LightGBM for Imbalanced Auto Insurance Claim Prediction: A Comparative Study on the Porto Seguro Dataset

Jacky Chen
School of Computing and Academic Studies, BCIT
Vancouver, BC, Canada
jchen574@my.bcit.ca

Nicky Cheng
School of Computing and Academic Studies, BCIT
Burnaby, BC, Canada
nicky_cheng@outlook.com

Luying Cai
School of Computing and Academic Studies, BCIT
Burnaby, BC, Canada
lcai25@my.bcit.ca

Bryan Rachmat
School of Computing and Academic Studies, BCIT
Burnaby, BC, Canada
Rachmat.bryan@gmail.com

Abstract

LightGBM (Light Gradient Boosting Machine) is an efficient Gradient Boosted Decision Trees (GBDT) model. Its performance advantages come from core designs like histogram optimization, Gradient-based One-Side Sampling (GOSS), and Exclusive Feature Bundling (EFB). When applied to high-dimensional feature datasets and large-scale datasets, it demonstrates outstanding properties, such as fast training speed, low memory consumption, and robust predictive performance even with complex data distributions.

This study addresses the challenge of predicting auto insurance claims, using the Porto Seguro dataset from Kaggle. This dataset is typically characterized by sparse features, high missing value ratio, and class imbalance, which pose great difficulties for traditional models. Our experiments on this dataset show that LightGBM significantly outperformed all baseline models (including XGBoost and Random Forest). On our experimental setup, it achieved the highest AUC score of 0.6329, while maintaining training time of approximately 88.89 seconds. This paper also provides a comprehensive review of LightGBM, laying a solid foundation for future learning.

Keywords

LightGBM, Gradient Boosting, Auto Insurance, Imbalanced Classification, Porto Seguro

1 Introduction

Predicting auto insurance claim probability is a key task in the auto insurance industry, and an important part affecting risk management and pricing strategies. However, real-world data in this field often faces major challenges, including high dimensionality, sparsity, and a large number of missing values. Traditional methods usually struggle to strike a balance between prediction accuracy and computational efficiency when processing such complex datasets.

This study applies the LightGBM algorithm to this task. It not only relies on its own algorithmic characteristics, but also enhances performance through hardware optimization, such as efficient adaptation to multi-core computing architecture and reduction of memory access overhead. Eventually, it accelerates training speed significantly without sacrificing accuracy, thus achieving more precise prediction results.

Our main contribution is conducting a rigorous comparative analysis between LightGBM and classic tree-based models. We apply these models to the dataset of the Porto Seguro Safe Driver Prediction Challenge, to show how modern algorithms address limitations of traditional methods in industrial-scale classification tasks.

2 Background

2.1 Formal Problem Definition

In the Porto Seguro Safe Driver Prediction Challenge, we define this prediction problem as a supervised binary classification task. The dataset, denoted by $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, comprises n instances. Each instance consists of a d-dimensional feature vector, $x_i \in \mathbb{R}^d$, which includes characteristics of the driver and vehicle. The corresponding target label, $y_i \in \{0, 1\}$, indicates whether a claim was filed (1) or not (0).

The primary goal is to determine the function $F : \mathbb{R}^d \to [0, 1]$ that serves as an estimate for the conditional probability $P(y = 1 \mid x)$. This estimation is achieved by training the model to minimize the expected loss $\mathcal{L}$ across the dataset, as formalized by the equation:

$$\hat{F} = \arg\min_{F} \sum_{i=1}^{n} \mathcal{L}(y_i, F(x_i)). \tag{1}$$

Given the nature of the classification task, the Gini impurity criterion is utilized to evaluate and split nodes within the decision tree models. A key characteristic of the dataset is its high dimensionality and sparsity, with missing values explicitly denoted by -1.

2.2 Gradient Boosting Framework

Our work is based on the Gradient Boosting Decision Tree (GBDT) framework. GBDT is an ensemble method that combines multiple simple decision trees sequentially, building an effective predictive model by incrementally adding trees. In each training round, the new decision tree focuses solely on the prediction errors of the previous ensemble model. It corrects these errors by fitting *pseudo-residuals*—the gap between predicted values and actual targets.

The model update follows this formula:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x), \tag{2}$$

where $F_m(x)$ is the prediction result of the ensemble model after the m-th round, $F_{m-1}(x)$ is the result of the previous $(m-1)$-th

Figure 1: Conceptual illustration of GBDT / LightGBM ensemble structure (placeholder figure).

round, $h_m(x)$ is the new decision tree trained in this round, and γ_m is the learning rate (which controls the correction strength of the new tree).

Through iterative error correction, GBDT captures complex nonlinear relationships that single decision trees often fail to detect.

3 Related Work

3.1 Historical Context and Evolution

The evolution of predictive algorithms has progressed from single estimators to complex ensembles. Boosting improves predictive performance by iteratively combining multiple weak classifiers into a strong learner. A foundational implementation, AdaBoost, adjusts instance weights by increasing the weights of misclassified examples while decreasing the weights of correctly classified ones [4]. This reweighting mechanism forces the algorithm to focus on hard-to-predict cases in later iterations, combining weak learners through an additive model to form the final classifier.

Building on this foundation, Gradient Boosting further generalizes the framework by using the negative gradient of the loss function as the pseudo-residual target for each iteration [5]. Each new weak learner is trained to fit these gradients, correcting the residual errors accumulated by the previous ensemble. Although highly effective, traditional gradient boosting implementations face computational limitations, motivating optimized frameworks such as XGBoost [6] and LightGBM [7]. These modern systems introduce algorithmic and system-level optimizations, including sparsity-aware computation, histogram-based split finding, and dimension reduction, which enable scalable and efficient training on high-dimensional, sparse, or industry-scale datasets.

3.2 Traditional Algorithms

To evaluate the performance of LightGBM, we compare it against three representative baseline algorithms that capture different modeling assumptions and levels of complexity. Logistic Regression serves as a linear baseline: it is fast and interpretable, but it assumes additive and approximately linear relationships between features and the target, making it poorly suited for datasets with strong nonlinear interactions or high feature sparsity.

Decision Trees represent the simplest nonlinear alternative. They operate by recursively splitting the data into increasingly pure subsets, selecting the feature and threshold that best separate the classes at each step. They can model hierarchical feature interactions but suffer from high variance, instability, and a strong tendency to overfit, especially on imbalanced or noisy datasets.

Random Forest extends decision trees through bagging, reducing variance and improving robustness. However, because each tree is trained independently, Random Forest cannot correct errors made by earlier trees and often struggles to capture subtle patterns compared to boosting methods. Its computational cost also scales poorly with many sparse or high-dimensional features.

By comparing LightGBM with these three baselines, we highlight how gradient boosting overcomes the limitations of linear models (lack of nonlinearity), single trees (overfitting and instability), and bagging ensembles (limited sequential error correction), demonstrating the benefits of modern boosted tree frameworks.

3.3 Recent Developments

Recent work on LightGBM has started to focus not only on accuracy but also on interpretability. Many studies now use simple model-agnostic tools such as SHAP or LIME to explain which features influence LightGBM's predictions, making the model easier to understand in real applications. Some recent systems also combine LightGBM with other models, for example LSTM or KNN, to improve performance on time-series or high-dimensional tasks. A recent patent shows one such hybrid approach, where LSTM, LightGBM, and KNN are used together for traffic prediction, reflecting how modern boosting methods are being applied in more practical and interpretable ways [8].

4 Modern Algorithm Description

LightGBM is an ensemble algorithm composed of multiple components that work together to predict values. The main features of the algorithm are the following: decision trees, gradient boosting, Gradient-based One-Side Sampling (GOSS), and Exclusive Feature Bundling (EFB). Together, these algorithms combine to improve the training time of the model without making sacrifices in accuracy [1].

LightGBM's baseline algorithm is the decision tree. Decision Tree algorithms often use information gain or related criteria to decide split points for each branch of the tree. LightGBM's implementation of decision trees is based on histograms because the implementation performs better on sparse datasets. The better performance is a result of the algorithm taking feature bin values regardless of the feature value being zero [1]. The decision tree algorithm in LightGBM is enhanced by gradient boosting.

LightGBM builds on gradient boosting algorithms on decision trees. Gradient boosting algorithms calculate the steepest gradient descent direction for each iteration of the decision tree. Testing Gradient Boosted Decision Trees (GBDT) has revealed enhanced performance in accuracy when compared to contemporary decision trees [3]. GBDTs scan all data to decide on splitting data, which makes the algorithms time consuming with large datasets. LightGBM fixes this by removing features using Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) [1].

To reduce the number of data points used for split finding, LightGBM employs Gradient-based One-Side Sampling. This algorithm first checks how large each gradient is, then the largest gradients are kept intact, while the smallest gradients are randomly sampled. This changes the GBDT algorithm to operate over a subset composed of the largest gradients and a subsample of the smallest gradients, which reduces computation time without lowering accuracy [1]. LightGBM further enhances training time by using Exclusive Feature Bundling.

LightGBM also uses Exclusive Feature Bundling to remove features. This algorithm constructs a weighted undirected graph with each edge representing conflicts between features. Using the graph,

Figure 2: Illustration of GOSS and Exclusive Feature Bundling in LightGBM (placeholder figure).

the algorithm then sorts the features in descending order of weight. To finish bundling, the algorithm checks each feature to assign it to a bundle with a small conflict or creates a new bundle. The algorithm decides what features to merge into which bundles by adding offsets to the features' values. EFB changes the time complexity of histogram building from $O(\text{data} \times \text{feature})$ to $O(\text{data} \times \text{bundle})$ [1].

5 Application and Implementation

We apply LightGBM to Porto Seguro's Safe Driver Prediction Challenge on Kaggle. The challenge is to predict the chance that an auto insurance holder files a claim using machine learning [2]. The dataset that we are using contains many features and has sparse columns. In the dataset, missing data is represented by -1. We use LightGBM in a classification context [2].

The implementation of the model consists of loading both the training and validation datasets, cleaning the data of missing values, performing a train–validation split, training the model, and predicting labels. The training dataset consists of all feature columns. To clean missing values, the median of the column replaces any negative value. The train–validation split is 80% for training and 20% for validation. We use the training split to train the model and the validation split to predict labels.

Our biggest caveat in our implementation is not using the test dataset. This adaptation is necessary because the challenge has already closed. This caveat is also because the experiments focus more on how the algorithm is better than our baselines rather than solving the original competition problem. The next section goes into more detail about our experiments.

6 Experimental Evaluation

6.1 Dataset

Experimental Data Analysis. We use the Kaggle Porto Seguro Safe Driver Prediction dataset, with the following characteristics:

- Rows: ~595,000 training samples.
- Columns: 59 features + 1 binary target.
- Target column: `target` = 1 if the customer filed a claim, and `target` = 0 otherwise.
- Feature types: binary, categorical, continuous, and ordinal. Features that belong to similar groupings are tagged using postfixes, for example `bin` to indicate binary and `cat` to indicate categorical features. Features that do not include this are mostly continuous or ordinal.

Binary features: 17 features, with > 95% being 0. Categorical features: 14 features. Continuous or ordinal features: 26 features. Total: 59 features.

Missing values: values that are missing are not indicated using `N/A` but -1 instead. The features with the most missing values are `ps_car_14`, `ps_reg_03`, `ps_car_05_cat` and `ps_car_03_cat`. Missing values are handled using mean or median value imputation. We use an 80/20 train–validation split via `train_test_split(random_state=42)`.

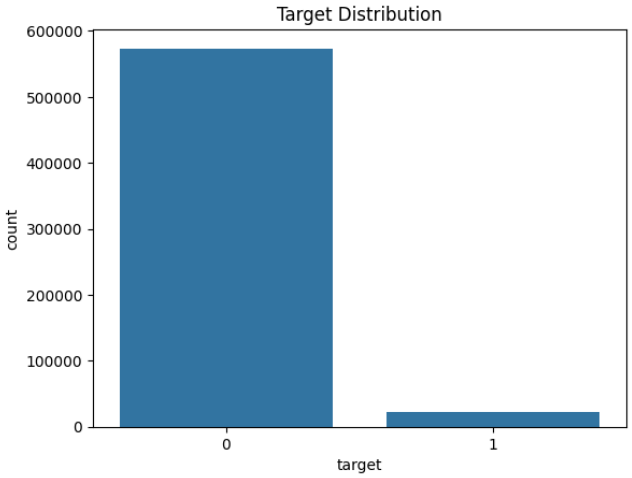

Figure 3: Porto Seguro Target Distribution

Imbalance / skew: the claim rate is very low (~ 3–4%), making this a strongly imbalanced classification problem. Only the training dataset contains labels. The test set is unlabeled and therefore not used for evaluation; all metrics are derived from a validation split within the training dataset.

6.2 Baseline Algorithms

To compare the modern LightGBM, the baseline algorithms we chose are Logistic Regression, Decision Tree, and Random Forest.

Linear/Logistic Regression. Linear and logistic regression are among the most basic models used in machine learning. They assume an approximately linear relationship between input and output. These models are fast, simple, and easy to interpret, but they cannot capture non-linear relationships or complex feature interactions [9]. As such, they serve as a baseline for minimum expected performance, and modern models are expected to outperform them.

Decision Tree. A single decision tree can capture non-linear patterns through recursive splitting, but it is highly prone to overfitting and instability [10]. Decision trees also serve as weak learners for ensemble and boosting methods, where multiple trees are combined to improve stability and accuracy.

Random Forest. Random Forest is an ensemble of decision trees trained on different subsets of data and features. This bagging mechanism reduces overfitting and improves prediction stability [11]. Random Forests often perform well without extensive tuning, but their averaging process cannot correct sequential errors and may be outperformed by gradient boosting in tasks requiring more fine-grained modeling.

6.3 Performance Metrics

Runtime. Runtime refers to the total time, measured in seconds, that a model takes to train on the dataset. It shows the computational cost and efficiency of an algorithm. Comparing training times across different algorithms helps pinpoint which models scale well and which might cause performance issues.

Table 1: Model performance comparison on validation set (placeholder numbers).

Model	ROC AUC	Train time (s)
Logistic Regression	0.6202	93.23
Decision Tree	0.0.6010	5.23
Random Forest	0.0.6285	133.52
LightGBM	0.6329	88.89

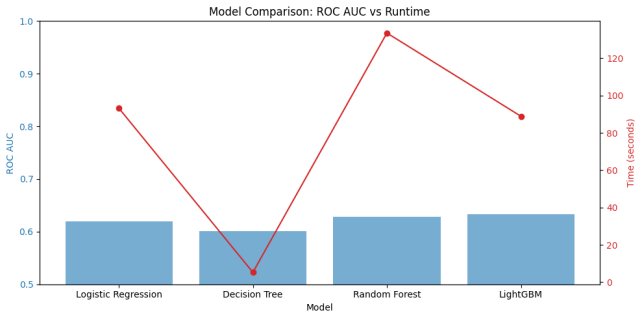

Figure 4: ROC curves of baseline models vs. LightGBM

Table 2: Model performance comparison on test set and Kaggle leaderboard.

Model	**Kaggle Public**	**Kaggle Private**
Logistic Regression	0.24312	0.23871
Decision Tree	0.20966	0.20429
Random Forest	0.25690	0.25219
LightGBM	0.26243	0.25679

ROC AUC (Receiver Operating Characteristic Area Under the Curve). ROC AUC measures how well a model distinguishes between positive and negative classes across all classification thresholds. It assesses the ability of predicted probabilities; a higher AUC indicates that the model consistently gives higher scores to true positive cases compared to true negatives. We chose this because this metric is especially useful for imbalanced or skewed datasets, such as this insurance claim dataset, where accuracy and other threshold-based metrics can be misleading when one class is much more common than the other. ROC AUC avoids this problem by being independent of thresholds and focusing only on the model's ability to separate the two classes.

6.4 Software / Hardware Environment

All experiments were conducted in a controlled and reproducible environment using Google Colab. This ensures practicality and makes collaboration more accessible. Furthermore, not much memory is needed, which is usually the challenge when using Google Colab for larger tree ensembles.

6.5 Kaggle Results

Kaggle submissions confirmed the validation results. LightGBM achieved the strongest public (0.26243) and private (0.25679) leaderboard scores, matching its top validation performance. Random Forest ranked second but remained slower and less accurate than LightGBM. Logistic Regression performed reasonably well for a linear model, outperforming the Decision Tree. The Decision Tree delivered the weakest scores, reflecting its limited ability to model the dataset's nonlinear patterns. Overall, the Kaggle outcomes reinforce that gradient boosting provides the best balance of accuracy and efficiency for this task.

7 Analysis

We evaluated four machine learning models: Logistic Regression, Decision Tree, Random Forest, and LightGBM using ROC AUC as the primary performance metric and training time as an indicator of computational efficiency. The results show a clear ranking of performance, with LightGBM achieving the highest AUC (0.6329) while maintaining a moderate training time of approximately 88.89 seconds. Despite being only slightly slower than Logistic Regression, LightGBM provides markedly better predictive discrimination due to its ability to capture complex nonlinear relationships and higher-order feature interactions. Its histogram-based split finding and leaf-wise tree growth strategy further enhance its efficiency and suitability for high-dimensional data such as the Porto Seguro dataset.

Random Forest ranks as the second-best model, achieving an AUC of 0.6285 but requiring the longest training time (133.52 seconds). The substantial computational cost reflects its ensemble structure, where multiple bootstrapped trees are trained independently. Nevertheless, Random Forest demonstrates the stabilizing benefits of bagging, outperforming both Logistic Regression and the single Decision Tree. In settings where robustness and reduced variance are priorities, Random Forest remains a strong choice.

The Decision Tree model yields the weakest performance with an AUC of 0.6010. This outcome is expected, as a single tree is prone to overfitting and highly sensitive to noise, particularly in sparse or imbalanced datasets like Porto Seguro. However, its extremely low training time (5.23 seconds) and interpretability make it a viable option when transparency or strict runtime constraints are required.

Logistic Regression performs competitively with an AUC of 0.6202 and a moderate training time of 93.23 seconds. As a linear model, it does not naturally capture nonlinear structure unless extensive feature engineering is performed. Nonetheless, it remains a stable and interpretable baseline model, especially when avoiding overfitting is important.

Overall, the empirical findings align with theoretical expectations: ensemble methods outperform single trees; boosting approaches such as LightGBM generally surpass bagging methods when nonlinear structure is present; and implementation-level optimizations in LightGBM result in a favorable balance of speed and accuracy. The training-time differences—particularly between Random Forest and LightGBM—illustrate how algorithmic design and efficiency influence practical model performance beyond predictive accuracy alone.

8 Discussion

LightGBM outperforms the traditional models primarily because of its gradient boosting framework, which incrementally corrects the residual errors of previous trees, producing a more refined and discriminative model. LightGBM's design choices, such as histogram-based splitting, leaf-wise growth, and the ability to handle sparse and high-dimensional input efficiently, contribute to both its superior accuracy and its fast training time. These optimizations make LightGBM especially suitable for large, imbalanced, or noisy datasets such as Porto Seguro.

However, this performance comes with trade-offs. LightGBM is considerably less interpretable than a simple decision tree. While Random Forest is also difficult to interpret globally, LightGBM often requires additional tools to explain predictions in a trustworthy way. Boosted models can produce poorly calibrated probabilities, prompting the need for additional post-processing. LightGBM also has a higher risk of overfitting if not properly regularized or tuned, making hyperparameter selection and early stopping crucial components of model development.

In terms of practical adoption, LightGBM is the strongest choice when you need both high predictive accuracy and fast inference. It works especially well in production systems that can handle more advanced models and where explainability can be provided through additional tools. Random Forest is still useful when you want a model that is more stable and easier to reason about at a feature level, even if it is not the most efficient. In very simple or highly regulated settings where transparency, auditing, or ease of explanation matter more than accuracy, a basic model like a shallow decision tree may still be the preferred option, even though its performance is lower.

One unexpected finding in this study was the particularly poor performance of the Decision Tree, which achieved an almost random AUC. This result reinforces how inadequate single trees can be for complex pattern recognition, especially in highly imbalanced datasets. Another notable observation is that Random Forest, despite being an older ensemble method, required significantly more computation time than LightGBM while still producing lower accuracy, highlighting that algorithmic evolution has produced models that are not only more accurate but also substantially more efficient.

9 Conclusion

This study compared four tree-based machine learning methods: Logistic Regression, Decision Tree, Random Forest, and LightGBM for predicting next-year insurance claims using the Porto Seguro dataset. LightGBM delivered the strongest overall performance, achieving the highest ROC AUC while also training much faster than Random Forest. Random Forest provided moderate predictive power but suffered from very long training times, and the single Decision Tree performed poorly, offering little ability to discriminate between claimers and non-claimers. These results emphasize that modern gradient boosting algorithms provide the best balance of accuracy and efficiency for this problem.

10 Future Work

Future work can expand this study in several meaningful ways. One direction is hyperparameter tuning. Using methods like grid search, random search, or Bayesian optimization to see whether more carefully optimized settings can push model performance beyond what we observed with default parameters. Another promising opportunity is additional feature engineering, such as creating interaction terms or using embedding-style encodings for high-cardinality categorical features, which are common in insurance datasets and may help models capture more complex patterns. The comparison could also be broadened by testing other modern boosting algorithms such as CatBoost or XGBoost to understand how LightGBM performs relative to alternative state-of-the-art methods.

On the evaluation side, incorporating additional performance metrics such as precision–recall AUC, calibration curves, or cost-sensitive measures would provide a more complete understanding of model performance, especially in imbalanced or high-risk prediction settings. Future work could also apply these models to other datasets such as credit risk or healthcare claims to test how well the findings generalize beyond the Porto Seguro data. Evaluating performance across multiple datasets would help confirm whether the advantages of LightGBM persist in other contexts and whether traditional models behave consistently across different distributions, ultimately making the results more reliable for real-world deployment.

References

[1] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, 2017.

[2] A. Howard, A. Moala, and W. Reade. Porto Seguro's Safe Driver Prediction. Kaggle Competition, 2017. https://kaggle.com/competitions/porto-seguro-safe-driver-prediction.

[3] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[4] Yoav Freund and Robert E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55, 1 (Aug. 1997), 119–139. https://doi.org/10.1006/jcss.1997.1504

[5] Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29, 5 (Oct. 2001), 1189–1232.

[6] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*. ACM, New York, NY, 785–794. https://doi.org/10.1145/2939672.2939785

[7] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. Curran Associates, Inc., 3146–3154.

[8] W. Liu, J. Zhao, G. Yuan, W. Fu, Z. Yu, W. Liu, J. Zhu, K. Luo, et al. A highway service-area traffic prediction method based on LSTM–LightGBM–KNN (in Chinese). Chinese Patent CN113344254A, 2021. Available at: State Intellectual Property Office of China.

[9] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. 2023. *An Introduction to Statistical Learning*, Chapter 3: Linear Regression. Springer International Publishing, 69–134. https://doi.org/10.1007/978-3-031-38747-0_3

[10] Yan-Yan Song and Ying Lu. 2015. Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, 27(2).

[11] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1, 5–32. https://doi.org/10.1023/A:1010933404324