Detecting Credit Fraud with 6 Different Models In Machine Learning

Wesley Kieu
Department of Data Science
San Jose State University
San Jose, USA
wesley.kieu@sjsu.edu

***Abstract*—In credit card transaction, it is hard to determine which is fraudulent while which aren't most of the time in the real world, fraudulent transactions are on the smaller scale which in order to collect data, there would be an imbalance. This machine learning research focus on several different supervised learning algorithms to see which model performs better in an imbalance dataset for credit fraud. Model performance was measured by recall, precision, F1-Score, and ROC-AUC, focusing on the fraud class metrics. Overall, the results of this research shows XGBoost provides the most effective balance and detection rate on catching fraudulent transactions. With tuning different models, XGBoost performs well and still needs to be improved in order to catch those fraud transactions.**

I. Introduction

The objective of this project is to detect credit fraud in an imbalance dataset with 6 different models. Each model will have its baseline and then improved through hyperparameter tuning, class balancing techniques, and performance optimization. The models I selected were Logisitic Regression, Random Forest, XGBoost, Deep Learning Neural Network, Light Gradient Boosting Machine (LGBM), and K-Nearest Neighbor (KNN). To evaluate this models, accuracy won't matter as much but Precision, Recall, F1, and ROC-AUC, and AUPRC will be the important primary metrics.

II. Dataset

A. Understanding the Dataset

The dataset used in this project was created during a research collaboration of Worldline and the Machine Learning Group of ULB (Université Libre de Bruxelles). This dataset contains credit card transactions in September 2013 by European cardholders. The transaction timeframe span between two days with 492 fraud out of 284,807. There are 31 columns where features V1-V28 are the principal components obtained with PCA due to confidentiality. Time and Amount are not transformed, and Class takes values of 1 for fraud and 0 for real.

III. Data Preprocessing

Features *V1–V28* are already transformed using Principal Component Analysis (PCA), only the *Time* and *Amount* features require additional preprocessing. To make sure all features are on a comparable scale, the *Time* and *Amount* columns were standardized using the *StandardScaler* method. After, I split the dataset to features and targets (X, y), before dividing into training, validation, and test sets.

A. Shuffle and Splitting the Dataset

The dataset was divided into training, validation, and test sets using a 70–15–15 split. Stratified sampling was applied and the dataset was shuffled before splitting which is important because the dataset was ordered in time. This should prevent some sort of bias as the start and should contain a good mix between both classes.

IV. Model Architecture

A. Logistic Regression

1) Baseline Model: Logistic Regression was selected as the first baseline model due to its simplicity, efficiency, and interpretability. If the baseline model performs well, more complex models may not be necessary. The model was trained using the `class_weight='balanced'` parameter to address class imbalance, a maximum of 1000 iterations to ensure convergence, and a random state of 42 for reproducibility.

TABLE I
Baseline Logistic Regression Model

Parameter	Value
class_weight	balanced
max_iter	1000
random_state	42

TABLE II
Baseline Logistic Regression Evaluation Metrics

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	0.98	0.99	42648
1 (Fraud)	0.06	0.89	0.12	74
Accuracy			0.98	42722
Macro Avg	0.53	0.93	0.55	42722
Weighted Avg	1.00	0.98	0.99	42722

2) Evaluation: Looking at the fraud row, the model performed poorly with a precision of 0.06 and a recall of 0.89, resulting in a low F1-score of 0.12. This indicates that although the model was able to identify most fraudulent transactions, it generated a large number of false positives.

3) Hyperparameter Tuning: With hyperparameters, since there is an imbalance in fraud to legitimate cases, I am going to use Synthetic Minority Over-sampling Technique (SMOTE) where it will create synthetic minority class which is Fraud. This will balance the dataset better since it will oversample the fraud class. After, I will use a GridSearchCV from sikit-learn to test different hyperparameters. Here is a list below:

TABLE III
LOGISTIC REGRESSION HYPERPARAMETER GRID FOR TUNING

Hyperparameter	Values Tested
penalty	{l1, l2}
C	{0.001, 0.01, 0.1, 1, 10}
class_weight	balanced
max_iter	1000

TABLE IV
LOGISTIC REGRESSION EVALUATION WITH SMOTE

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	0.98	0.99	42648
1 (Fraud)	0.06	0.88	0.11	74
Accuracy			0.98	42722
Macro Avg	0.53	0.93	0.55	42722
Weighted Avg	1.00	0.98	0.99	42722

TABLE V
LOGISTIC REGRESSION EVALUATION WITH TUNING WITHOUT SMOTE

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	0.98	0.99	42648
1 (Fraud)	0.07	0.89	0.13	74
Accuracy			0.98	42722
Macro Avg	0.53	0.94	0.56	42722
Weighted Avg	1.00	0.98	0.99	42722

4) Results: With the evaluations, both hyperparamter testing did not improve. It only improve 1% over or under. This shows a big imbalance still on classifying fraud transactions.

B. Random Forest

1) Baseline Model: With random forest, my baseline parameters are:

TABLE VI
RANDOM FOREST BASELINE MODEL

Parameter	Value
n_estimators	100
random_state	42
class_weight	balanced
n_jobs	-1

2) Evaulation: With the baseline model, the fraud transaction increase in precision and recall (0.93 and 0.72) with an F1-score of 0.81. This improves from the past Logistic Regression Model.

3) Hyperparameter Tuning: With testing different hyperparameters, I wanted to use GridSearchCV to test different parameters like *n_estimators* and *max_depth*. Also, I kept the class balance by keeping the *class_weight='balanced'*

TABLE VII
BASELINE RANDOM FOREST EVALUATION

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	1.00	1.00	42648
1 (Fraud)	0.93	0.72	0.81	74
Accuracy			1.00	42722
Macro Avg	0.96	0.86	0.90	42722
Weighted Avg	1.00	1.00	1.00	42722

TABLE VIII
RANDOM FOREST HYPERPARAMETER

Parameter	Search Range
n_estimators	[100, 200, 300]
max_depth	[10, 20, 30, None]

4) Results: The model didn't improve nor it decrease. It stayed the same as GridSearch CV came back with *max_depth* as 20 and *n_estimators* as 100 which is the baseline. The ROC-AUC of the baseline model was a 93% which means a randomly chosen fraud case gets a higher predicted score than a randomly chosen legitimate case.

C. XGBoost

1) Baseline Model: With XGBoost model, in the table, these are the baseline foundation parameters.

TABLE IX
XGBOOST BASELINE MODEL CONFIGURATION

Parameter	Value
n_estimators	300
max_depth	5
learning_rate	0.05
subsample	0.8
colsample_bytree	0.8
scale_pos_weight	*neg/pos ratio*
random_state	42
n_jobs	-1
eval_metric	logloss
tree_method	hist

TABLE X
BASELINE XGBOOST EVALUATION

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	1.00	1.00	42648
1 (Fraud)	0.80	0.81	0.81	74
Accuracy			1.00	42722
Macro Avg	0.90	0.91	0.90	42722
Weighted Avg	1.00	1.00	1.00	42722

2) Evaluation: We can see that XGBoost performed better and more accurately in detecting fraud. The Precision and Recall are of by a percent which indicates a good spotting for the model. The ROC AUC was 97% which shows a good indicator of data imbalance and recongizing the fraud cases.

3) Hyperparameter Tuning: The hyperparameters I want to test are *n_estimators*, *max_depth*, and *learning rate* with 3 cv folds. The table of the values are listed in Table XI.

TABLE XI
XGBOOST HYPERPARAMETER TUNING

Hyperparameter	Values Tested
n_estimators	{200, 300}
max_depth	{4, 5}
learning_rate	{0.05, 0.1}
cv folds	3

TABLE XII
XGBOOST TUNING EVALUATION

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	1.00	1.00	42648
1 (Fraud)	0.88	0.80	0.84	74
Accuracy			1.00	42722
Macro Avg	0.94	0.90	0.92	42722
Weighted Avg	1.00	1.00	1.00	42722

4) Results: The model increase in precision and recall decreased by 1% which is ultimately still good. The F1-Score is at a 84% which is a 3% increase from our baseline model of XGBoost. The ROC AUC has a 97% which is similar to the last model of Random Forest. over. The best parameters that were listed were a learning rate of 0.1, *max_depth* of 5 and *n_estimators* of 300. This shows good progression to a better model since we improve by tuning it with those hyperparameter.

D. LightGBM

1) Baseline Model: With the baseline model, this was a fairly new model I learned and took a foundation parameter to start with. The reason why I went with LGBM is it's speed since Random Forest and XGBoost took quite some time. There was a paramter scale_pos_weight which helps with imbalance data like XGBoost, so to test between XGBoost and LightGBM, I want to see the different outcomes.

TABLE XIII
LIGHTGBM BASELINE MODEL ARCHITECTURE

Parameter	Value
n_estimators	300
learning_rate	0.05
max_depth	-1
num_leaves	31
subsample	0.8
colsample_bytree	0.8
scale_pos_weight	*neg/pos ratio*
random_state	42
n_jobs	-1

TABLE XIV
LIGHTGBM EVALUATION RESULTS

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	0.99	1.00	42648
1 (Fraud)	0.17	0.81	0.29	74
Accuracy			0.99	42722
Macro Avg	0.59	0.90	0.64	42722
Weighted Avg	1.00	0.99	1.00	42722

2) Evaluation: Looking at this baseline model, it perform poorly compared to Random Forest and XGBoost. It was quick in terms of training but still perform badly. But it increase better than the Logistic Regression model which can be tune to some level.

3) Hyperparameter Tuning: With the hyperparameter, I chose a smaller grid search due to how efficient the other models were. But to test different hyperparameter, I decided to test relativity the same as XGBoost to see any outcome while having the class_weight balanced and across 3 cv folds.

TABLE XV
LIGHTGBM HYPERPARAMETER FOR TUNING

Hyperparameter	Values Tested
n_estimators	{200, 300}
max_depth	{-1, 5}
learning_rate	{0.05, 0.1}
cv folds	3

TABLE XVI
LIGHTGBM TUNING EVALUATION RESULTS

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	0.98	0.99	42648
1 (Fraud)	0.08	0.81	0.15	74
Accuracy			0.98	42722
Macro Avg	0.54	0.90	0.57	42722
Weighted Avg	1.00	0.98	0.99	42722

4) Results: With LGBM, the model did not improve which actually decreased on the paramters when tuning. The learning rate as it increase actually decrease in precision dramatically which results in the F1-Score. From the baseline model, I can conclude that XGBoost still leads in detecting the fraud cases.

E. Neural Network

1) Baseline Architecture: With the neural network, there was 3 hidden layers (32, 64, 128) with all ReLU as the activation function. The optimizer was Adam with a learning rate of 0.001. I did a batch size of 2048 since the data was imbalance and trained about 50 epochs. Although accuracy was measure mainly, I did ensure recall and precision as metrics within the model.

TABLE XVII
NEURAL NETWORK MODEL ARCHITECTURE

Component	Value
Input Dimension	*number of features*
Hidden Layer 1	Dense(128), ReLU
Hidden Layer 2	Dense(64), ReLU
Hidden Layer 3	Dense(32), ReLU
Output Layer	Dense(1), Sigmoid
Optimizer	Adam (learning rate = 0.001)
Loss Function	Binary Cross-Entropy
Batch Size	2048
Epochs	50 (with EarlyStopping)

2) Evaluation: The model perform poorly ranking close to Logistic and LGBM. Neural network suffers from imbalance data, so in order to increase this, the use of SMOTE is needed

TABLE XVIII
DEEP NEURAL NETWORK EVALUATION RESULTS

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	0.99	1.00	42648
1 (Fraud)	0.16	0.86	0.26	74
Accuracy			0.99	42722
Macro Avg	0.58	0.93	0.63	42722
Weighted Avg	1.00	0.99	0.99	42722

to improve and balanced out the dataset. There are too many false positives which is something we don't want. But the recall is 0.86 which great but F1-score is low.

3) Hyperparameter Tuning: To tune, I want to focus on threshold, SMOTE, and Dropouts. If the Thresholds is higher, my hypothesis may show an increase in precision which could be closer to recall. SMOTE allows to oversample so the features and targets can be balanced between 3 sets (Train, Validation, Test). Since the ratio is off between the classes, dropouts can be used to reduce overfitting on the legitmate classes and classify the fraud transactions.

TABLE XIX
DNN TUNING PARAMETERS

Parameter	Values Tested
Threshold	{0.50, 0.60, 0.70, 0.80}
SMOTE	{Applied, Not Applied}
Dropout Rate	{0.2, 0.3, 0.5}

TABLE XX
DNN EVALUATION (DROPOUT = 0.0)

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	1.00	1.00	42648
1 (Fraud)	0.83	0.77	0.80	74
Accuracy			1.00	42722
Macro Avg	0.91	0.88	0.90	42722
Weighted Avg	1.00	1.00	1.00	42722

TABLE XXI
DNN EVALUATION (DROPOUT = 0.2)

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	1.00	1.00	42648
1 (Fraud)	0.80	0.81	0.81	74
Accuracy			1.00	42722
Macro Avg	0.90	0.91	0.90	42722
Weighted Avg	1.00	1.00	1.00	42722

4) Results: Looking at all results, the dropout wiht 0.2 achieved the best fraud F1-score of 0.81 with an ROC-AUC at 0.97. Testing higher dropout values made the model less strong when it comes to recall. With this dropout, the threshold was 0.6 which benefited from the baseline model as it was 0.5. This was one of the most improvements in tuning and testing different hyperparamters when it comes to an imbalance of data. Although it may not be the best model, a 0.81 F1 score with a 0.1 difference in precision and recall is a strong result.

TABLE XXII
DNN EVALUATION (DROPOUT = 0.3)

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	1.00	1.00	42648
1 (Fraud)	0.78	0.80	0.79	74
Accuracy			1.00	42722
Macro Avg	0.89	0.90	0.89	42722
Weighted Avg	1.00	1.00	1.00	42722

TABLE XXIII
DNN EVALUATION (DROPOUT = 0.5)

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	1.00	1.00	42648
1 (Fraud)	0.78	0.78	0.78	74
Accuracy			1.00	42722
Macro Avg	0.89	0.89	0.89	42722
Weighted Avg	1.00	1.00	1.00	42722

F. K-Nearest Neighbors (KNN)

1) Baseline Model: With KNN it does not learn explicit decision boundaries so it finds the nearest neighbors in the feature space when classifying. It's an overly simple model but reading more about KNN, it is extremely sensitive to imbalance data. Since fraud classes are low, I want to add SMOTE before fitting KNN. This allows the model to generate synthetic fraud samples and balancing the training data so it does not overfit. So my baseline model for KNN includes SMOTE + KNN.

TABLE XXIV
K-NEAREST NEIGHBORS (KNN) MODEL CONFIGURATION

Parameter	Value
n_neighbors	3
weights	distance
n_jobs	-1
Distance Metric	Euclidean (default)

TABLE XXV
KNN EVALUATION (SMOTE + KNN PIPELINE)

Class	Precision	Recall	F1-Score	Support
0 (Legitimate)	1.00	1.00	1.00	42648
1 (Fraud)	0.60	0.85	0.70	74
Accuracy			1.00	42722
Macro Avg	0.80	0.93	0.85	42722
Weighted Avg	1.00	1.00	1.00	42722

2) Evaluation: From the results of the baseline model, the model did improve from other models but isn't the best still but a good baseline. 0.60 Precision and 0.85 Recall with a F1-Score of 0.70 beats Logistic Regression and LGBM. From these results,

3) 2) Parameter Testing (k = 3, 5, 7): Testing these parameters controls the number of nearest neighbors. Like finding the closest 3, 5, and 7 neighbors. The baseline started with 3 because I wanted to reduce the noise and reduce overfitting of capturing other data. Now to test on different k parameters, an issue may capture noise and be too smooth where the majority class dominates.

TABLE XXVI
KNN FRAUD-CLASS PERFORMANCE K=5

Class	Precision	Recall	F1-Score	Support
1 (Fraud)	0.48	0.85	0.62	74

TABLE XXVII
KNN FRAUD-CLASS PERFORMANCE K=7

Class	Precision	Recall	F1-Score	Support
1 (Fraud)	0.41	0.88	0.56	74

4) Results: Both k values decrease in F1-score which shows that the model is capturing the noise of the dataset. With the baseline model of an F1-Score of 0.70, shows a good learning on capturing the nearest neighbors of the class. Tuning k lies with seeing how noise affects precision and recall. Taking the oversampling help the model balanced the dataset before putting it into KNN which performed better than expected. Although it is a simple model, it outperforms Logistic and LGBM.

V. CONCLUSION

Overall, the objective of this research was to test 6 different models (Logistic Regression, Random Forest, XGBoost, LGBM, Deep Neural Network, and KNN) on an imbalance dataset on detecting fraud transactions. The imablance dataset allow different hyperparamter tuning in order to capture the fraud transaction. Many models had different precision and recall trade offs but the key metrics use to evaluate were: precision, recall, F1-score, and ROC-AUC.

The winning model lies within XGBoost when tuning from its baseline model. Achieving a high F1-score of 0.84 with a precision of 0.88 and a recall of 0.80. The hyperparamter that improved this model were a learning rate of 0.1, max depth of 5 and n-estimator of 300. It was a 3% increase from the baseline model.

Looking at Fig. 2 and Fig. 3, the ROC and Precision–Recall curves show the clear dominance of the XGBoost model over all other classifiers, making it the most favorable model based on our evaluation metrics. While most models maintain precision and recall within a reasonable range, Fig. 1 highlights that Logistic Regression and LightGBM perform worse compared to the others.

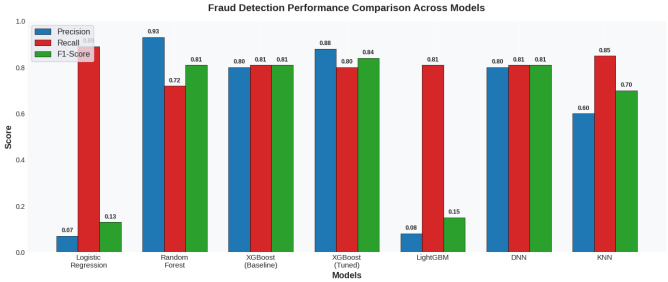

Fig. 1. Model Comparison Evaluations

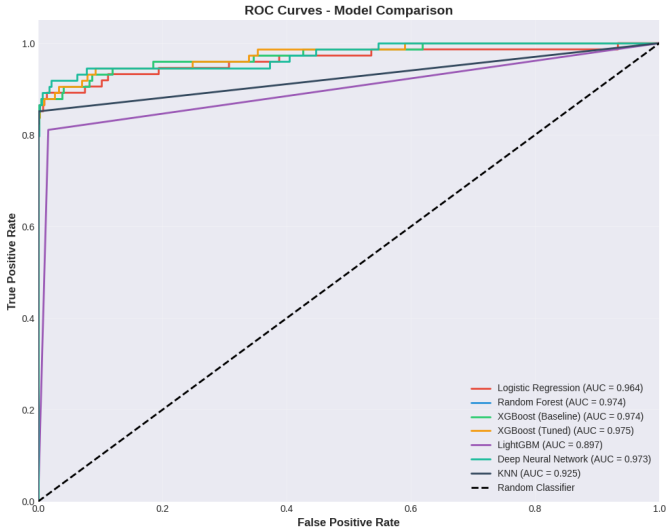

Fig. 2. ROC Curves - Model Comparison

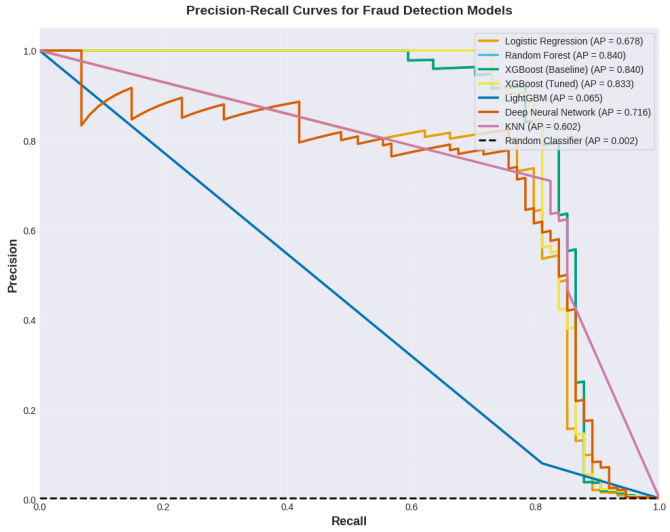

Fig. 3. Precision-Recall Curve

The second best model is the Deep Neural Network where Precision and Recall are only 0.01 off with an F1-score of 0.81. This was tuned after the baseline model was done poorly. In a neural network, overfitting is sensitive when there is an imbalance. With the baseline model, it proves just that. The tuned model allowed dropouts to reduce overfitting, and alongside other parameters and using a different threshold, the model improved.

Imbalance dataset are everywhere and in this research project, SMOTE helped models across different times. Although, XGBoost is the best model so far. It is not the best because 16% of the fraud data was not found which is still critical to any system in credit transaction in the real world setting. These models set a baseline into improving and testing more hyperparameters to increase precision and recall.

REFERENCES

[1] Machine Learning Group, ULB (Université Libre de Bruxelles). *Credit Card Fraud Detection Dataset*. Available at: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data

[2] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi. *Calibrating Probability with Undersampling for Unbalanced Classification*. IEEE Symposium on Computational Intelligence and Data Mining (CIDM), 2015.

[3] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi. *Learned Lessons in Credit Card Fraud Detection from a Practitioner Perspective*. Expert Systems with Applications, 41(10), 4915–4928, 2014.

[4] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi. *Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy*. IEEE Transactions on Neural Networks and Learning Systems, 29(8), 3784–3797, 2018.

[5] A. Dal Pozzolo. *Adaptive Machine Learning for Credit Card Fraud Detection*. PhD Thesis, Machine Learning Group (MLG), ULB, supervised by Gianluca Bontempi.

[6] F. Carcillo, A. Dal Pozzolo, Y.-A. Le Borgne, O. Caelen, Y. Mazzer, and G. Bontempi. *SCARFF: A Scalable Framework for Streaming Credit Card Fraud Detection with Spark*. Information Fusion, 41, 182–194, 2018.

[7] F. Carcillo, Y.-A. Le Borgne, O. Caelen, and G. Bontempi. *Streaming Active Learning Strategies for Real-Life Credit Card Fraud Detection: Assessment and Visualization*. International Journal of Data Science and Analytics, 5(4), 285–300, 2018.

[8] B. Lebichot, Y.-A. Le Borgne, L. He, F. Oblé, and G. Bontempi. *Deep-Learning Domain Adaptation Techniques for Credit Card Fraud Detection*. INNSBDDL 2019: Recent Advances in Big Data and Deep Learning, pp. 78–88, 2019.

[9] F. Carcillo, Y.-A. Le Borgne, O. Caelen, F. Oblé, and G. Bontempi. *Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection*. Information Sciences, 2019.

[10] Y.-A. Le Borgne and G. Bontempi. *Reproducible Machine Learning for Credit Card Fraud Detection — Practical Handbook*. Available online.

[11] B. Lebichot, G. Paldino, W. Siblini, L. He, F. Oblé, and G. Bontempi. *Incremental Learning Strategies for Credit Card Fraud Detection*. International Journal of Data Science and Analytics.