Title: Comparative Performance of Random Forest vs. Gradient Boosting for Fraud Detection

Author : Zeba Samiya

Introduction

Problem Statement

Fraud detection in financial transactions is a significant challenge for banks and financial institutions worldwide, particularly due to the imbalance in fraud-related data. The rapid increase in online transactions makes it critical to employ efficient machine learning algorithms to identify fraudulent activities. Traditional rule-based systems struggle to adapt to the evolving nature of fraud, which is why machine learning techniques, particularly ensemble learning methods like Random Forest (RF) and Gradient Boosting (GB), are gaining traction. This research will compare these two popular machine learning techniques to determine their effectiveness in fraud detection, particularly focusing on accuracy, computational efficiency, and the handling of imbalanced datasets.

The hypothesis to be tested is as follows:
Primary Hypothesis: Gradient Boosting achieves higher fraud detection accuracy than Random Forest due to its sequential learning approach, which helps it better capture complex patterns.
Secondary Hypothesis: Random Forest is computationally more efficient and provides faster predictions, making it better suited for real-time fraud detection systems.

Purpose

This study is necessary because while previous research has demonstrated the efficacy of Random Forest and Gradient Boosting in fraud detection, there is limited research comparing these two methods on a real-world fraud detection dataset with a specific focus on computational efficiency and their ability to handle imbalanced data. By addressing this gap, this research will provide practical insights for financial institutions, helping them make informed decisions when selecting fraud detection algorithms.

In particular, this study will:

- Compare the performance metrics of Random Forest and Gradient Boosting in the context of fraud detection.
- Explore the computational efficiency and real-time applicability of both algorithms.
- Investigate how well each method handles imbalanced datasets, a common issue in fraud detection.

Scope and Limitations

The scope of this research will focus on:

- **Two machine learning algorithms:** Random Forest and Gradient Boosting, specifically XGBoost and LightGBM.
- **Data:** The Kaggle Credit Card Fraud Detection Dataset will be used, which consists of over 280,000 transactions, with only 0.17% fraud cases.
- **Metrics:** The study will evaluate the algorithms based on Precision, Recall, F1-Score, AUC-ROC, and Training Time.

Limitations:

- The research will not involve the implementation of deep learning models (e.g., CNNs or RNNs), which may also be useful for fraud detection.
- The study will use synthetic oversampling (SMOTE) to balance the imbalanced dataset but will not delve into alternative resampling techniques like undersampling or cost-sensitive learning.

Definition of Terms

- Random Forest (RF): A bagging-based ensemble learning method that uses multiple decision trees to improve model robustness.
- Gradient Boosting (GB): A boosting-based ensemble method that combines weak learners sequentially to reduce model error.
- SMOTE (Synthetic Minority Oversampling Technique): A technique used to address data imbalance by generating synthetic samples.
- AUC-ROC (Area Under the Curve - Receiver Operating Characteristic): A performance metric for classification models, measuring their ability to distinguish between classes.
- Hyperparameter Tuning: The process of optimizing machine learning model parameters (such as learning rate, tree depth, and estimators) to improve performance.
- Feature Importance Analysis: A method to determine which features contribute most to a model's predictions. Random Forest uses feature_importances_, while Gradient Boosting uses SHAP (Shapley Additive Explanations).

Literature Review

This literature review synthesizes key studies that examine the effectiveness of Random Forest and Gradient Boosting for fraud detection and related financial applications.

Citation	Focus Area	Relevance to Research
Sarker, Ananya, et al. (2024). "Credit Card Fraud Detection Using Machine Learning Techniques"	Comparison of Artificial Neural Networks (ANNs) and Logistic Regression (LR) for fraud detection.	Provides background knowledge on machine learning techniques for fraud detection but does not directly compare Random Forest vs. Gradient Boosting. Useful for understanding alternative ML approaches.
Bontempi, Gianluca. (2015). "Calibrating Probability with Undersampling for Unbalanced Classification."	Impact of dataset imbalance on fraud detection models, focusing on undersampling techniques.	Highly relevant since fraud detection data is imbalanced. This study helps in deciding whether SMOTE (oversampling) or undersampling should be used in this research.
Carcillo, Fabrizio, et al. (2018). "Streaming Active Learning Strategies for Real-Life Credit Card Fraud Detection"	Real-time fraud detection using Random Forest, Gradient Boosting, and XGBoost.	Provides insights into how both RF and GB perform in real-time fraud detection. Supports the secondary hypothesis that Random Forest is computationally more efficient.
Chen, Tianqi, & Guestrin, Carlos. (2016). "XGBoost: A Scalable Tree Boosting System."	Introduction of XGBoost, a Gradient Boosting model optimized for performance.	Essential for this research as it directly compares Random Forest vs. Gradient Boosting, showing why boosting methods may be better for fraud detection.
Zięba, Maciej, et al. (2016). "Ensemble Boosted Trees with Synthetic Features Generation in Application to Bankruptcy Prediction."	Comparing Random Forest and Gradient Boosting in financial classification tasks (bankruptcy prediction).	Highly relevant because bankruptcy prediction and fraud detection share similarities. Supports the hypothesis that Gradient Boosting offers better accuracy, while Random Forest is easier to optimize.

Research Plan

Framework / Environment / Required Tools

Environment Setup

The research will be conducted on the following hardware and software environment:

Hardware Specifications

- Device: MacBook Pro
- Processor: Apple M2 Chip
- Memory: 16GB RAM
- Storage: 512GB SSD
- Operating System: macOS 15.3.1 (24D70)

Software, Libraries, and Monitoring Tools:

Category	**Tools / Libraries**
Programming Language	Python (3.9 or later)
Development Environment	Jupyter Notebook (via Anaconda)
Machine Learning Libraries	Scikit-learn, XGBoost, LightGBM
Data Processing & Visualization	Pandas, NumPy, Matplotlib, Seaborn
Feature Importance Analysis	SHAP (Shapley Additive Explanations)
Imbalanced Data Handling	SMOTE (Synthetic Minority Oversampling Technique)
Monitoring & Experiment Tracking	MLflow (for logging experiments), TensorBoard (for visualizing model training)

This setup ensures efficient computation, scalability, and ease of implementation while performing model training and evaluation.

Disk Utilization Monitoring

Since machine learning model training can generate large temporary files, logs, and datasets, it is necessary to ensure that sufficient disk space is available throughout the research process.

Current Disk Utilization Status

Using the `df -h` command, the system has **262 GiB (~57%) of free space**, which is adequate for storing:

- The dataset (Kaggle Credit Card Fraud Detection Dataset).
- Preprocessed data and feature-engineered versions.
- Model checkpoints and training logs.

Disk Utilization Measurement Approach

To ensure storage availability and prevent system slowdowns, disk usage will be monitored regularly using:

1. **macOS Terminal Command (`df -h`)**: This will provide a snapshot of free and used disk space before and after model training.
2. **Python `psutil` Library**:
    - This script will log available disk space before, during, and after model training to track utilization:

      ```
      Import psutil
      print("Disk Usage:", psutil.disk_usage('/'))
      ```

3. **Periodic Log Cleanup**:
    - MLFlow experiment logs and temporary files will be deleted periodically to free up disk space:

      ```
      rm -rf ~/.cache
      ```

This monitoring ensures that **model training does not cause storage constraints**, especially given that **Gradient Boosting models may require more storage** due to checkpointing and iterative updates.

Data Acquisition

The dataset used in this research is the Kaggle Credit Card Fraud Detection Dataset, a widely used real-world dataset for fraud detection. The dataset consists of 284,807 transactions, out of which only 0.17% are fraudulent (492 fraud cases), making it a highly imbalanced dataset.

Dataset Details

- Total Transactions: 284,807
- Legitimate Transactions: 284,315 (~99.83%)
- Fraudulent Transactions: 492 (~0.17%)
- Features: 30 (including transaction amount, time, and anonymized variables)
- Class Label: Binary classification (0 = Legitimate, 1 = Fraudulent)

Data Collection & Preprocessing

1. Data Download & Loading: The dataset will be downloaded from Kaggle and loaded into Python using Pandas.
2. Feature Engineering: StandardScaler or MinMaxScaler will be applied to normalize transaction values.
3. Handling Imbalanced Data:
    - SMOTE (Oversampling): To create synthetic fraud examples and balance the dataset.
    - Baseline (Without Oversampling): To compare model performance with and without data balancing.
4. Splitting the Data:
    - 80% for training, 20% for testing using stratified K-Fold cross-validation to ensure class distribution is maintained.

Test Methodologies

This section details the structured approach for implementing and evaluating Random Forest (RF) and Gradient Boosting (GB) for fraud detection.

1. **Data Preprocessing**

- Load dataset and handle missing values (if any).
- Scale features using StandardScaler / MinMaxScaler.
- Apply feature selection to remove highly correlated variables.
- Split the dataset into train (80%) and test (20%).

2. **Model Implementation**

(A) Random Forest Setup

- Use sklearn.ensemble.RandomForestClassifier for training.
- Hyperparameter tuning:
    - n_estimators: Number of trees (100, 200, 500).
    - max_depth: Maximum tree depth.
    - min_samples_split: Minimum samples for node split.
    - class_weight='balanced' to handle data imbalance.

(B) Gradient Boosting Setup

- Implement XGBoost and LightGBM.
- Hyperparameter tuning:
    - learning_rate: Step size (0.01, 0.1, 0.2).
    - n_estimators: Number of boosting rounds.
    - max_depth: Tree depth (3, 5, 7).
    - subsample: Fraction of data used per iteration.

3. **Model Evaluation Metrics**

To assess the performance of Random Forest (RF) and Gradient Boosting (GB) for fraud detection, the following key evaluation metrics will be used. Each metric will be quantified using specific functions and methods in Python.

Metric	**What It Measures**	**Goal**	**How It Will Be Measured**	**What We Are Calculating**
Precision (%)	Measures the proportion of correctly identified fraudulent transactions out of all transactions classified as fraud.	Minimize false positives; ensure fraud alerts are accurate	`precision_score(y_true, y_pred)` from `sklearn.metrics`.	**Percentage (%)** of correctly classified fraudulent cases among predicted frauds
Recall (%) (Sensitivity)	Measures how well the model correctly identifies actual fraudulent cases.	Minimize false negatives; ensure fraud cases are not missed.	`recall_score(y_true, y_pred)`.	**Percentage (%)** of actual fraudulent transactions that were correctly detected.
F1-Score (%)	Balances Precision and Recall.	Ensure a trade-off between false positives and false negatives.	`f1_score(y_true, y_pred)`	**Percentage (%)** that represents the harmonic mean of Precision and Recall.
AUC-ROC Score (0-1 scale)	Measures the model's ability to distinguish between fraud and legitimate transactions.	Higher AUC-ROC means better fraud detection performance.	`roc_auc_score(y_true, y_pred)`, plus ROC curve visualization.	**Score between 0 and 1**, where 1 is perfect fraud detection, 0.5 is random guessing.

Confusion Matrix (Count Values)	Shows true positives, false positives, true negatives, and false negatives to analyze misclassification errors.	Understand the type of errors the model makes.	`confusion_matrix(y_true, y_pred)`, visualized with `sklearn.metrics.plot_confusion_matrix()`.	Table showing actual vs. predicted classifications (count values).
Training Time (Seconds)	Measures how long each model takes to train.	Determine computational efficiency of RF vs. GB.	Python's `time()` function: `start_time = time.time()`, `end_time = time.time()`.	**Time in seconds** taken to train RF & GB.
Inference Time (Milliseconds)	Measures how long it takes to classify a new transaction.	Ensure the model is fast enough for real-time fraud detection.	`time()` function will measure batch prediction time.	**Time in milliseconds** per transaction classification.
Memory Usage (MB)	Tracks how much RAM is used during training and inference	Evaluate which model is more memory-efficient.	`psutil.virtual_memory()` (Python).	**Memory consumption** in megabytes (MB) during training & inference.
Disk Utilization (GB)	Ensures adequate storage for datasets, models, and logs.	Prevent storage issues affecting model training.	`df -h` (macOS), `psutil.disk_usage('/')` (Python).	**Total disk space** (GB) used before & after training.

4. **Feature Importance Analysis**

- Random Forest: Extract feature importance using `.feature_importances_`.
- Gradient Boosting: Use SHAP (Shapley Additive Explanations) to understand the model's decision-making.

5. **Computational Efficiency Testing**

- Measure training and inference time for both models.
- Compare memory usage of RF vs. GB.

Estimated Timeline

The research will follow a structured timeline to ensure systematic progress.

Phase	**Activities**	**Duration**
Phase 1: Planning & Preparation	1. Finalizing references & related work review 2. Confirming setup & environment	Week 1
Phase 2: Environment Setup	1. Setting up the development environment. 2. Initial testing of libraries & tools. 3. Measuring disk utilization to ensure adequate memory.	Week 2
Phase 3: Data Acquisition & Preprocessing	1. Download and explore dataset 2. Feature Engineering & Data Balancing 3. Train-test split & Cross-validation setup	Week 3-4
Phase 4: Model Implementation	1. Implement Random Forest Model. 2. Implement Gradient Boosting Model (XGBoost & LightGBM). 3. Perform initial testing with default parameters.	Week 5-6
Phase 6: Feature Importance & Interpretability	1. Extract Feature Importance from RF & GB. 2. Compare which features contribute most to fraud classification using SHAP.	Week 9

Phase 7: Computational Efficiency Testing	1. Compare Training & Inference Time of RF & GB. 2. Measure Memory & Disk Utilization during model execution. 3. Assess real-time applicability for fraud detection.	Week 10
Phase 8: Comparative Analysis & Discussion	1. Summarize trade-offs between RF & GB. 2. Discuss results, including computational costs vs. accuracy benefits.	Weeks 11 - 12
Phase 9: Report Writing & Final Submission	1. Compile all results, format in IEEE style. 2. Perform final edits, review, and submit the report.	Weeks 13 - 14

Feasibility

This research is feasible within the allocated time frame, but some challenges and risks must be considered.

Challenges & Risks

1. **Imbalanced Data Handling:**
    - Fraud cases are very rare (~0.17%), which can lead to biased model predictions.
    - Solution: Use SMOTE oversampling and compare results with baseline models.
2. **Computational Resources:**
    - Gradient Boosting requires more computation compared to Random Forest.
    - Solution: Use optimized versions (LightGBM instead of XGBoost) to reduce computational cost.

3. **Hyperparameter Optimization Complexity:**
    - Tuning parameters for XGBoost/LightGBM can be time-consuming.
    - Solution: Use GridSearchCV or RandomizedSearchCV to efficiently find the best parameters.
4. **Interpretability of Gradient Boosting Models:**
    - Tree-based models lack direct interpretability.
    - Solution: Use SHAP to visualize feature contributions.

Hypothesis / Expected Outcomes

The primary hypothesis is that Gradient Boosting will outperform Random Forest in terms of accuracy and handling imbalanced datasets due to its sequential approach, which helps in capturing complex patterns in fraud detection. However, the secondary hypothesis predicts that Random Forest will be faster and more computationally efficient, making it more suitable for real-time applications.

Expected Outcomes

- Gradient Boosting is expected to show higher recall and precision due to its iterative nature, but with a higher training time.
- Random Forest should be more computationally efficient with similar performance in terms of accuracy but might struggle with the complexity of the imbalanced dataset.

Metric	**Expected Outcome**
Precision & Recall	GB expected to perform better
F1-score	GB expected to be higher
AUC-ROC	GB expected to be superior
Training Time	RF expected to be faster
Inference Time	RF expected to classify transactions faster
Memory Usage	RF expected to require less
Disk Utilization	GB expected to use more storage
Confusion Matrix Analysis	GB expected to have fewer false negatives

If results differ from expectations, additional hyperparameter tuning and ensemble hybrid models may be considered.

References

[1] A. Sarker, S. Rahman, and T. Ahmed, “Credit card fraud detection using machine learning techniques,” *J. Comput. Commun.*, vol. 12, no. 6, pp. 1–11, Jun. 2024. [Online]. Available: https://doi.org/10.4236/jcc.2024.126001

[2] G. Bontempi, “Calibrating probability with undersampling for unbalanced classification,” in *Proc. IEEE Symp. Comput. Intell.*, Jan. 2015. [Online]. Available: https://www.academia.edu/98154430/Calibrating_Probability_with_Undersampling_for_Unbalanced_Classification

[3] F. Carcillo, Y. Le Borgne, O. Caelen, and G. Bontempi, “Streaming active learning strategies for real-life credit card fraud detection: Assessment and visualization,” *Int. J. Data Sci. Anal.*, vol. 5, no. 4, pp. 285–300, 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1804.07481

[4] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2016, pp. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[5] M. Zięba, A. Tomczak, and M. Tomczak, “Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction,” *Expert Syst. Appl.*, vol. 58, pp. 93–101, Oct. 2016. [Online]. Available: https://www.ii.pwr.edu.pl/~tomczak/PDF/[MZSTJT].pdf