Yan Zeng *Contact: dirkzeng@gmail.com* *Cell: (402)570-7945*

Chime Take-home Exercise

Yan Zeng *Contact: dirkzeng@gmail.com* *Cell: (402)570-7945*

Part I: SQL Exercise

Exercise One:

Assumptions:
1) There may not be a deposit transaction every day
2) Assume we are creating a daily view that covers each calendar day of the past 30 days, even those days when there were no transaction activities
3) To ensure all calendar days are in the summary table, we need a system ***calendar_date*** table (usually it's a database system table). If there is none, it's also an easy procedure to create such a table and write it into the analytics DB

Exercise Two:

Assumptions:
The final summary contains all *user_id*; for those without *cash_deposit*, the *id*, *timestamp*, and *amount* fields are NULL

The SQL coding logic can be found here

Yan Zeng *Contact: dirkzeng@gmail.com* *Cell: (402)570-7945*

Part II: Decision Tree Models for Credit Card Fraud Detection

Note: Refer here for the coding part

Summary

I chose a very imbalanced credit card fraud dataset for this exercise that resembles production data in the money movement space. Accordingly, I applied three different decision tree techniques:

- A baseline model
- Upsampling the training data
- Applying weights to the "minority class."

To select the final model, I analyzed precision, recall, and the area under the precision-recall curve (AUPRC) as primary metrics to address the imbalance issue. **The weighted decision tree model has the best performance out of three because it could optimize the fraud detection efficacy ("recall") and reduce friction for good users (best precision).**
I also identified top feature candidates for potential risk rules and policies to mimic the production use case.

Data Introduction

The dataset for this exercise comes from a past Kaggle data challenge. It contains credit card transactions made in September 2013 by European cardholders. I chose this dataset because there was a 0.172% fraud rate of more than 284K transactions. The imbalanced nature of the problem resembles the "bad" rate in the money movement space (e.g., fraud/non-fraud related ACH return rate).

Decision Tree Modeling Process

Note: please refer to the corresponding sections in the notebook for code/output details

Data Preparation and Exploration

- **Data storage and environment**
  To mimic the production environment, I used **AWS Sagemaker and S3** to store the dataset and managed the subsequent analytics and model training part.
- **Data exploration**
  I performed the following exploratory data analysis (EDA) and data sanity check:
  - **Check the missing data** and null attributes. Since it's a clean dataset, there is no need to handle the missing data treatment and encoding.
  - **Understand the feature distribution** and identify anomalies.

Yan Zeng *Contact: dirkzeng@gmail.com* *Cell: (402)570-7945*

Training/Testing Data Split

For this exercise, I chose the random split. In an actual problem setting, when there are other identifier categories such as product type, acquisition channel, device type, and account age, I would also explore stratified sampling.

Decision Tree Modeling Techniques

- **Modeling approach**
  - **Baseline model:** Built to establish the performance benchmark. There are no sampling techniques, weights, or minority class imputation involved.
  - **Upsampling the "fraud" class in training data**: In the training dataset, I added more "fraud" transaction copies by oversampling the original fraud transactions with replacement.
  - **Applying weights to the "fraud" class**: In this analysis, the assumption is missing detecting a fraudulent transaction ("false negative") would be more costly than miss treating a "good" transaction ("false positive"). As a result, I applied additional weights to the "fraud" class when tuning the hyperparameters for the model.
    Note well, it's important to balance the customer experience with friction introduced to reduce fraud loss. In production, the weight application is usually heuristic and more conservative.

- **The model hyperparameter tuning**
  Overall, I applied grid search and cross-validation to identify the optimal hyperparameters (especially the following parameters):
  - Information criterion: between "gini" and "entropy"
  - Tree depth: iteration between 3 - 8
  - Class weight: between balanced and the weighted decision tree model

Decision Tree Modeling Evaluation

- **Precision, Recall, and AUPRC as primary metrics**
  After applying the trained models to the testing dataset, I evaluated the precision, recall, AUPRC, and accuracy score (AUROC). Due to the imbalanced nature of the data, precision, recall, and AUPRC are used as primary evaluation metrics. Besides, they are more intuitive from a business standpoint.
  When evaluating the AUPRC, it's also important to reference the overall fraud rate as a baseline to understand the incremental lift of the model, as AUPRC is a relative metric.
  The **accuracy** score (AUROC) is used to understand the performance difference among all three models on a relative basis. I limited the use of AUROC because the over-representation of "healthy" transactions could make the accuracy score misleading.

Table 1 summarizes the performance statistics of all three models. **Overall, the weighted decision tree model has the best precision performance with a comparable recall relative to the other two models.**
From a risk/product standpoint, using the weighted decision tree model could optimize the fraud detection efficacy and the user experience.

Table 1. Precision-Recall, AUPRC Comparison

	Precision	Recall	AUPRC	Accuracy Score (AUROC)
Baseline Model	0.08	0.80	0.4835726068362793	0.984763850319725
"Upsampled" Decision Tree Model	0.10	0.81	0.49507423573172354	0.9880089799653143
Weighted Decision Tree Model	0.54	0.78	0.6391459064157986	0.9985210720631577

- **Confusion Matrix Comparison**
  Because it's intuitive from a business standpoint, I also looked into the performance difference among all confusion matrices from three models (Appendix). Overall, the weighted decision tree model has the lowest misclassification rate in identifying a good transaction as a "fraudulent" one.

Identify Top Features from Decision Tree Modeling

Other than potentially used in the production environment, the decision tree models also help identify top features are used for risk rules and policies.
The model importance analysis (Appendix) of all three models suggests features in V14, V17, V12, and V20 are possible candidates to implement risk rules around them.

Out-of-scope topics

With more time, I would also evaluate the following areas that are common in a production environment

- **Decision Tree Model Score Calibration**
  A common issue with the tree-based model is over-fitting. I would explore probability calibrations (e.g., Isotonic regression or Platt scaling) to avoid overfitting
- **Create synthetic data in the "fraud" category to address data imbalance**
  Use techniques such as SMOTE

Yan Zeng *Contact: dirkzeng@gmail.com* *Cell: (402)570-7945*

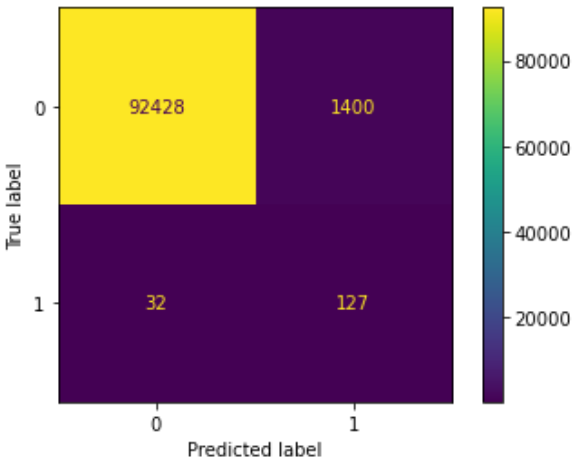
Appendix

Confusion Matrix Comparison

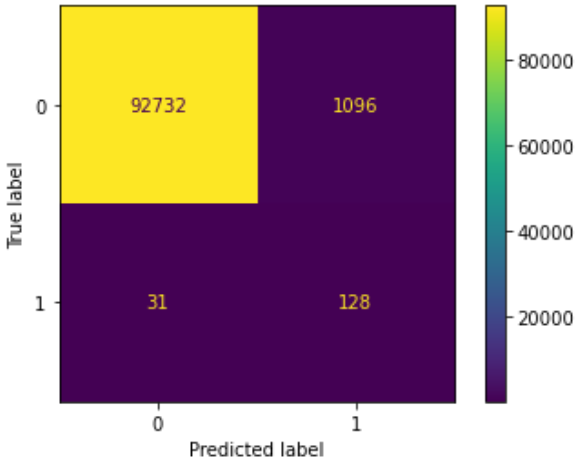
As shown in Figure 1, the confusion matrix of the weighted decision tree has the lowest “false positive” rate in misclassifying “good transactions” as “fraud”

Figure 1. Confusion Matrix Comparison

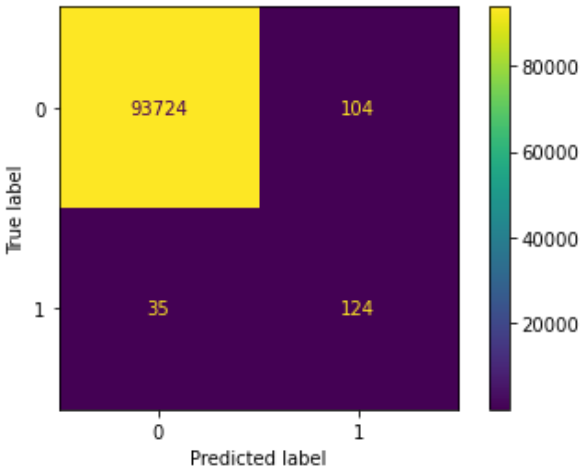
Baseline Model



Upsampling Model



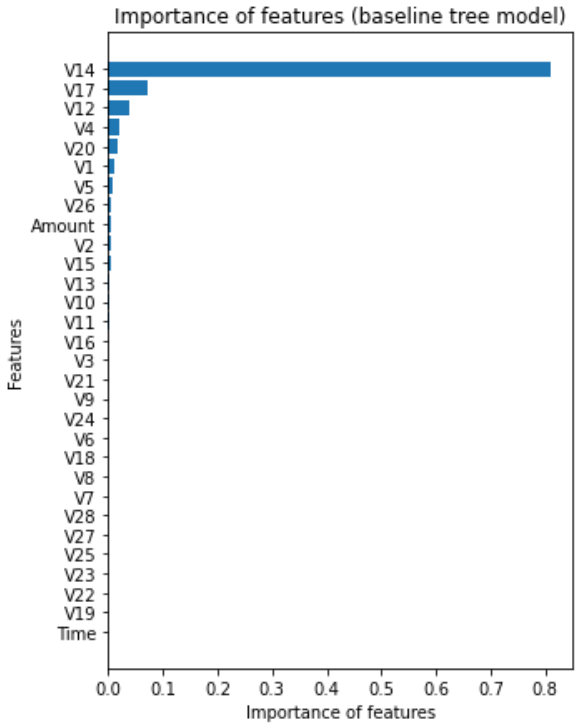
Weighted Model



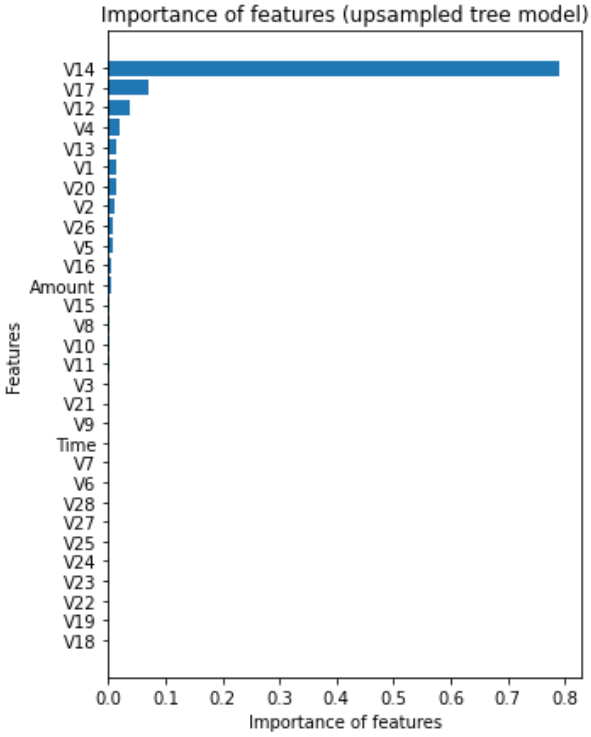
Yan Zeng *Contact: dirkzeng@gmail.com* *Cell: (402)570-7945*

Feature Importance Comparison

Baseline Model



Upsampling Model



Weighted Model