Capstone Proposal: Detecting Instances of Credit Card Fraud

Domain Background: This is a form of anomaly detection that is practical for the industry I want to go into. It is a common problem that many companies face. The number of major data breaches has increased from 470 in 2012 to 1091 in 2016.[1] For each of these instances, the amount of leaked information can be enormous. For example, the largest data breach in 2018 was from the Indian government, having over 1.1 billion names, unique identity numbers, and banking information was revealed.[2] Since any company can have a data breach, information on how to detect instances of fraud have become even more important. Since credit card transactions are not classified, and are often scarce in comparison to regular transactions, agencies need to use things such as anomaly detection, or outlier detection, to classify them.[3] One metric that is sued to create a consistency score. This is done by clustering the transactions and creating a consistency score to see the similarity between several data points. If the score is seen as inconsistent with the person's other transactions, it is flagged.

This particular problem is for the finance industry and is working on trying to detect credit card fraud. There are many jobs that are based on this exact problem and many utilize advanced ML models. Although there are instances of unsupervised learning and anomaly detection for fraud detection, this dataset is classified, so we will be using a classification method to detect these anomalies.

Problem Statement: Given this dataset, can we build a machine learning model to detect instances of credit card fraud? This will be a supervised classification problem

Datasets and Inputs: The dataset comes from the Kaggle Competition Credit Card Fraud Detection (https://www.kaggle.com/mlg-ulb/creditcardfraud). The dataset has 284,808 observations. The dataset is classified into whether observations were fraudulent or not. The dataset is highly imbalanced (482 fraudulent observations, 1.6%). The dataset for fraudulent activities is skewed towards very small transactions (under $5). The non-fraudulent rows are also skewed to smaller purchases, but not as strongly. The features of the dataset from a PCA of the original dataset. The datasets that are not transformed are Time, Amount, and whether or not the row is fraudulent or not (1 or 0 respectively). Time is the time elapsed between the first transaction in the dataset and the current transaction and amount is the amount of money in the given purchase.

---

[1] Steele, Jason. "Credit Card Fraud and ID Theft Statistics." *CreditCards.com*, 10 June 2019, www.creditcards.com/credit-card-news/credit-card-security-id-theft-fraud-statistics-1276.php.

[2] Katz, Eitan. "The 20 Biggest Data Breaches of 2018." *Dashlane Blog*, 9 Aug. 2019, blog.dashlane.com/data-breaches-2018/.

[3] Porwal, Utkarsh, and Smruthi Mukund. "Credit Card Fraud Detection in e-Commerce: An Outlier Detection Approach 1 / 10." *Https://Arxiv.org/*, Cornell University, 10 Oct. 2019, arxiv.org/pdf/1811.02196.pdf.

Benchmark: I will implement a number of different potential solutions to this. My benchmark model will be a basic logistic regression.

Solution Statement: The solutions will include visualizations and resampling the dataset to becoming more balanced. We will explore potentially using feature reduction (such as PCA) and potential feature engineering, but I am not sure if this is valid, since the columns are masked and (unfortunately the columns are masked, so this might not be possible or make sense).

The models I plan to build are logistic regression, random forest, XGBoost, and a neural network. If I have time, I will also be exploring tensor flow and more advanced deep learning models.

Evaluation metrics: Some of the metrics I will be using are sensitivity, f1-score, and AUPRC to evaluate if the dataset is accurate. If we were not going to rebalance the dataset, I would likely use f1-score, sensitivity or AUPRC. The reason why we would use sensitivity is because this would be used as a warning system for fraudulent activity. The cost of missing fraudulent activity (the activity continues unnoticed) is much higher than flagging normal activities (sending alerting the card holder and the card holder stating it is inaccurate). F1-score would be a good balance between specificity and sensitivity. AUPRC allows you to get a visualization to see how the model when comparing the trade-off between precision and recall across its different decision thresholds. The higher the area is under the model, the better the model.

Before the dataset is rebalanced, evaluation metrics like accuracy and specificity would not work, because if they the model says nothing is fraudulent, they would say it is almost 99% accurate/specific. After I rebalance the dataset, accuracy, precision, and confusion matrixes can be used on the training set. Since we are not up sampling the validation set, one of the prior metrics will be used.

Project Design: The project will be formed by exploring the dataset for various insights and correlations. I will create visualizations to display any of my findings. I will also clean the data for potential outliers and missing data. We will likely use randomization of distribution to impute missing data. I will also build a benchmark to see how a basic model would work. We will then perform cross validation to create a validation set. We will use tune the number of cross validations that should be used in our final model.

During this cross-validation step, we can either 1) up sample the fraudulent activities, 2) down sample the nonfraudulent activities, or 3) not do any sampling to the dataset. I would likely go with one of the two options because of how imbalanced the dataset is. If we do not do any rebalancing, the dataset will likely not have enough observations to be able to predict fraudulent cases. Both under sampling and oversampling has pros and cons. If we under sample, we will only use ‘real’ data which is good, but because there is so little fraudulent data, our sample sizes will be very small. If we up-sample, our sample size will be larger, but we will have to impute imaginary data or bias our model to data that is in our training set. This will help

with model accuracy in the test set, but may make it less accurate in the validation set or in the test set. Despite these cons, I think that we will likely use up sampling. How much we will up-sample will be likely tuned as well. If we have time, we will likely try all three methods to see how the performance is affected.

Because the dataset is rather skewed and appears to have some outliers, we will consider dropping outliers to avoid skewing our results. Whether or not we drop the dataset, we will normalize our dataset so the data is skewed. There are instances where non-normalized data is a more powerful predictor of fraud, so we will likely run the model on both versions of the dataset. After this, I would explore the ability to feature engineer or reduce features of the dataset. Feature engineering may not be possible because the data is already run through PCA to get to this input, but we can potentially drop uninformative features. I will then create more advanced models and finetune their configurations through either Bayesian optimization or grid search. We will submit our model to Kaggle to see how it compares to their test dataset and see how we rank.