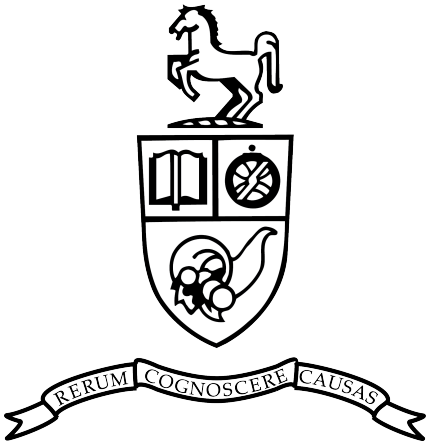
Car Insurance Conversion Prediction - Classification Model

Project Summary



University of Guelph, Guelph
November 2024

Project Summary submitted to University of Guelph in partial fulfillment of the requirements of DATA*6100

Arpan Sharma and Harsh Tiwari

Car Insurance Conversion Classification Prediction Project

Executive Summary

In this project, we are working on a classification task on a dataset of car insurance quotes, and to make predictions on an unseen test dataset. Our main task is to predict whether a person who requested the insurance quote took the policy. Two models **Logistic Regression** and **XGBoost (Gradient Boosting)** are compared in this project. Key insights:

- The dataset include training and test set which had 26 features and 117k rows combined with class imbalance with approximate ratio of 8:2 in training set.
- The Business Constraints linked with advertising insurance included Revenue of $5.50 for each True Positive. and Cost of $1 for each Positive prediction.
- XGBoost model has better ROC-AUC scores than Logistic Regression model but later outperformed on leader-board scores generating more revenue.

Data Pre-processing

In data preprocessing, we began by exploring the dataset to identify numerical and categorical columns. We combined the training and test sets into a single data frame to facilitate consistent handling of missing data.

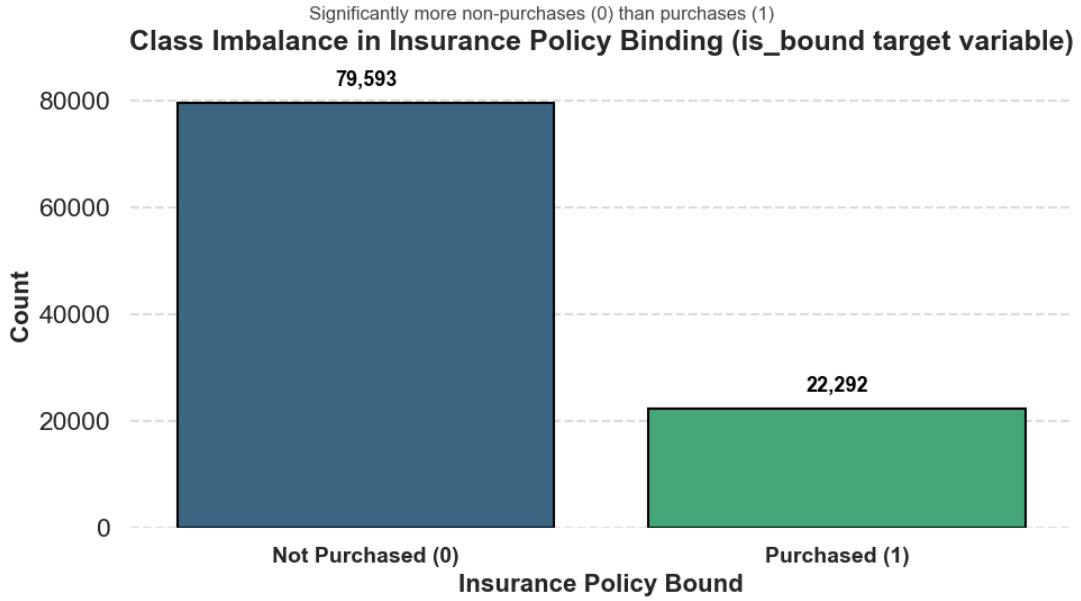


Figure 1: Bar chart showing the count of purchases (1) and non-purchases (0) in the target variable.

Initial exploration demonstrated class imbalance as seen in figure 1 showing one class represented in around 78% of the data where as other class representation is just 22%. As seen in figure 2 there are some columns with around 99% null data. To handle these, we converted them into binary features as they have impact on insuarance.

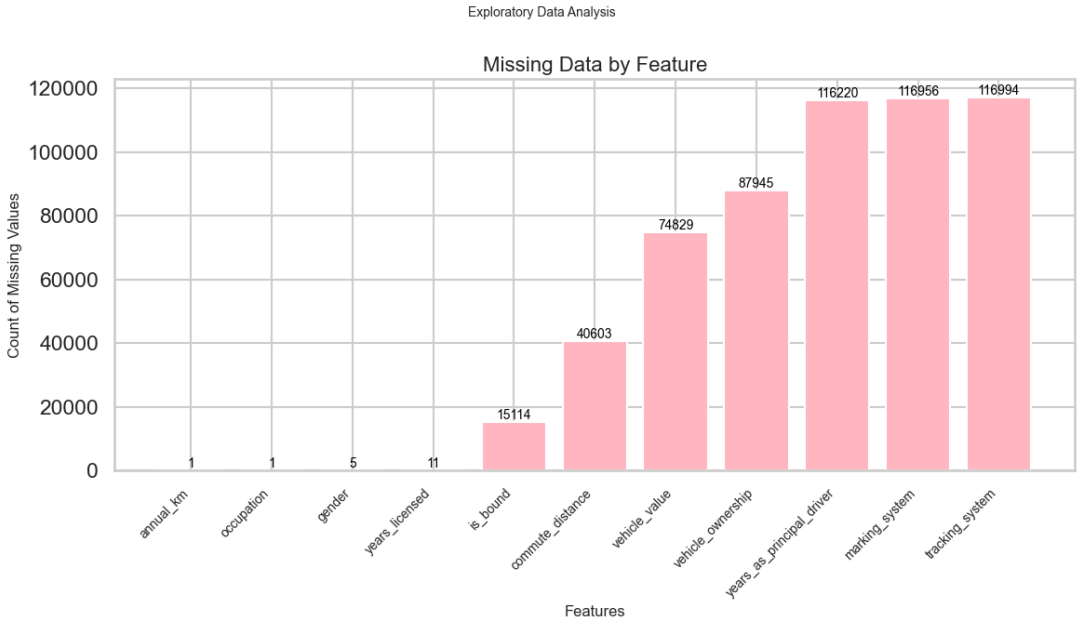


Figure 2: Bar chart showing the count of missing data for each feature in the dataset. Each bar is annotated with the count of missing values.

For rest columns, we used various methods like imputation with null representing categories using data information table as guide. For numerical columns with missing values, we imputed them using the median of the respective columns according to categorical columns that have close relationships with them according to data description for example commute_distance with medians of categories from vehicleuse column. We processed the values in the vehicle make column using regular expressions to standardize them. To enhance the accuracy of our models, we engineered new features to extract location information from area and postal code data and client age and vehicle age from year related features. Outliers were addressed by setting an upper limit and removing rows that exceeded it. we converted categorical features into numerical ones using one-hot encoding.

Modelling method 1 - Logistic Regression

Logistic Regression is a statistical model used for binary classification tasks. It models the probability that a given input X belongs to a particular category (e.g., class 1) using the logistic function. The logistic function outputs a probability between 0 and 1:

f(x) = $1\frac{}{1+e^{-x}}$

P(y = 1 | X) = $\frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_n X_n)}}$

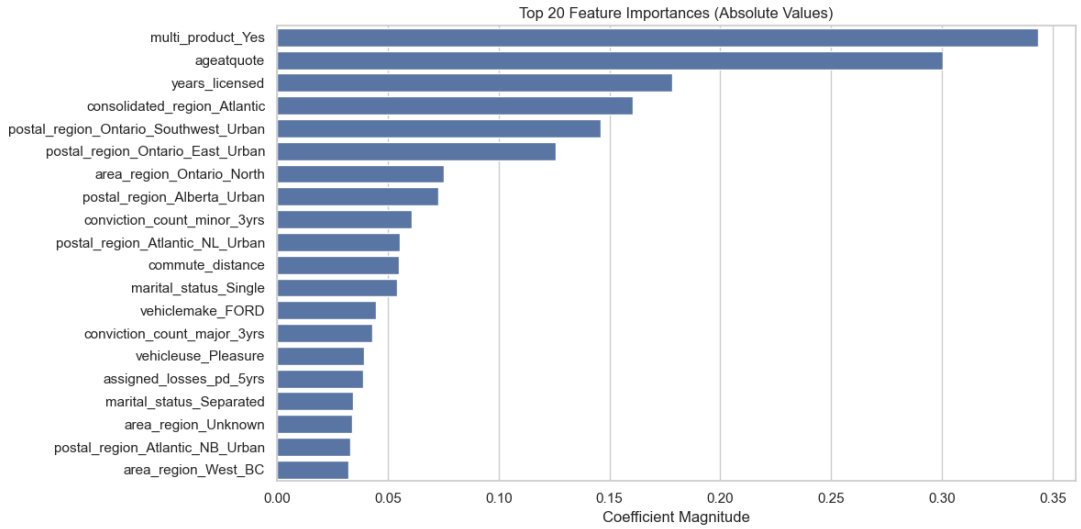
Here, β_0 is the intercept and β_i are the coefficients for each feature X_i. Logistic Regression estimates these coefficients by maximizing the likelihood of the observed data, effectively minimizing the negative log-likelihood loss function.

Predictions are made by comparing the estimated probability to a threshold (typically 0.5):

- If $P(y = 1 \mid X) \geq 0.5$, the predicted class is 1.
- Otherwise, the predicted class is 0.

Feature Selection and Model tuning: Using elastic net regularization, the model concurrently selects features and adjusts hyperparameters to guarantee interpretability and optimal performance. L1 (Lasso) and L2 (Ridge) penalties are combined in elastic net. By encouraging

sparsity and setting some feature coefficients to zero, the L1 component chooses the most important characteristics. Hyperparameter tuning is done via grid search with cross-validation, experimenting with different regularization strengths (*Cs*) and L1-to-L2 trade-offs (*l1_ratios*) to find the best combination for reducing log-loss. The sparse model created by this integrated method retains only the most important features, and the optimal hyperparameters maximize both generalization ability and prediction performance.



Model Validation scores: The model achieved a ROC-AUC score of 0.645, indicating moderate ability to distinguish between classes. With an average precision of 0.613, the model shows reasonable precision-recall performance. An EER threshold of 0.509 was determined for balanced error rates.

Modelling method 1 - XGBoost

XGBoost (Extreme Gradient Boosting) is an advanced implementation of gradient boosting designed for efficiency and scalability. It operates in three phases:

- **Training Phase:** Decision trees are built iteratively.
- **Prediction Phase:** Predictions for each instance are computed as the sum of the outputs from all decision trees.
- **Decision Threshold:** A probability score is compared against a decision threshold to make the final prediction.

The objective function of XGBoost is given by:

$$\text{Obj} = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

where:

- $L(y_i, \hat{y}_i)$ is the binary logistic loss function, measuring the difference between actual probabilities and predicted probabilities:

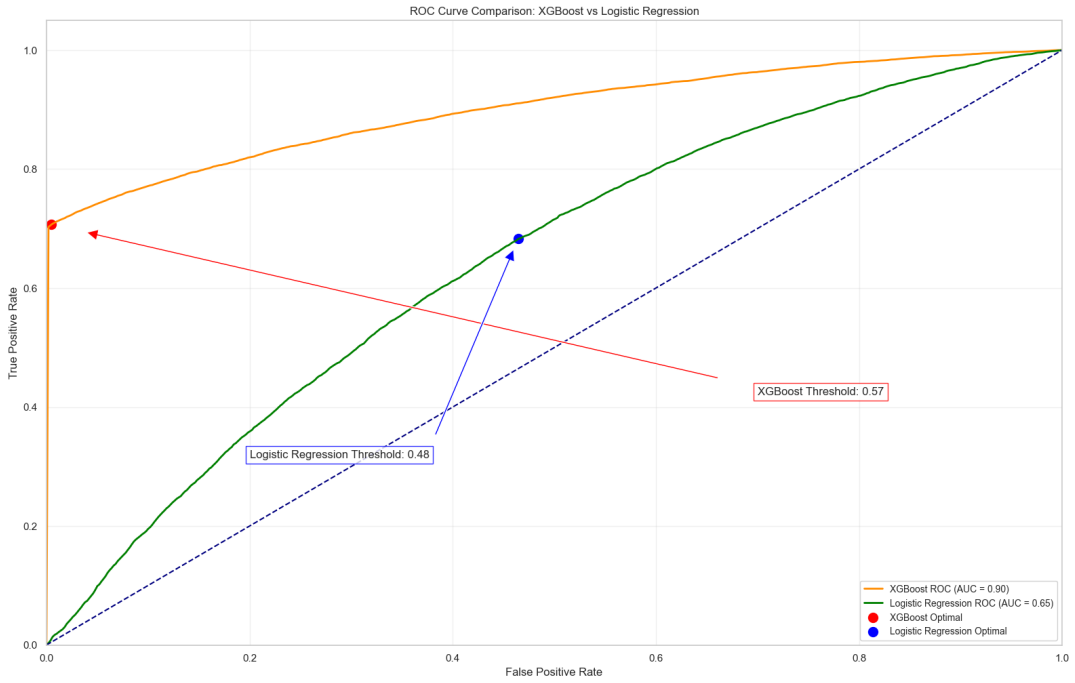
$$L(y_i, \hat{y}_i) = -[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

- $\Omega(f_k)$ represents the regularization term used to penalize the complexity of the trees.

Model Tuning: Model is tuned to find the best parameters using Random Search CV. Random Search uses every possible combination using the parameter grid given to find the best possible set of parameters which will work on the validation set.

Model Validation scores: The model achieved a ROC-AUC score of 0.896, indicating moderate ability to distinguish between classes. With an average precision of 0.926, the model shows reasonable precision-recall performance. An EER threshold of 0.452 was determined for balanced error rates.

Comparing the methods by looking at ROC curves



This ROC curve comparison demonstrates the performance of both XGBoost and Logistic Regression models. The Area Under Curve (AUC) indicates each model's discriminatory ability, with higher values showing better performance. Optimal thresholds (marked with dots) indicate the best balance between true positive and false positive rates for each model. These points represent where the trade-offs between sensitivity and specificity are optimized for practical application.

The graph below shows : This graph compares the Receiver Operating Characteristics (ROC) curves of XGBoost and Logistic Regression, illustrating their performance in distinguishing between classes. The orange line showing the ROC curve of XGBoost has an Area Under Curve (AUC) of 0.90, indicating strong discriminatory power, while the green line representing the ROC curve of Logistic Regression model has a lower AUC of 0.65. The dashed line represents a random classifier (AUC = 0.50), and both models surpass this baseline. The optimal threshold for XGBoost is 0.57 (red point), balancing sensitivity and specificity, whereas Logistic Regression's optimal threshold is 0.48 (blue point). Overall, XGBoost demonstrates a clear advantage in handling the classification task. Weak AUC is generally linked with weak performance but according to test set scores from leaderboard. Logistic Regression model performs better suggesting the model's predictions align more closely with the specific evaluation metric used on the leaderboard.

Metric	Logistic Regression	XGBoost
Test Set Accuracy (%)	58	77
Test False Negative Rate (%)	42	85
Test False Positive Rate (%)	40	7
Advertising Revenue (Cents Per Person)	13	5

Table 1: Test set Evaluation on Leadeboard

Conclusion

We evaluated both Logistic Regression and XGBoost models for the given task and observed that Logistic Regression outperformed XGBoost on the test data. The superior performance of Logistic Regression suggests that it is better suited to the underlying data structure, potentially indicating that XGBoost may have overfitted to the training data. This outcome highlights that the problem may favour a linear modeling approach. While XGBoost is highly effective in capturing complex, non-linear relationships and often achieves higher accuracy on training and validation sets, its flexibility can lead to overfitting in scenarios with simpler or more linear feature-target relationships. In contrast, Logistic Regression, with its inherent regularization and model simplicity, demonstrates better generalization to unseen data, making it more appropriate for this specific problem.