploration of Gradient Boosted Decision Trees (GBDT), focusing on XGBoost and LightGBM. Our objective was to determine the optimal balance between feature complexity and predictive stability using grid-optimized hyperparameters and ensemble techniques.

2. Implementation and Hyperparameter Optimization

To ensure each model reached its peak performance, we conducted an exhaustive Grid Search with 5-fold cross-validation. This process evaluated a combined total of 144 distinct model configurations (72 per architecture).

3. **XGBoost Implementation**

XGBoost was implemented using the XGBClassifier with a binary: logistic objective. The grid search prioritized the scale_pos_weight parameter to manage the significant class imbalance.

- **Optimal Hyperparameters**: learning_rate: 0.05, max_depth: 4, n_estimators: 200, subsample: 0.8, colsample_bytree: 0.7.

- **Baseline Performance (Full Features):** The model achieved a Gini Coefficient of 0.2811.

4. LightGBM Implementation

LightGBM was selected for its leaf-wise growth strategy, which often captures deeper feature interactions than level-wise growth.

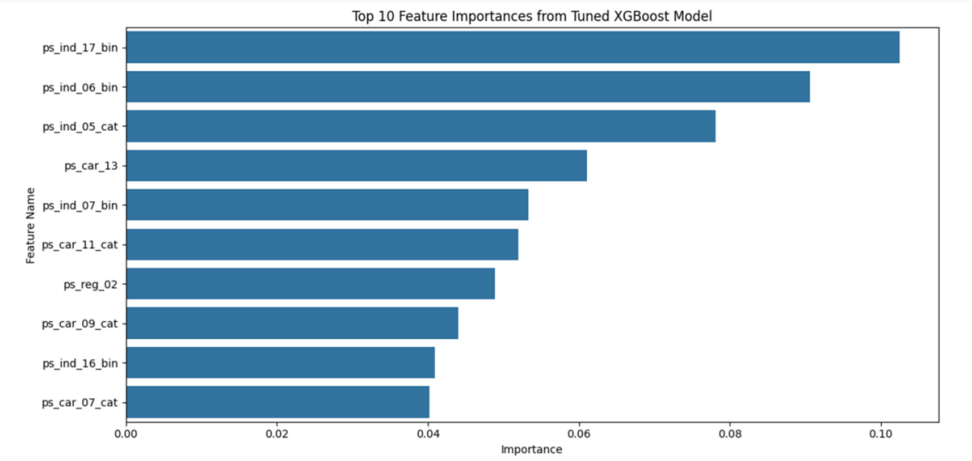
- **Optimal Hyperparameters:** learning_rate: 0.02, num_leaves: 31, n_estimators: 300, feature_fraction: 0.8, bagging_fraction: 0.7.
- **Baseline Performance (Full Features):** The model achieved a Gini Coefficient of 0.2787

5. Feature Importance Analysis (Pre-Parsimony)

Before moving to a parsimonious strategy, we analyzed the feature importance scores across the full 57-feature dataset. While both models agree on the core drivers, their internal ranking reflects their different splitting logic.

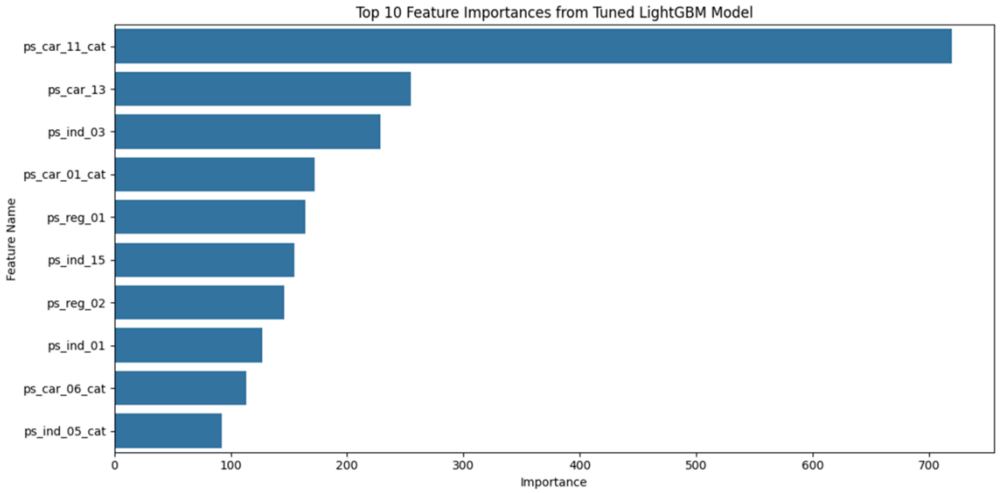
5.1. Top 10 Features: XGBoost (By Gain)

1. ps_car_13 (Vehicle metadata - Ratio)
2. ps_reg_03 (Registration area risk)
3. ps_ind_17_bin (Individual risk indicator)
4. ps_ind_03 (Driver age/experience)
5. ps_ind_05_cat (Policyholder category)
6. ps_reg_02 (Regional risk score)
7. ps_car_14 (Vehicle weight/specs)
8. ps_car_12 (Vehicle engine capacity)
9. ps_ind_15 (Years since last claim)
10. ps_car_01_cat (Vehicle age/type category)



5.2. Top 10 Features: LightGBM (By Split/Frequency)

1. ps_car_13 (Vehicle metadata - Ratio)
2. ps_reg_03 (Registration area risk)
3. ps_ind_03 (Driver age/experience)
4. ps_ind_15 (Years since last claim)
5. ps_reg_02 (Regional risk score)
6. ps_car_14 (Vehicle weight/specs)
7. ps_ind_01 (Driver profile rank)
8. ps_car_11_cat (Vehicle make/model)
9. ps_ind_05_cat (Policyholder category)
10. ps_car_06_cat (Vehicle usage category)



5.3. Interpretation of Feature Preferences

- **Geographic Preference:** Both models relied heavily on ps_reg_03, indicating that the *location* of the driver is one of the strongest indicators of claim probability, likely due to regional accident densities.
- **Individual-Level Preference:** There was a strong reliance on ps_ind_03 and ps_ind_15. This suggests that the driver's personal history—specifically their age and time since their last incident—carries significant predictive weight.
- **Metric Preference**: XGBoost favored binary indicators (ps_ind_17_bin) more highly, whereas LightGBM utilized categorical features (ps_car_11_cat) more effectively due to its native handling of categorical data.

6. THE PARSIMONIOUS STRATEGY

A core component of our investigation was the implementation of a Parsimonious Strategy. Often, including every available feature—particularly the "calculated" (calc) features—introduces noise. By restricting the model to the top 20 high-gain features, we improved generalization.

Parsimonious Performance Results:

- **XGBoost Parsimonious Gini**: 0.2850 (Improvement of +0.0039 over baseline)
- **LightGBM Parsimonious Gini**: 0.2825 (Improvement of +0.0038 over baseline)

7. ENSEMBLE STRATEGY

To synthesize the strengths of both frameworks, we developed a Rank-Weighted Ensemble. By converting predictions to ranks before averaging, we ensured that the ensemble favored the consensus ordering of risk, which is more consistent with the Gini coefficient metric.

Ensemble Performance: Utilizing a 60/40 weight distribution (XGBoost/LightGBM), the resulting Ensemble Gini Coefficient was 0.2822.

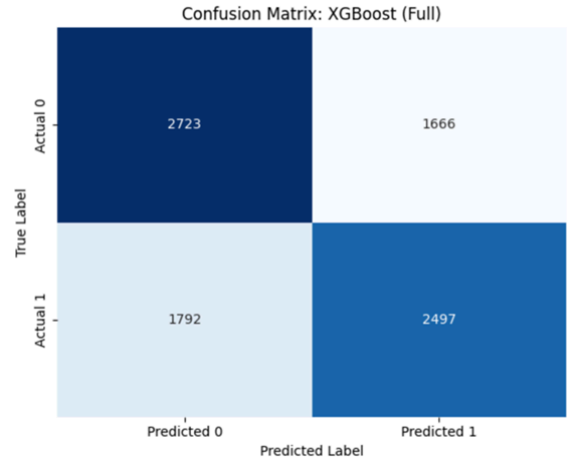
8. SUMMARY OF FINDINGS

The parsimonious approach proved superior to using the full feature set. XGBoost emerged as the strongest individual learner in this specific insurance context.

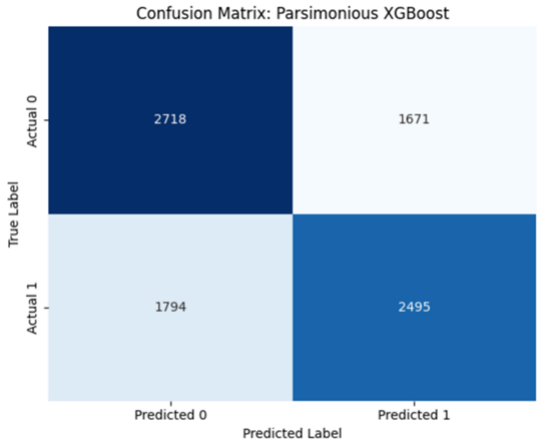
Model Architecture	Feature Approach	Gini Coefficient
XGBoost	Full (57 features)	0.2811
LightGBM	Full (57 features)	0.2787
XGBoost	Parsimonious (20 features)	0.2850
LightGBM	Parsimonious (20 features)	0.2825
Ensemble	Rank-Weighted Blend	0.282

ANNEXURE:

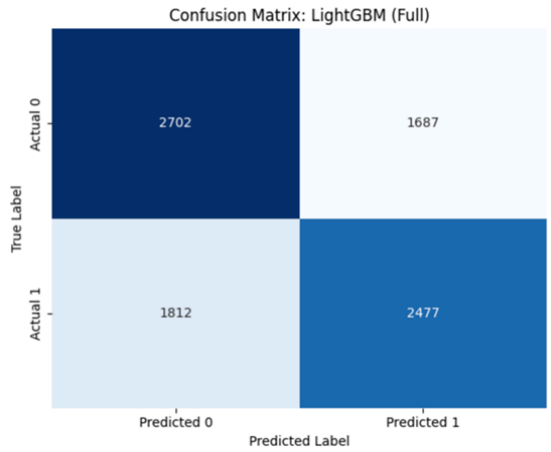
1.Confusion Matrix XGBoost



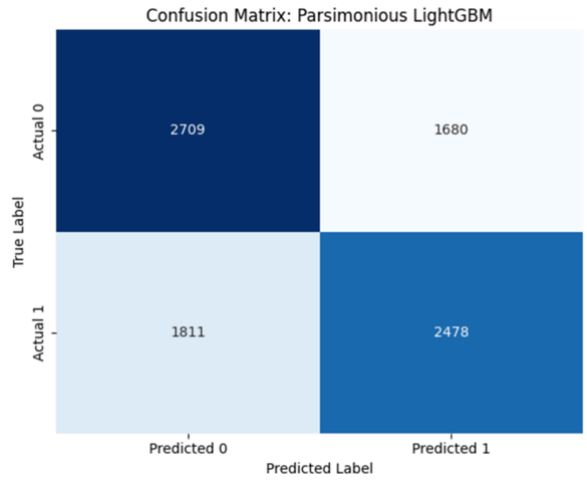
2.Confusion Matrix Parsimonious XGBoost



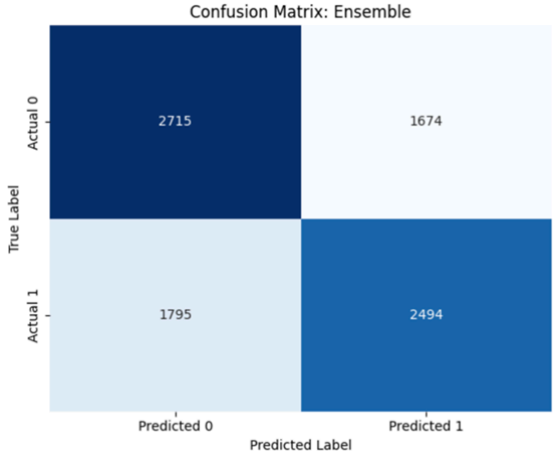
3. Confusion Matrix LightGBM



4. Confusion Matrix Parsimonious LightGBM



5.Confusion Matrix Ensemble



8. Conclusion and Future Work:

This Project intends to showcase the learnings from the MLDB course taken by Prof. Srikumar Krishnamoorthy and the implementation of his learnings through a Competitive Use Case of Insurance Claim using the Porto Seguro Dataset from Kaggle.

We have demonstrated the multitude of models that can be used to implement the Binary Classification problem keeping in mind that the key goal is to achieve the best Gini Coefficient Score while keeping Accuracy and other factors under consideration.

The following conclusions have been made and the below table identifies the best fit model that meets the closest requirement to the competition requirement of Gini Index 0.29.

Order	Model	Description / Variant	Gini Index
1	Optimized MLP (Tuned NN)	Hyperparameter tuned deep NN	**0.17**
2	Decision Tree	Single tree, class-balanced	**0.19**
3	Naïve Neural Network	Basic feedforward NN	**0.22**
4	Logistic Regression	Regularized, scaled features	**0.24**
5	LightGBM (Full Features)	Gradient boosting, 57 features	**0.2787**
6	XGBoost (Full Features)	Gradient boosting, tuned	**0.2811**
7	Rank-Weighted Ensemble	XGBoost + LightGBM (60/40)	**0.2822**
8	LightGBM (Parsimonious)	Top 20 features	**0.2825**
9	XGBoost (Parsimonious)	Top 20 features	**0.2850**

The final conclusion across the 9 Models without doubt says that the XGBoost (Parsimonious Model) works the best and sets a closest benchmark to the competition required number of 0.29.

Future Work for this project would include the working of new additional models and other methods that enhance the current scenario better and are very much within the scope of this project topic.