2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI) | 978-1-6654-3875-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/IRI51335.2021.00054

Impact of Hyperparameter Tuning in Classifying Highly Imbalanced Big Data

John Hancock and Taghi M. Khoshgoftaar
College of Engineering and Computer Science
Florida Atlantic University
Boca Raton, Florida 33431
jhancoc4@fau.edu, khoshgof@fau.edu

***Abstract*—Working with Machine Learning algorithms and Big Data, one may be tempted to skip the process of hyperparameter tuning, since algorithms generally take longer to train on larger datasets. Hyperparameter tuning is not something we get for free; one must allocate either more computing time or resources to run more iterations of experiments with different hyperparameter settings to find their optimal values. For small datasets the extra resources needed may be negligible, but for larger datasets resources necessary for tuning may be considerable. In this study, due to the size of the data we use, we find experiments where we do hyperparameter tuning take many times longer to complete than experiments where we do not do any tuning. Here, our contribution is to provide evidence that the investment in computing resources for hyperparameter tuning pays off, since we show that, for experiments involving highly imbalanced Big Data, we obtain better results when we incorporate hyperparameter tuning. With regard to the performance of CatBoost and LightGBM classifiers, we compare both default and tuned hyperparameter values. Furthermore our experiments encompass different techniques for encoding high-cardinality categorical features. We find in all cases, regardless of the classifier or encoding technique for categorical features, classifiers with tuned values for hyperparameters yield better results than those with default values. To the best of our knowledge, we are the first to do such a study on hyperparameter tuning to analyze the performance of LightGBM and CatBoost in classifying highly imbalanced Big Data.**

***Keywords*—Gradient Boosted Decision Trees, CatBoost, LightGBM, Class Imbalance, Medicare Fraud, Anomaly Detection**

1. Introduction

An abundance of data is a blessing and a curse for Machine Learning (ML). On one hand, more training instances enable us to train more sophisticated models. For example, the introduction of the Imagenet [1] dataset precipitated breakthroughs in image classification. Without Imagenet, researchers may not have been able to realize those breakthroughs [2]. However, training models with large datasets comes with a cost; one must allocate computing resources sufficient to process the data, and train ML algorithms with it. Many ML algorithms also include adjustable hyperparameters [3]. One may face a dilemma of whether to spend time and money discovering optimal values of hyperparameters for particular ML tasks. Here, we intend to answer the research question: does hyperparameter tuning have a significant impact on classifier performance for large imbalanced datasets? For the impatient, the answer to that question is "Yes." For those who are curious as to how we arrive at that conclusion, we provide a comprehensive explanation in the pages that follow.

The datasets we select for our experiments are constructed from the Centers for Medicare and Medicaid Services Part D Prescriber Public Use Files [4]. Hereafter, we refer to this data as the Part D data. The Part D data meets the definition of Big Data according to [5]. Furthermore, the Part D data is tabular and heterogeneous, containing both categorical and numeric features. The data has one feature – Drug Name – that has thousands of possible values. Thanks to the built-in support in CatBoost and LightGBM for encoding categorical features, as mentioned previously, we can use the Drug Name feature as direct input for both LightGBM and CatBoost. For a survey on techniques for encoding categorical features, please see [6]. Also, since the Part D data is heterogeneous, according to the authors of the seminal paper [7] on CatBoost, such data is well-suited for processing by decision tree-based ML algorithms, such as Gradient Boosted Decision Trees (GBDTs) [8]. For an in-depth review of applications of CatBoost, please see the survey [9]. Hence, we select CatBoost and another GBDT implementation, LightGBM, for comparison in the task of classifying this Part D data for Medicare Fraud detection.

Medicare fraud detection is a worthwhile pursuit, since we suspect corrupt healthcare providers bill Medicare for services they did not provide, unnecessary services, and so on. In aggregate, these nefarious activities add up to what would appear to be the misappropriation of billions of dollars in taxpayer money. In 2017, the United States Government Accountability Office reported to Congress that the Centers for Medicare and Medicaid Services (CMS) made $52 billion in improper payments [10]. Some part of the improper payments are payments to fraudulent healthcare providers.

Here, we contribute to the study of Medicare fraud

978-1-6654-3875-9/21/$31.00 ©2021 IEEE
DOI 10.1109/IRI51335.2021.00054

detection in several ways. First, we provide data on the performance of LightGBM and CatBoost in the task of Medicare fraud detection to assist researchers in determining which GBDT implementation yields the best performance. Furthermore, we show that the hyperparameter tuning of classifiers for Medicare fraud detection is worth the allocation of the requisite additional resources, since we obtain better performance from both classifiers when we tune hyperparameters. In fact, we find performance is better in a statistically significant sense. Hyperparameter tuning consumes more processing time, or more system resources depending on how it is implemented; sequentially, or in parallel. One can try different combinations of hyperparameters iteratively, in which case they will consume more processing time, or one can run experiments where the classifiers have different values for hyper-parameters in parallel, in which case they will consume more computing resources. Since the size of the dataset we are training on already exhausts available system resources, we are forced to do tuning sequentially. Our experience in conducting the experiments in this study shows that experiments that complete in hours without hyperparameter tuning require several days to complete, and some would require more than one week to complete. The remainder of the present work is organized as follows: first we review related works, then we discuss the data we use, and its preparation for ML tasks. Then, we cover the methodology of our experiments. After that, we present results and statistical analysis to draw conclusions.

1.1. Related Work

Here we discuss works that are pertinent to this study. One similar study is "Influence of Optimizing XGBoost to Handle Class Imbalance in Credit Card Fraud Detection" [11]. In their study, Priscilla and Prabha evaluate XGBoost [12] with several combinations of sampling techniques and hyperparameter optimization for credit card fraud detection. While Priscilla and Prabha report that they use a technique similar to the one we use to discover optimal values for hyperparameters, we do not find where they report ranges of values searched during optimization. The datasets they use are similar to ours in that they are imbalanced, but the Medicare data we use here has over ten times the number of samples. Furthermore, Priscilla and Prabha use only XGBoost, and here we evaluate two learners: CatBoost and LightGBM.

In "Semantic Embeddings for Medical Providers and Fraud Detection" [13], Johnson and Khoshgoftaar investigate Medicare fraud as we do here, and they cover techniques for representing categorical features in Medicare data. In their study, they create several datasets using various methods of encoding the Provider Specialty feature in Medicare data. Then, they compare the performance of several learners in combination with the different datasets. Here we are more interested in the impact of hyperparameter tuning. So, we either use provider-level Medicare data augmented with summary statistics of procedure-level features, or we use learners' built-in encoding to handle categorical features.

In an earlier study Bauder and Khoshgoftaar [14] treat fraud detection in imbalanced Medicare Big Data as a regression problem. They use some of the same publicly available Medicare data we use here, to build regression models to predict the expected Medicare payment for services provided. When the error the regression model makes in estimating payment is large, the provider is labeled fraudulent. The reasoning behind their technique is that if providers are over-billing Medicare, their regression model will predict a much lower amount billed than the actual amount billed; hence, indicating fraudulent activity. Here, we treat fraud detection as a classification problem.

Another work that takes a statistical approach to detect Medicare fraud is Ekin *et al*. [15]. In their study the authors employ a "concentration function" that measures the concentration of Medicare payments to healthcare providers. The curve of the concentration function is an indicator of how uniformly payments are spread out among providers. When payments are not distributed uniformly, we take it as an indicator that a provider is fraudulently collecting more payments than the provider is entitled to. Ekin *et al*. do a case study to apply the concentration function on a cohort of a small number of doctors which works well, but we do not find in their study where they apply their technique on a large scale as we do here.

Another approach to anomaly detection in Medicare Big Data is documented in Branting *et al* [16]. In their study, the authors use graph embedding techniques to model relationships between healthcare providers, medications, procedures, and their locations. Then, they derive features related to providers based on attributes of the graphs' representations of the providers, and use these features to train classifiers to predict whether the provider appears in the List of Excluded Individuals and Entities (LEIE) [17]. Their study uses data from 2012 through 2014. Branting *et al*. conclude provider location is a strong predictor of fraudulent activity; however, models that use provider location as a feature are susceptible to unfairly labeling providers in specific locations as fraudulent. Here, we do not leverage provider location data.

In a different approach to combining the LEIE and the Public Use Files (PUF) data from CMS to do Medicare Fraud detection, Herland *et al*. [18], use the LEIE to label Part B, Part D, and DMEPOS Medicare data to create datasets amenable to classification by ML algorithms. Their study encompasses data from 2012 through 2015. The learners they employ are the Apache Spark implementations of Logistic Regression, Gradient Boosted Trees, and Random Forest. Our dataset incorporates data from 2013 through 2018. Moreover, we focus on hyperparameter tuning in the recent GBDT implementations CatBoost and LightGBM. Since these learners have their own built-in encoding techniques for categorical features, we are capable of using the individual records of the Part D dataset, whereas Herland *et al*. choose to use data aggregated at the provider level.

Another study that investigates sampling techniques for

fraud detection in Medicare Big Data is [19]. In their study, Su *et al*. combine feature engineering techniques with sampling techniques to come up with a framework for Medicare fraud detection. The feature engineering part of their survey centers around the discovery that if we represent healthcare providers as points in the space where one coordinate is the number of services provided, and the second coordinate is the number of procedure codes associated with the provider, the points representing fraudulent providers will lie apart from those points representing legitimate providers. Su *et al*. derive four novel features for their dataset based on this abstract representation of healthcare providers and services they offer. The other part of their framework is related to sampling techniques one would use to enhance classification results for highly imbalanced datasets. It includes two adjustable parameters to control the amount of oversampling and undersampling performed before model training. While Su *et al*. focus on feature engineering and sampling techniques, here we are concerned with the impact of hyperparameter tuning for GBDT learners in classifying Medicare Big Data. Also, Su *et al*. report results for datasets from 2013 and 2014. Here, our results encompass bigger datasets that have data from 2013 through 2018.

1.2. Algorithms

The focus of this study is the importance of hyperparameter tuning; therefore, ML algorithms (learners) play a vital role. The learners in this study are both Gradient Boosted Decision Tree GBDT implementations. GBDT learners are a class of ensemble techniques that leverage collections of decision trees to do ML tasks. The key aspect that GBDT learners have in common is that the ensemble is built iteratively, and the next tree to be added to the ensemble is the one that minimizes the value of the gradient of a loss function defined to evaluate the performance of the ensemble, as it is formed. For more detail on GBDTs please see Friedman's study on them, [8]. GBDTs are not the only supervised learning technique that performs well in the task of Medicare fraud detection. For example, Johnson and Khoshgoftaar apply Neural Networks in [20]. GBDTs are supervised learners. For an example of an unsupervised approach to Medicare fraud detection please see [21]. The first GBDT implementation we use here is CatBoost [7]. CatBoost is a recent GBDT implementation that incorporates optimizations for encoding categorical features, and avoiding overfitting, that its predecessor GBDT implementations might be susceptible to. For encoding categorical features, Prokhorenkova *et al*. introduce Ordered Target Statistics (TS). Ordered TS is a technique that uses class labels to encode categorical features, but ensures that the encoding of a specific feature does not contain any information about the label associated with that feature. Similarly, to avoid overfitting in adding the next tree to the GBDT ensemble, CatBoost employs Ordered Boosting, that ensures labels used to fit the next tree in the ensemble are not used to estimate the gradient of the loss function defined on the output of the ensemble, when selecting the best tree to add to the ensemble.

The second GBDT implementation we use is LightGBM [22]. Chronologically, LightGBM was released prior to CatBoost. LightGBM incorporates two key optimizations to the GBDT algorithm that enable it to yield accuracy similar to other GBDT implementations, but it has shorter training times than other GBDT implementations. The first optimization is Exclusive Feature Bundling (EFB). EFB is a technique for reducing the number of features in a dataset by combining sparse features that do not take the same values simultaneously. GBDTs train faster on datasets with fewer features. The second optimization in LightGBM is Gradient-based One-Side Sampling (GOSS). Since GBDTs add Decision Trees to the ensemble based on the gradient of a loss function, the training examples that contribute more to the gradient of the loss function have more of an impact on how GBDTs select the next tree to add to the ensemble. Under GOSS, we select a threshold value for the amount a training example contributes to the loss function, and we retain all training examples that contribute more than the threshold value, but only a sample of those examples that contribute less than the threshold values. This concludes our discussion on related works and background information. Now, we move on to discuss how we carry out our experiments.

2. Methodology

Here, we cover the important techniques we use to conduct our experiments. First, we cover how we treat the Part D data in order to prepare it for experiments. After that, we go over details on software utilized to conduct our experiments, including the library we use for hyperparameter tuning of both CatBoost and LightGBM.

2.1. Data Sources

As mentioned previously we use Medicare Part D data for experiments. The Part D data contains information on healthcare providers, and the medications they prescribe for their patients in aggregate, at the level of a particular medication, for all the provider's patients for the year. The Part D data comes as a Character Separated Value (CSV) file. One row in the CSV file corresponds to how a provider prescribed one medication to all patients for one year. Please see Table 1 for a description of the Part D data in the state it is in as published by CMS. We list the subset of features relevant to the present work; for descriptions of all columns, please see [4].

For half of our experiments, we use Part D data aggregated at the provider level. We aggregate the Part D data with a technique similar to the one used by Bauder and Khoshgoftaar in [23]. The technique is as follows: first we collate the data into groups with the same NPI, provider type and year, then we calculate aggregated figures over the collections of numeric features in the groups. For each numeric feature we calculate the mean, median, sum,

TABLE 1. FEATURES FROM PART D DATA FROM CMS

Name	Description	Type
National Provider ID (NPI)	Not applicable	Used for labeling and aggregating, discarded before classification
provider_type	Code for Medical provider's practice or specialty	Categorical
drug_name	Name of medication prescribed	Categorical
bene_count	The sum of then number of unique Medicare Part D beneficiaries having one or more claims for the drug.	Numeric
total_claim_count	The number of claims filed by this provider for this medication	Numeric
total_30_day_fill_count	The number of 30-day refills the prescriber writes	Numeric
total_day_supply	The sum of the number of days in all prescriptions	Numeric
total_drug_cost	Dollar amount sum of cost for all prescriptions	Numeric

standard deviation, minimum and maximum values, and we replace the numeric feature with the aggregated figures. We also discard the provider_type categorical feature, so that all remaining features are numeric. This is how we process Part D data to generate what we refer to as the Part D data of aggregated record type.

For the other half of our experiments, we simply extract the columns listed in Table 1, and we indicate to CatBoost and LightGBM that the provider_type and drug_name features are categorical. To mark a feature as categorical for CatBoost, we pass a list of categorical feature names to the CatBoostClassifier constructor function. For LightGBM, we set the data type of the categorical features in the data structure we pass to LightGBM to "category". We refer to data handled this way as data of individual record type.

Whether we are working with the data of aggregated record type or individual record type, we must label records to indicate the provider is fraudulent, or not fraudulent. We accomplish this by joining the Part D data to the LEIE data. When a healthcare provider appears in the LEIE it means the provider is banned from submitting claims to Medicare for violating rules on properly submitting claims to Medicare. The LEIE also contains information on the period of time for which the provider is excluded from submitting claims to Medicare. We apply logic that takes into account that date range to determine if the record for the provider should be labeled as fraudulent. Once we have labeled all the Part D data, we discard the NPI, since identifiers are not appropriate features for ML algorithms.

Once we have the data labeled, we can report class ratios. We find the data is imbalanced. We report precisely how imbalanced by giving the ratios of positive to negative samples in Table 2.

This finishes our discussion on data and preprocessing.

TABLE 2. IMBALANCE RATIOS OF DATASETS

Record Type	Positive count	Negative count	Ratio
Aggregated	3,209	5,273,376	0.00060853
Individual	149,353	147,628,286	0.00101168

Now we move on to cover methods we use to conduct experiments.

2.2. Experimental design

We run all experiments as Python [24] programs. Furthermore, we rely heavily on the Scikit-learn [25] Python library for designing experiments. As alluded to earlier, we utilize Python libraries for CatBoost and LightGBM. For experiments where hyperparameter tuning is performed, we use the Scikit-learn RandomizedSearchCV object to handle tuning. We employ RandomizedSearchCV with LightGBM and CatBoost according to the pattern Buitinck *et al*. demonstrate with RandomizedSearchCV, and another ML algorithm in [26]. When we mention experiments with "tuned hyperparameters", we mean that we use the technique given by Buitinck *et al* in those experiments. For all experiments, we conduct 6 iterations of 5-fold cross validation to obtain a total of 30 Area Under the Receiver Operating Characteristic Curve (AUC) [27] results. For all experiments with CatBoost, we use the GPU implementation of CatBoost. For experiments with LightGBM we use the CPU implementation of it. The hyperparameter values we selected for tuning are based on suggestions from the online documentation for LightGBM [28] and CatBoost [29]. Please see Tables 3 and 4 for listings of hyperparameter values we use.

TABLE 3. CATBOOST HYPERPARAMETER TUNING VALUES

Parameter name	Values sampled	Parameter description (Default value)
depth	4,5,6,7,8	Controls maximum depth of decision tree (6)
l2_leaf_reg	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 50, 100, 200	Coefficient for regularization term (3)
auto_class_weights	None, Balanced, SqrtBalanced	Selects class weight formula used (None)
n_estimators	100,250,500,1000	Number of trees in the ensemble (1000)

We conduct a total of 8 experiments. We have two datasets: the Part D data of aggregated record type, and the Part D data of individual record type. For each dataset we conduct four experiments, one for each of: CatBoost with default hyperparameters, CatBoost with tuned hyperparameters, LightGBM with default hyperparameters, and LightGBM with tuned hyperparameters. As mentioned above, for experiments with data of individual type records, we use

TABLE 4. LIGHTGBM HYPERPARAMETER TUNING VALUES

Parameter name	Values sampled	Parameter description (Default value)
max_depth	Random integer between 10 and 50	Controls maximum depth of decision tree (No limit)
min_child_weight	1×10^{-5}, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 1×10^{4}	Controls weight of instances necessary to form leaf node in Decision Tree (1×10^{-3})
subsample	Random number drawn uniformly from $[0.2, 0.8]$	Ratio of subsample size to take from training instances (1.0)
reg_lambda	0, 0.1, 1, 5, 10, 20, 50, 100	Coefficient for regularization term (0.0)

CatBoost and LightGBM's built-in encoding for categorical features.

This concludes the discussion on data sources and experimental design. We provide it to ensure our results are reproducible. Now we move on to present results and analysis thereof.

3. Results

Table 5 lists results of the 8 experiments in pairs, so that results for models with tuned parameters are directly beneath results for the same model with default values for hyperparameters. In all cases, we see that hyperparameter tuning produces better results.

Table 5 establishes that mean AUC values for experiments with models that have tuned hyperparameters are higher than those of experiments with models having default values for hyperparameters. We check whether the differences are due to random chance, or if they are statistically significant. We accomplish that with a series of z-tests [30], where we test for the difference in the mean AUC values that learners with default values for hyperparameters yield, and mean AUC values that learners with tuned values for hyperparameters yield.

TABLE 5. MEAN AND STANDARD DEVIATION (SD) OF PERFORMANCE FOR 30 ITERATIONS OF 5-FOLD CROSS VALIDATION

Classifier	Record Type	Encoding	Tuning	Mean AUC	SD AUC
CatBoost	Aggregated	N/A	N	0.75765	0.00994
CatBoost	Aggregated	N/A	Y	0.78637	0.01185
CatBoost	Individual	Built-in	N	0.70336	0.00293
CatBoost	Individual	Built-in	Y	0.71573	0.00424
LightGBM	Aggregated	N/A	N	0.64758	0.01858
LightGBM	Aggregated	N/A	Y	0.79584	0.00729
LightGBM	Individual	Built-in	N	0.72506	0.00146
LightGBM	Individual	Built-in	Y	0.72548	0.00119

Table 6 shows that for the $\alpha = 0.05$ significance level, the difference in mean AUC values for models with tuned hyperparameters is statistically significant in each experiment. The increase in the mean AUC value we see for LightGBM with aggregated data is interesting. This result makes the strongest case for hyperparameter tuning out of all experimental outcomes.

TABLE 6. z-TESTS FOR SIGNIFICANT DIFFERENCES IN PERFORMANCE

Classifier	Record Type	z-test p-value	Tuned Performance
CatBoost	Aggregated	0.00000	Better
CatBoost	Individual	0.00000	Better
LightGBM	Aggregated	0.00000	Better
LightGBM	Individual	0.01459	Better

Each row compares classifiers with default values for hyperparameters versus classifiers with tuned vales for hyperparameters; We use $\alpha = 0.05$ for the significance level.

Another result we obtain from experiment outcomes are the actual values of hyperparameters that RandomizedSearchCV discovers. After a program completes execution of RandomizedSearchCV's "fit" function, the RandomizedSearchCV object has a "best_params_" member. The best_params_ member is a data structure that contains optimal values for hyperparameters that RandomizedSearchCV discovers. Therefore, after each experiment iteration we save the best_params_ data structure. In Table 7 we report the most likely to occur, or mode, of each optimal value RandomizedSearchCV finds for the 30 experimental outcomes of our study. The mode is the value others would be most likely to obtain in attempts to reproduce our results. Tables 5 through 7 encompass all the results we have to share for this report.

TABLE 7. MODE VALUES FOR HYPERPARAMETERS FOUND FROM TUNING

Classifier	Record Type	Hyperparameter	Mode
CatBoost	Aggregated	n_estimators	1000
CatBoost	Aggregated	l2_leaf_reg	100
CatBoost	Aggregated	depth	4
CatBoost	Aggregated	auto_class_weights	Balanced
CatBoost	Individual	n_estimators	1000
CatBoost	Individual	l2_leaf_reg	2
CatBoost	Individual	depth	7
CatBoost	Individual	auto_class_weights	Balanced
LightGBM	Aggregated	max_depth	36
LightGBM	Aggregated	min_child_weight	0.00100
LightGBM	Aggregated	reg_lambda	10
LightGBM	Aggregated	subsample	0.21778
LightGBM	Individual	max_depth	35
LightGBM	Individual	min_child_weight	100.00
LightGBM	Individual	reg_lambda	50
LightGBM	Individual	subsample	0.22558

4. Conclusion

We learn from this study that hyperparameter tuning for experiments involving highly imbalanced Big Data can have a significant impact on experiment results. Therefore, the answer to our research question posed at the beginning of this study, "Does hyperparameter tuning have a significant impact on classifier performance for large imbalanced datasets?" is "Yes." Although it is true that ML models take more time and computing resources to train on larger datasets, we show that one can obtain better results with hyperparameter tuning, with two different GBDT implementations: LightGBM, and CatBoost. Not only are results better, but we show results are better in a statistically significant sense. Furthermore, we showed this to be the case with highly imbalanced, Big Data. We did not apply sampling techniques to address class imbalance, so one avenue for future research is to incorporate sampling techniques into experiments we conduct here. Another possibility for future research would be to incorporate the Provider State feature as in [31], provided one can show this does not bias learners to specific geographic areas. The key take-away from our study is that although libraries that furnish ML algorithm implementations have sensible default values for hyperparameters, to allow for the best potential performance, one should attempt hyperparameter tuning even though a dataset may be considered Big Data.

Acknowledgment

The authors would like to express their gratitude to the reviewers at the Data Mining and Machine Learning Laboratory of Florida Atlantic University for the help in preparing this study.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[3] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[4] CMS Office of Enterprise Data and Analytics, "Medicare fee-for service provider utilization & payment data part d prescriber public use file: A methodological overview," 2020.

[5] A. De Mauro, M. Greco, and M. Grimaldi, "A formal definition of big data based on its essential features," *Library Review*, 2016.

[6] J. T. Hancock and T. M. Khoshgoftaar, "Survey on categorical data for neural networks," *Journal of Big Data*, vol. 7, pp. 1–41, 2020.

[7] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 6638–6648. [Online]. Available: http://papers.nips.cc/paper/7898-catboost-unbiased-boosting-with-categorical-features.pdf

[8] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[9] J. T. Hancock and T. M. Khoshgoftaar, "Catboost for big data: an interdisciplinary review," *Journal of Big Data*, 2020.

[10] S. J. Bagdoyan. (2018) Testimony before the subcommittee on oversight, committee on ways and means, house of representatives. [Online]. Available: https://www.gao.gov/assets/700/693156.pdf

[11] P. C. Victoria and P. D. Padma, "Influence of optimizing xgboost to handle class imbalance in credit card fraud detection," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2020, pp. 1309–1315.

[12] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016. [Online]. Available: http://dx.doi.org/10.1145/2939672.2939785

[13] J. M. Johnson and T. M. Khoshgoftaar, "Semantic embeddings for medical providers and fraud detection," in *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE, 2020, pp. 224–230.

[14] R. A. Bauder and T. M. Khoshgoftaar, "A novel method for fraudulent medicare claims detection from expected payment deviations (application paper)," in *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, July 2016, pp. 11–19.

[15] T. Ekin, F. Ieva, F. Ruggeri, and R. Soyer, "On the use of the concentration function in medical fraud assessment," *The American Statistician*, vol. 71, no. 3, pp. 236–241, 2017.

[16] L. K. Branting, F. Reeder, J. Gold, and T. Champney, "Graph analytics for healthcare fraud risk estimation," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Aug 2016, pp. 845–851.

[17] LEIE. (2017) Office of inspector general leie downloadable databases. [Online]. Available: https://oig.hhs.gov/exclusions/index.asp.

[18] M. Herland, T. M. Khoshgoftaar, and R. A. Bauder, "Big data fraud detection using multiple medicare data sources," *Journal of Big Data*, vol. 5, no. 1, pp. 1–21, 2018.

[19] Y. Su, X. Zhu, B. Dong, Y. Zhang, and X. W. Wu, "Medfrodetect: Medicare fraud detection with extremely imbalanced class distributions," in *The Thirty-Third International FLAIRS Conference (FLAIRS-32)*, 2020.

[20] J. M. Johnson and T. M. Khoshgoftaar, "Medicare fraud detection using neural networks." *Journal of Big Data*, vol. 6, no. 1, pp. 1 – 35, 2019.

[21] R. A. Bauder, R. da Rosa, and T. M. Khoshgoftaar, "Identifying medicare provider fraud with unsupervised machine learning," in *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, July 2018, pp. 285–292.

[22] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3146–3154. [Online]. Available: http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

[23] R. A. Bauder and T. M. Khoshgoftaar, "The effects of varying class distribution on learner behavior for medicare fraud detection with imbalanced big data," *Health information science and systems*, vol. 6, no. 1, p. 9, 2018.

[24] G. Van Rossum and F. Drake, "Python 3 reference manual createspace," *Scotts Valley, CA*, 2009.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[26] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[27] M. Bekkar, H. K. Djemaa, and T. A. Alitouche, "Evaluation measures for models assessment over imbalanced data sets," *J Inf Eng Appl*, vol. 3, no. 10, 2013.

[28] Microsoft Corporation. Parameters tuning. [Online]. Available: https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html.

[29] Yandex. Parameter tuning. [Online]. Available: https://catboost.ai/docs/concepts/parameter-tuning.html.

[30] L. Tom, F. Anna, and V. A.N., *Research and Statistics for Social Workers*. Routledge, 2019.

[31] J. T. Hancock and T. M. Khoshgoftaar, "Gradient boosted decision tree algorithms for medicare fraud detection," *SN Computer Science*, vol. 2, no. 4, pp. 1–12, 2021.