Comparative Analysis of ML Algorithms for Predicting Automobile Insurance Claims

1. INTRODUCTION

Northbridge sells automobile insurance policies, and customers pay an annual premium to insure their vehicle. We Assume that each customer has only one vehicle and one driver (i.e., the customer). Also, if any customer gets into an accident, they will always open a claim with Northbridge.

1.1. PROBLEM STATEMENT

The objective of this analysis is to build a binary classification predictive model that identifies whether a customer will submit a claim based on various customer, location demographics, and vehicle attributes.

1.2. DATASET

The dataset we are using to train & test our models consists of 382154 user records which also includes features such as age, gender, annual premium, previous insurance status, and vehicle history, with the target variable being whether a customer submitted a claim or not, annotated as 0 or 1

2. Data Preprocessing and Feature Engineering

2.1. PREPROCESSING

Since sometimes the data we have is not processed, Effective data preprocessing and feature engineering are critical steps in building a robust predictive model. The goal was to clean and transform the data to enhance feature relevance.

2.1.1. Handling Outliers:

All features except Annual Premium didn't have any outliers, the identified outliers in the Annual Premium feature, could introduce skewness and affect model performance. Log transformation was applied on the feature ensuring that the distribution remained informative without undue influence from high-premium anomalies.

2.1.2. Missing Values

The Dataset was complete with no missing values, we confirmed this by examining null values across each feature, ensuring no data gaps impacted model training.

2.2. FEATURE ENGINEERING

Feature engineering was essential to extract meaningful patterns from the data, especially for categorical and high cardinality features.

2.2.1. Binning

We grouped Age into brackets (e.g., 20–29, 30–39) to Reduce the dimensionality of this feature and to ultimately reduce the number of categories. As certain age group can be grouped together for collective prediction.

2.2.2. One-Hot Encoding

Features with low cardinality (<10) i.e. vehicle age and Age (after binning) were one-hot encoded. one-hot encoding was applied to prevent order assumptions and let the model capture distinct risk associated with each vehicle age category and customer age group.

2.2.3. Encoding High-Cardinality Features

High-cardinality features are important to encode because they can impact model performance, increase computation time, and make data analysis more complex.

- **Region Code**: Given the large number of unique values in Region Code, we opted for **target encoding**, which replaced each region with the average claim rate for that region.
- **Policy Sales Channel**: Policy Sales Channel had a high negative correlation with the target, which made it important to retain while managing high cardinality. We used **target encoding** to encode its values.

3. ALGORITHMS & METHODOLOGY

3.1. Model Selection Rationale

We will perform following 3 algorithms to find WSD of different in sentences:

- **XGBoost** Classifier: Known for its robustness and efficiency, XGBoost is a powerful gradient boosting algorithm that handles high-cardinality features and non-linear relationships well. It is also effective at managing class imbalance and feature importance, making it ideal for identifying key factors affecting claim likelihood.

- **Random Forest** Classifier: It is an ensemble method of decision trees as base learners; Random Forests capture a wide range of feature interactions and are less prone to overfitting due to their bagging approach. This model provided a strong benchmark for comparing predictive performance.

- **LightGBM** Classifier: like XGBoost, is designed for gradient boosting but with enhanced speed and memory efficiency. It is well-suited for large datasets and can handle missing values, making it an excellent choice for further optimizing predictive power and reducing training time.

3.2. Model Training & Evaluation

3.2.1. Data Splitting

The dataset was split into training (80%) and testing (20%) sets to evaluate generalization. Cross-validation was performed to ensure model consistency and avoid overfitting.

3.2.2. Hyperparameter Tuning

We used randomized search cv with StratifiedKFold and various feature selection methods to identify optimal hyperparameters for each model, focusing on parameters like learning rate, max depth, and number of estimators. This tuning helped enhance model accuracy while controlling complexity.

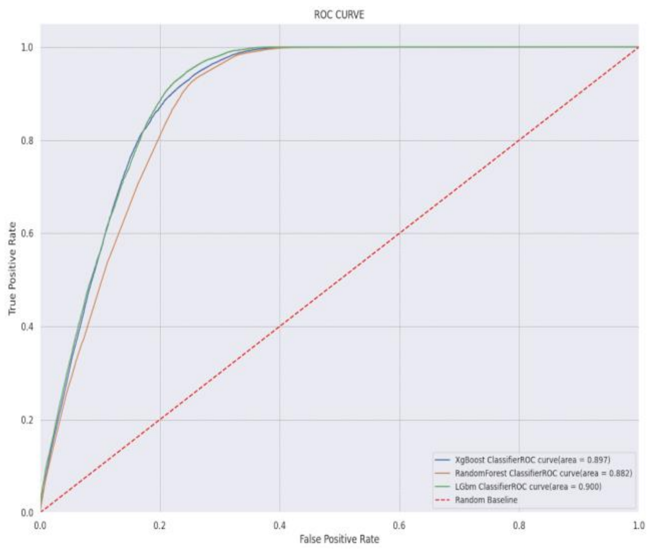
3.2.3. Evaluation Metrics:

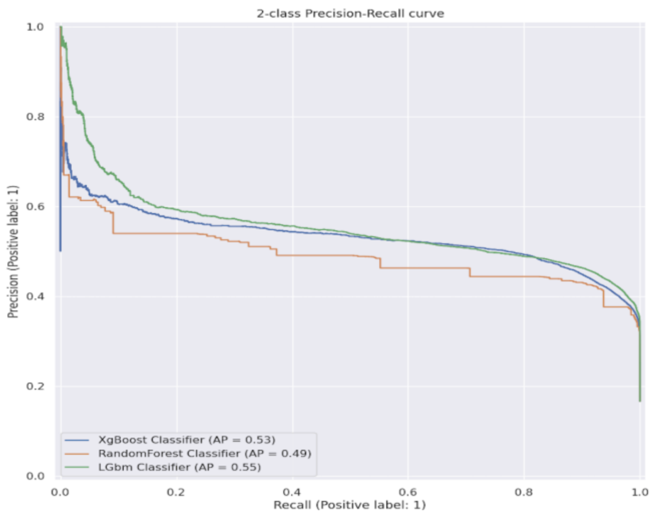
We used **accuracy** and **F1-score** to assess model performance, as both are informative for binary classification. Additionally, classification metrices like **confusion matrix** and **ROC** curve were used to interpret model performance on false positives and false negatives.

4. RESULTS & ANALYSIS

S.No	Model	Test Accuracy (%)
1	**XGBoost**	84.46
2	**Random Forest**	83.97
3	**LightGBM**	84.67

The XGBoost classifier achieved an overall test accuracy of 84.5% and performed well for the majority class (class 0). However, its recall for class 1 was relatively low, meaning that it struggled to identify the minority class correctly. The F1-score of 0.4554 for class 1 indicates moderate performance for the minority class, and the AP score of 0.53 shows moderate precision across recall values.





The Random Forest classifier achieved a slightly lower accuracy than XGBoost. Its performance on class 1 was weaker, with a low recall of indicating that it failed to capture a large portion of the positive cases. The F1-score for class 1 also reflects this challenge. However, the model performed well on class 0. The AP score of 0.49 and ROC AUC of 0.882 are slightly lower compared to XGBoost. The LightGBM classifier achieved the highest accuracy (84.7%) among the three models and showed an improvement in recall and F1-score for class 1 compared to the other classifiers. Its AP score of 0.55 and ROC AUC of 0.900 suggest that it provides a slightly better balance between precision and recall for the minority class The Precision-Recall curve and ROC curve highlights that LightGBM performed slightly better than XGBoost and Random Forest, achieving the highest average precision and area under the ROC curve.

5. Conclusion

Overall, the LightGBM classifier showed the best performance across multiple metrics, particularly in achieving a better balance between precision and recall for the minority class. This makes it a favorable choice for this dataset.

6. Limitations of the Analysis

The dataset lacks some features that might impact claim behavior, such as the customer’s driving history, accident severity, and credit score. This limitation reduces the model's predictive power. The data has an imbalance in the target variable which can lead to biased predictions towards the majority class. Although we attempted to handle this by using boosting algorithms however it still is a challenge.

7. Suggestions for Improvement

- Fine-tuning the LightGBM model for further improvement.
- Exploring additional external features, such as historical claim patterns, credit scores or calamities in region to enhance predictive accuracy.

8. References:

- https://www.kaggle.com/code/prashant111/lightgbm-classifier-in-python
- https://www.brokerlink.ca/blog/does-gender-affect-car-insurance
- https://www.thezebra.com/autoinsurance/driver/other-factors/
- https://scikitlearn.org/1.5/modules/generated/sklearn.feature_selection.RFECV.html

9. Reproducibility

Below are the hyper-parameters used for all the algorithms used in our experiments.

1. XGBoost Classifier

- n_estimators : 1000
- max_depth : 5
- learning_rate : 0.01

2. Random Forest Classifier

- n_estimators : 100
- min_samples_split : 5
- max_depth : None

2. LightGBM Classifier

- n_estimators : 500
- learning_rate : 0.01

10. Apendix

Test Results XGBoost:

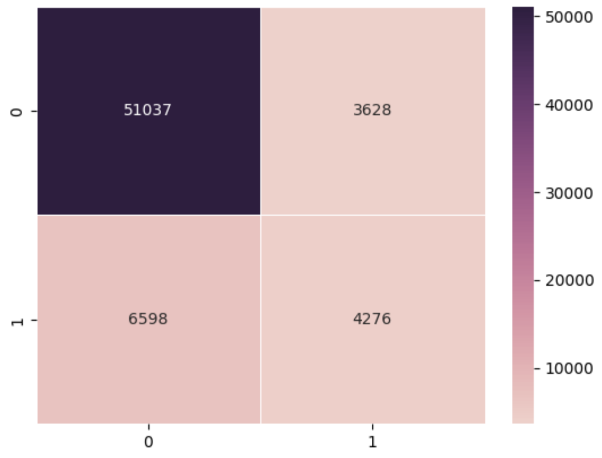


Fig: Confusion Matrix

Response	precision	recall	f1-score
No Claim (0)	0.885521	0.933632	0.908940
Claim (1)	0.540992	0.393232	0.455427
accuracy			0.843971

Test Results RandomForest:

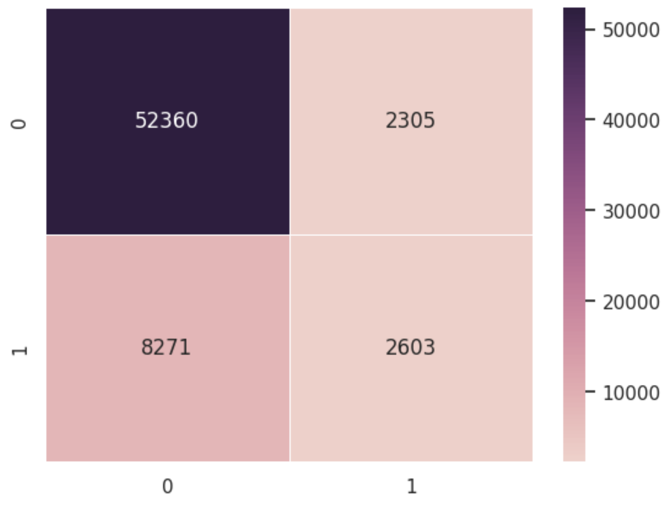


Fig: Confusion Matrix

Response	precision	recall	f1-score
No Claim (0)	0.863585	0.957834	0.908271
Claim (1)	0.530359	0.239378	0.329869
accuracy			0.838630

Test Results LightGBM:

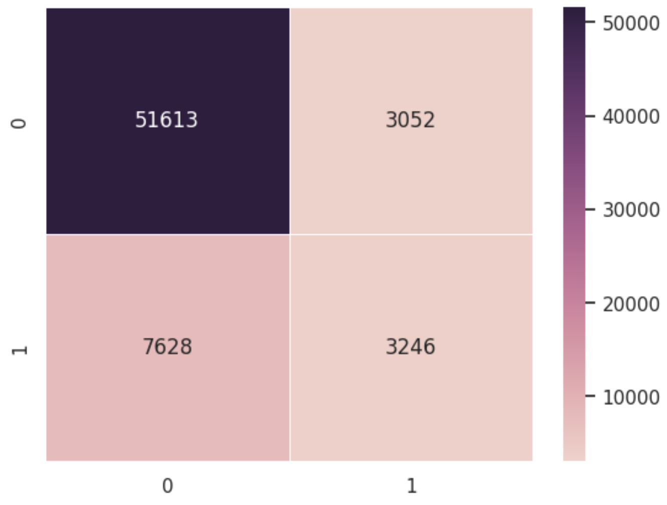


Fig: Confusion Matrix

Response	precision	recall	f1-score
No Claim (0)	0.88623	0.944169	0.906238
Claim (1)	0.570402	0.408510	0.468057
accuracy			84.6746