Fraud Detection Using Machine Learning Documentation Report

Introduction:

Fraudulent activity poses a significant threat to the financial landscape, requiring sophisticated tools for detection and prevention. This report details the development and evaluation of a machine-learning model for fraud detection, focusing on addressing class imbalance and optimizing performance for real-world applications.

We explore feature engineering, model selection, training, evaluation, and performance improvement strategies, culminating in a robust and effective system for identifying fraudulent transactions. Dive into the details to understand the model's capabilities and potential in safeguarding against financial crime.

Data Collection:

The dataset used for this project was sourced from Kaggle and can be accessed through the following link We are using this dataset from Kaggle: https://www.kaggle.com/datasets/dermisfit/fraud-transactions-dataset?select=fraudTrain.csv

This dataset is divided into train and test, with the same 23 columns.

train: 1.30m

test: 556k.

The dataset contains a large number of fraudulent data.

Data Exploration:

- Identified a significant class imbalance (0.5% fraud cases).
- Found outliers in transaction amounts and user locations.
- Visualized patterns using scatter plots and histograms.
- Have not found any missing values in the dataset.
- Number of columns: 22
- Relevant columns: Age, trans_date_trans_time, cc_num, merchant, category, amt, is_fraud,

Preprocessing

Outlier Removal:

- Removal of transactions with amounts above 3000, as they were not identified as fraudulent and could potentially distort the analysis.
- Removal of transactions with longitudes less than -130 and amounts between 1000 and 2000, as they exhibited suspicious patterns involving states AK and HI.

Categorical Feature Encoding:

- Label encoding of categorical features (merchant, category, street, city, state, job) using OrdinalEncoder with -1 as the unknown value

Feature Engineering:

Time-Based Feature Extraction:

- Extracted trans_hour and trans_day_of_week from trans_date_trans_time to capture temporal patterns.

Age Calculation:

- Converted dob to DateTime format.
- Calculated age as the difference between the current date and the dob in years.

High-Risk Indicator Features:

- Created new columns
- high_risk_merchant (1 for high-risk merchants, 0 otherwise)
- high_risk_city (1 for high-risk cities, 0 otherwise)
- high_risk_state (1 for high-risk states, 0 otherwise)
- high_risk_job (1 for jobs associated with higher fraud risk, 0 otherwise)

Pipeline for Consistency:

- Employed a pipeline from scikit-learn to streamline feature processing and encoding, ensuring consistent application to both training and test datasets.

Model Selection:

- Rationale: Logistic Regression was chosen due to its suitability for binary classification tasks, aligning with the goal of predicting fraudulent transactions (0 or 1). It offers interpretability, allowing for understanding the relationship between features and the target variable

Model Training:

- Data Splitting: The dataset was already divided into separate training and test sets for model development and evaluation.
- Pipeline for Preprocessing and Modeling: A pipeline was constructed using scikit-learn's Pipeline functionality to streamline the process:
- ColumnTransformer:
- StandardScaler: Applied to numerical columns (list the specific columns).
- OneHotEncoder: Applied to categorical columns (e.g., 'gender').
- Logistic regression: Employed as the chosen classification model.

Training Process

- The pipeline was fitted to the training dataset, encompassing both preprocessing and model training.
- Model performance was evaluated on both training and test sets to assess overfitting and generalization.

Model Evaluation:

Model Evaluation Metrics:

- Metrics Employed:
- Accuracy
- Precision
- Recall
- F1-score
- AUC-ROC
- Confusion matrix

Evaluation of Training Set:

- Accuracy: 0.9937
- Precision: 0.323
- Recall: 0.0773

- F1 Score: 0.125
- AUC-ROC: 0.5382

Threshold Adjustment:

- A function evaluate_at_threshold was implemented to explore model performance at different probability thresholds.
- This enables customization of the trade-off between precision and recall, aligning with specific project requirements

Performance Improvement:

Initial Model Performance:

- The initial model exhibited poor performance on the training set, with low recall (0.0773) and F1-score (0.125), indicating a failure to capture fraudulent transactions effectively.
- This underfitting was attributed to significant class imbalance.

Addressing Class Imbalance:

- Class Weighting: Initial attempts using class weighting and regularization did not yield substantial improvements.
- Resampling Techniques: The following resampling techniques were explored:
- Random Under-sampling: Reduced the size of the majority class to match the minority class.
- Random Over-sampling: Replicated samples from the minority class to increase its size.
- SMOTE (Synthetic Minority Over-sampling Technique): Generated synthetic minority samples to augment the dataset.
- SMOTE-ENN (SMOTE and Edited Nearest Neighbors): Combined SMOTE with undersampling to refine synthetic samples and address potential noise.

Performance Results:

- Random Under-sampling:
- Accuracy: 0.8516
- F1-score: 0.8365
- Random Over-sampling:
- Accuracy: 0.8528

- F1-score: 0.8376
- SMOTE:
- Accuracy: 0.8558
- F1-score: 0.8415
- SMOTE-ENN:
- Accuracy: 0.9511
- F1-score: 0.9514

Best Performing Approach:

SMOTE-ENN emerged as the most effective technique, significantly improving model performance across various metrics.

Key Considerations:

Threshold Adjustment: Explore different probability thresholds to fine-tune the trade-off between precision and recall, aligning with specific project requirements.

Cross-Validation: Incorporate cross-validation to obtain more reliable performance estimates and reduce overfitting potential.

Visualization: Utilize ROC curves and precision-recall curves to further assess model performance and guide threshold selection.

Business Context: Carefully consider the costs of false positives and false negatives in the context of the fraud detection problem to determine acceptable performance levels.

Just to take an overview we are going to run all estimators:

Model Overview

Here's the table formatted for easy pasting into Google Docs:

Index	Model Name	Model Precision
0	AdaBoostClassifier	0.9886201991465149
1	BaggingClassifier	0.9863453815261044
2	BernoulliNB	0.9808173477898249

3	CalibratedClassifierCV	0.9847144006436042
4	DecisionTreeClassifier	0.9672544080604534
5	DummyClassifier	0.524006908462867
6	ExtraTreeClassifier	0.9508320726172466
7	ExtraTreesClassifier	0.9971617786187322
8	GaussianNB	0.9644478063540091
9	GaussianProcessClassifier	0.9833333333333333
10	GradientBoostingClassifier	0.99231843575419
11	HistGradientBoostingClassifier	0.994269340974212
12	KNeighborsClassifier	0.9459459459459459
13	LabelPropagation	0.9337231968810916
14	LabelSpreading	0.9337231968810916
15	LinearDiscriminantAnalysis	0.997338065661047
16	LinearSVC	0.9847512038523274
17	LogisticRegression	0.9886822958771221
18	LogisticRegressionCV	0.9972401103955841
19	MLPClassifier	0.9797297297297297
20	NearestCentroid	0.9989384288747346
21	NuSVC	0.9918283963227783
22	PassiveAggressiveClassifier	0.9446280991735537
23	Perceptron	0.9809358752166378
24	QuadraticDiscriminantAnalysis	0.6181818181818182

25	RandomForestClassifier	0.9951377633711507
26	RidgeClassifier	0.9973333333333333
27	RidgeClassifierCV	0.997338065661047
28	SGDClassifier	0.9822580645161291
29	SVC	0.988831615120275

Addressing challenges:

In the Data Collection section, mention the size of the dataset (1.86M total, with 1.3M training and 556k test) for context.

In the Data Exploration section, highlight the most significant outliers removed and their potential impact on the analysis.

In the Model Training section, briefly mention the encountered Ram crash problem and the solution of shifting data to Google Colab. This adds realism and showcases problem-solving skills.

Strengthening the conclusion:

- Briefly summarize the best-performing model and its key advantages.
- Emphasize the practical implications of the project, e.g., potential fraud loss reduction or user protection enhancements.
- Briefly touch on future work beyond what's already listed, like potential deployment plans or integration with existing systems.

Minor adjustments:

- In the Feature Engineering section, consider merging the "High-Risk Indicator Features" and "Pipeline for Consistency" subsections for smoother flow.
- In the Model Overview table, you might want to consider listing the top 5-10 models instead of all 29 for conciseness.

Conclusion:

This document presented a comprehensive overview of the model development process for fraud detection, focusing on feature engineering, model selection, training, evaluation, and performance improvement.

Key Findings:

- The initial model trained on imbalanced data exhibited poor performance, highlighting the need for addressing class imbalance.
- Class balancing techniques, particularly SMOTE-ENN, significantly improved model accuracy, and F1-score, demonstrating the importance of handling data imbalances before model training.
- Evaluation using various metrics provided a deeper understanding of the model's strengths and weaknesses, allowing for informed choices on threshold selection and performance interpretation.
- Exploration of other classification algorithms revealed several models with comparable or even better performance than Logistic Regression, suggesting further investigation and potential model ensembling for future iterations.

Future Work:

- Implement cross-validation to obtain more robust performance estimates and reduce overfitting potential.
- Explore hyperparameter tuning to further optimize the chosen model.
- Employ additional class imbalance techniques and compare their effectiveness.
- Analyze feature importance to identify the most impactful features for fraud prediction.
- Consider model ensembling techniques to combine the strengths of different algorithms.
- Integrate domain knowledge to refine feature engineering and model interpretation.

This initial effort serves as a valuable foundation for building a robust and effective fraud detection system. Continuous improvement through future work will enhance the model's accuracy and real-world applicability, helping to mitigate financial losses and protect users from fraudulent transactions.