Accredited Ranking SINTA 2
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online on: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 7 No. 1 (2023) 129 - 137 ISSN Media Electronic: 2580-0760

A Comparative Study of CatBoost and Double Random Forest for Multi-class Classification

Annisarahmi Nur Aini Aldania[1], Agus Mohamad Soleh[2*], Khairil Anwar Notodiputro[3]
[1,2,3]Department of Statistics, Faculty of Mathematics and Natural Science, IPB University
[1]annisarahmi.nur@gmail.com, [2]agusms@apps.ipb.ac.id, [3]khairil@apps.ipb.ac.id

Abstract

Multi-class classification has its challenge compared to binary classification. The challenges mainly caused by the interactions between explanatory and responses variable are increasingly complex. Ensemble-based methods such as boosting and random forest (RF) have been proven to handle classification problems. We conducted this research to study multi-class classification using CatBoost, a method developed with gradient boosting and double random forest (DRF), RF's development that is good to be used when the resulting RF model is underfitting. Analysis was carried out using simulation and empirical data. In the simulation study, we generate data based on the distance between classes: high, medium, and low. The empirical data used is the industrial classification code, namely KBLI. CatBoost and DRF can rightly solve the multi-class classification problem at a high distance, measured by a 100% balanced accuracy score. At a medium distance, CatBoost and DRF produce balanced accuracy scores of 99.25% and 97.54%, respectively, whereas 32.37% and 23.97% at the low distance. In empirical studies, CatBoost's performance outperforms DRF by 4.27%. All the differences are statistically significant based on the t-test result. We also use LIME to explain individual predictions of CatBoost and learn words that contribute the most to an example class's prediction.

Keywords: catboost, double random forest, multi-class classification, lime

1. Introduction

Multi-class classification is a predictive modeling for qualitative response variables with more than two predicted classes. Determining the relationship between classes in a multi-class classification is more complex than in a binary classification [1]. This complexity makes it more challenging to produce good predictions. Some things that affect the difficulty of multi-class classification are interaction patterns that become more complicated between independent variables and response variables; features or independent variables that are many but not significant; the number of observations in each class is limited [2].

Decision tree-based methods can be used to overcome classification problems. While simple and easy to understand, the predictive ability of a single decision tree is not as good as some other machine learning methods. Thus, an ensemble method that combines several single trees was developed to produce better predictions. Furthermore, [3] and [4] show that combining trees gives better results than a single tree. Even though there are drawbacks when visualizing and interpreting the results compared to a single classification tree, the ensemble method could be a choice if the focus is on prediction and not interpretation [5]. Compared to other methods that can also provide good predictions, such as deep learning, for a small dataset, ensemble decision trees tends to have better performance and be more efficient in computation [6].

Boosting and random forest are examples of ensemble implementations. Boosting is an ensemble technique that combines several trees sequentially using information from the previous tree [7]. The combined trees are weak learners with low accuracy formed from residual models to increase predictive ability in areas with poor performance [8]. Boosting then becomes the basis for gradient boosting, which minimizes loss function.

There are several developments based on gradient-boosting decision trees, such as XGBoost [9], LightGBM [10], and CatBoost [11]. Some differences between CatBoost and other gradient boosting-based algorithms are CatBoost has two boosting modes, namely plain and ordered; it can process categorical features directly with ordered target statistics encoding;

Accepted: 15-12-2022 | Received in revised: 16-01-2023 | Published: 03-02-2023

it uses a symmetrical tree that is not easily overfitted. Another exciting aspect about CatBoost compared to XGBoot and LightGBM based on [11] is that the runtime process on the GPU, which the author use epsilon data for comparison, is even several times faster (25 times faster than XGBoost and around 60 times faster than lightGBM). The CatBoost algorithm has proven superior to other boosting methods in classifying imbalanced multi-class data [12]. Research [13] classifies classic Chinese Dongba documents into six categories and shows a better level of precision of CatBoost than random forest, KNN, decision trees, XGBoost and TensorFlow.

In contrast to boosting, random forest (RF) combine trees in parallel using bootstrap resampling [14]. The final prediction of RF is based on the majority vote of all trees formed. There are many developments from the RF model, one of which is the double random forest (DRF). DRF can produce better predictions when the resulting RF model is underfitting [15]. DRF performance was compared to thirty-four binary and multi-class classification data that had the possibility of underfitting using RF and showed that most DRF outperformed single classification tree, bagging, samme, and RF [15].

Multi-class classification can be applied to determine the industry classification code. Industry classification can be interpreted as a rule or principle in classifying an economic activity of every establishment (business or company) into a specific class code. In Indonesia, BPS – Statistics of Indonesia published the Indonesian industrial classification code (KBLI) as a guideline for classifying establishments based on their economic activity. According to the KBLI guidelines, every establishment can be classified into a five-digit number in the KBLI group based on their main activities and products. In determining the five-digit KBLI group code, sufficient knowledge is required so the misclassification code will not occur.

This study investigates CatBoost and DRF's performance on multi-class classification problems. The performance of both models will be calculated using a balanced accuracy test, false positive rate, and imbalanced accuracy metrics [16]. The study will use simulated and empirical data.

Simulation data will form a multi-class classification scenario, for each class will be close to the proportion of empirical data with the distance between classes: low, medium, and high. Simulation data is also designed so that RF will produce a model that can underfit the data. The 2015 Indonesian industrial classification code (KBLI) will be used for empirical data. On empirical data, RF could also make an underfit model based on the calculation of relative accuracy (explained in section 2). Because the data we use is underfitting with RF, we want to use DRF, which could make a better prediction under this condition—furthermore, we compared it with CatBoost, which also proved superior in handling multi-class classification. Ultimately, this study shows that CatBoost can overcome DRF both on simulation and empirical data.

2. Research Methods

Studies using simulation data were carried out to examine the performance of the two algorithms on controlled factors. and to study the model's characteristics. Furthermore, empirical data is used to examine the algorithm's performance in actual conditions.

2.1 Simulation Study

Simulation data is generated with a total class $C = 27$, the number of features or independent variables $p = 300$, and the total data generated $N = 800$. The amount of data per class in the simulation data will be grouped into five levels based on different ratios approaching empirical data, which are 1%, 5%, 6%, 10%, and 30%.

Table 1. Total Number of Data per Class

Level	Class	Number of classes per level	Data ratio per class (%)	Number of data per class
1	0 – 18	19	1	8
2	19 – 21	3	5	40
3	22	1	6	48
4	23 – 25	3	10	80
5	26	1	30	240

Table 1 summarizes the data generation structure for each class. Each of the first 19 classes (class 0 to class 18) has only a 1% data ratio compared to total data *N*. Hence, we only generate 8 data for each first 19 classes. Class 26 has the most observations, with a ratio of 30% of the total data *N*; this class has 240 observations.

The data imbalance ratio (IR) can be calculated using formula 1.

$$IR = \hat{p}_{max}/\hat{p}_{min} \quad (1)$$

with $\hat{p}_{max}$ and $\hat{p}_{min}$ are the maximum and minimum values of $\hat{p}$. In contrast, $\hat{p}$ is the ratio of data in each class. So, the IR value in simulation data is 30. This number shows that for every example in class 0, there are 30 examples in class 26.

This study will generate independent variables or features in each class from a normal distribution with a mean based on the centroid formed and a standard deviation of 1. All independent variables will be numerical data. The centroid of each class does not overlap and will have a vector size $1 \times p$ consisting of combinations of values 0 and 1, which are formed randomly. Furthermore, s is used as a centroid multiplier factor to determine the distance between classes. The final centroid calculation is based on the formula: *centroid* = *centroid* × 2 × *s*, then *centroid* =

DOI: https://doi.org/10.29207/resti.v7i1.4766

centroid – *s*. Three scenarios will be formed on the simulation data based on the value *s* between classes: high *s* =2, medium *s*=1, and low 0.5. The separation of classes will increase alongside the value of *s*.

For a more precise illustration, we simulate generating a high-distance scenario with the multiplier factor *s*=2. If there are two classes with two independent variables, the first step is to form two 1 × 2 centroid initiation vectors. Each vector represents the centroid in each class. Says (1,1) is the initial vector value for class 1 and (1,0) for class 2.

Initiated vector can be seen in Figure 1(a). The multiplier factor *s* will shift the initial centroid initiation position according to the previous formula. That step will be resulting the final centroid vector in class 1 having a value of (2, 2) and in class 2 having a value of (2, -2), as in Figure 1(b). This final centroid vector will then be used as the mean value in each class for generating data from a normal distribution with a standard deviation of 1, as in Figure 1 (c).

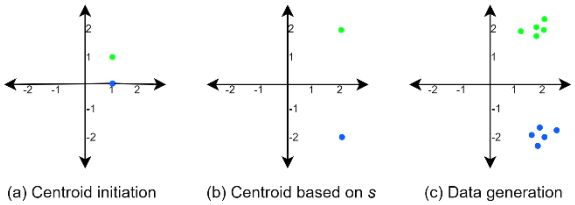

Figure 1. Illustration of Data Generations Process

We calculated the relative test accuracy for each scenario to check the possibility of a random forest producing an underfit model on the simulation data. The relative test accuracy compares random forest accuracy at a specific *nodesize* with random forest accuracy at the default *nodesize* of 1 [15]. If the relative accuracy is less than one, then the random forest is more likely to produce an underfit model on the data, and DRF can produce a better model.

Figure 2 shows this study's general flow chart of data analysis procedures. The analysis procedure for simulated data will begin by generating a dataset consisting of three data scenarios. The generation of simulation data will be repeated 100 times to ensure that the results obtained are not due to sheer coincidence.

We build Catboost and DRF models for each scenario. These models will be used to make predictions based on test data. Then the evaluation of the model is calculated based on predetermined metrics, which are balanced accuracy score, false positive rate, and imbalance accuracy metrics. Finally, we average the metrics for each scenario and each model as an evaluation method of the two models.

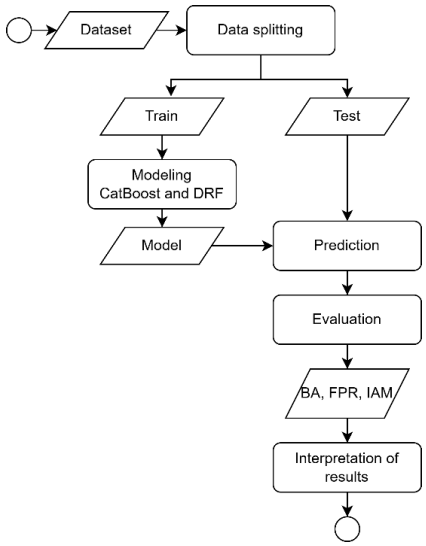

Figure 2. Illustration of Analysis Procedure

2.2 Empirical Study

The empirical data used is the industrial classification code for Indonesian establishment, KBLI, sourced from the 2016 economic census (SE2016) listing results. KBLI is a classification code for all economic activities into several business fields issued by the BPS – Statistics of Indonesia. The 2015 KBLI code is arranged hierarchically and consists of 21 alphabetical categories containing five-digit numerical groups.

The KBLI code of a company can be determined based on the main activity and product descriptions. Due to many existing KBLI codes, sufficient knowledge is required to carry out industrial classification correctly.

Table 2. Example of Empirical Data

Company ID	Main activities	Main products	KBLI
1	Menyediakan jasa catering (*Providing catering services*)	Makanan dan minuman (*Food and beverages*)	56120
2	Warung nasi aneka masakan sunda (*Selling various Sundanese dishes*)	Makanan dan minuman (*Food and beverages*)	56102

Table 2 shows an example of determining the KBLI code based on a description of the company's main activities and products. The first company whose main activity is "*providing catering services*," with the main product being "*food and beverages*," can be classified as the five-digit group code 56210 (*Catering services for specific events* (*catering services*)). The second company, with the main activity of "*Selling various Sundanese dishes*" and the main product also "*Food and beverages,"* can be grouped into KBLI 56102 (*food stores (warung makan)*).

DOI: https://doi.org/10.29207/resti.v7i1.4766

Table 3. Variables in the Study

Variables	Description	Data type
X1	Company's main activities	Text
X2	Company's main products	Text
Y	KBLI groups	Category

Details of the SE2016 listing results (L2 questionnaire) variables used in this study are the company's main activities, main products, and five-digit KBLI, as written in Table 3. The company's main activities and products are text data. This research is focused on modeling the five-digit KBLI code from the "I" category (*accommodation and food service activities*).

Before modeling, we will manually check the data to ensure that the main activities and products' descriptions match the KBLI group code. Only the match data will be used for modeling. The total empirical data chosen are as many as 3000 companies/ businesses.

Table 4 contains the distribution of the data we use in each class. The description column also briefly describes the main activities covered in each code. In the empirical data, models will predict 26 classes of KBLI groups. The distribution of the number of data in each category in Table 4 shows an imbalanced class. There is a class with the majority data, namely KBLI 56102, and a minority class, namely KBLI 56109. The empirical data imbalance ratio is 78. So, in empirical data for every example in class 56109, there are 78 examples in class 56102.

Table 4. Total Number of Data per Class

KBLI (class)	Description	Number of Data
55111	Five-star hotel	49
55112	Four-star hotel	64
55113	Three-star hotel	108
55114	Two-star hotel	63
55115	One-star hotel	53
55120	Losmen	282
55130	Cottage	155
55191	Youth hostel	28
55192	Campground	22
55194	Villa	344
55195	Apartment hotel	54
55199	Other short-term accommodation	92
55900	Other accommodation	128
56101	Restaurant	169
56102	Food stores	391
56103	Food stalls	109
56104	Mobile food service activities	105
56109	Restaurant and other mobile food	5
56210	Event catering	117
56290	Other food preparation establishment	66
56301	BAR	96
56302	Night's club	104
56303	Café	123
56304	Beverage stalls	108
56305	Traditional beverage service activities	95
56306	Mobile beverage service activities	167

The stages of analysis of empirical data, in general, will follow the flow chart in Figure 2. However, a pre-processing and feature extraction stage is needed before the model can process text data. Feature extraction converts text data into numeric data, which is carried out using TF-IDF (term frequency-inverse document frequency). TF-IDF can be formulated as *TF (t,d)* × *IDF (t)*. Whereas *TF (t,d)* specifies the number of occurrences of the word *t* in document *d*. Meanwhile, IDF can be calculated using formula 2.

$$IDF(t) = \log \frac{1+n}{1+df(t)} + 1 \quad (2)$$

Where *n* is the total number of documents in the corpus and $df(t)$ is the total number of documents in the corpus that contain the word *t*.

2.3. Classification Model with CatBoost

In the multi-class classification, only *plain* boosting mode on CatBoost can be run using the GPU. This mode works like a standard gradient boosting decision tree, only differs in processing category features directly using ordered target statistics encoding and constructing a tree using a symmetric tree as a default. Ensemble trees in CatBoost are formed sequentially.

Figure 3 illustrates the stages of modeling on CatBoost in plain mode. The prediction of the tree at the *m* stage depends on the result from the tree at the *m-1* stage. The tree created in CatBoost is a regression tree, so it is necessary to convert it into probability value so that the input data *x* can be predicted as class *C*.

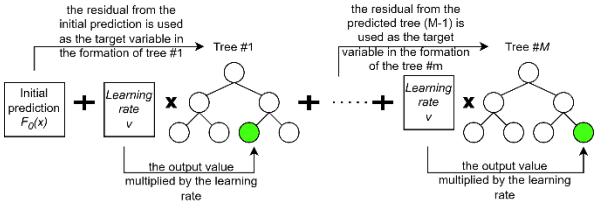

Figure 3. CatBoost Illustration

$$\frac{\sum_{i=1}^{N} w_i \log\left(e^{a_i t_i} / \sum_{j=0}^{M-1} e^{a_{ij}}\right)}{\sum_{i=1}^{N} w_i}, t \in \{0, \ldots, M-1\} \quad (3)$$

CatBoost uses the loss function in formula (3) for multi-class classification problems. Each iteration of boosting on CatBoost forms a tree. At each final node in the newly created tree, CatBoost stores *M* predictions for each *M* class.

Where t_i is the label value in the i-th data from the training data input. a_i is the result of applying the model to the i-th data. N is total data. w is the weight for the ith data and, by default, is 1. In multi-class classification problems, the conversion into opportunity values is carried out using the sigmoid function based on the "*RawFormula*" value of the CatBoost output. Furthermore, the data will be classified into the class with the highest probability value.

DOI: https://doi.org/10.29207/resti.v7i1.4766

2.4 Classification Model with DRF

The classification model using DRF combines a single decision tree in parallel according to the number of *ntree* in the model. In DRF, the tree will be formed using all training data so that all trees in DRF are created using the same data from the start [15]. The illustration in Figure 4 provides an overview of predictions using DRF, where each tree will predict the class of an input data *x*.

The final prediction is made by simple aggregation using the majority votes from the *B* trees formed as in formula 4. $C_B(x)$ in the final prediction of a class based on the independent variable *x*. *b* is the index of the tree formed. *j* is the response variable, the class to be predicted. $I(C_B(x)=j)$ is an indicator function with a value of 1 if the tree produces prediction of class *j* and a value of 0 otherwise.

$$C_B(x) = argmax_j \sum_{b=1}^{B} I(C_b(x) = j) \quad (4)$$

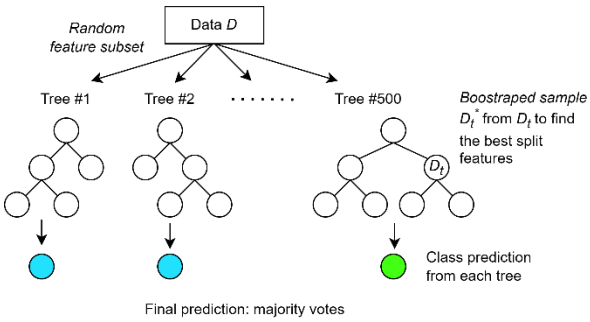

Figure 4. DRF Illustration

2.5 Evaluation Metric

An appropriate measure is needed to evaluate the performance of a multi-class classification model, especially if there is an imbalance in the data. Accuracy is a measure that is often used in classification problems, but accuracy is biased against the majority class, so it is not appropriate to use it on imbalanced data. Balanced accuracy that uses a weight so that each class is treated equally can be used [16]. Balanced accuracy works best when each class has the same importance. In essence, the balanced accuracy score is the average of the recall in each class [17]. Table 5 illustrates the prediction and actual data layout in multi-class classification. Recall value can be defined as the model's ability to correctly predict class *k* compared to all data in class *k*. Calculating the recall value can be seen in formula 5 and calculating balanced accuracy in formula 6.

This study will also calculate the false positive rate (FPR). The FPR states is the proportion of negative clases incorrectly identified as positive clases in the data. As an illustration of how to calculate FPR in multi-class classification, in Table 5, if we want to calculate the FPR of class 2, then the false positive (FP) dan false negative (FN) values are under the grey colour in the table. In contrast, the other column in the table is classified as true negative (TN). Formula 7 can be used for calculating FPR per class. In multi-class classification, the FPR value will be calculated for each class and then averaged in formula 8.

We will also calculate imbalanced accuracy metrics (IAM) [18] for model evaluation. IAM is a metric that primarily designs for multi-class imbalance datasets. IAM shows how well a classifier is expected not to classify a random instance in the incorrect classes. A higher value of IAM indicates better performance of the classifier in not mislabeling the instance. Calculations on this metric use the maximum value of the total diagonal other than the main diagonal $\left(\sum_{j\neq i}^{k} c_{ij}\right)$ or $\left(\sum_{j\neq i}^{k} c_{ji}\right)$ subtracted from the main diagonal value, divided by the total maximum from a row or column $(\max(c_{i.}, c_{.i}))$ and averaged (/k) to obtain the expectation. The calculation is presented in formula 9. The value of IAM is in the range of -1 to 1. IAM <0 indicates the average of incorrect prediction is higher than the number the classifier predicts correctly. IAM > 0 suggests that on average classifier has a higher correct prediction than the number it does not.

Table 5. Actual and Prediction per Class Ilustration

Actual	Prediction			
	Class 1	Class 2	…	Class k
Class 1	C_{11} (TN)	C_{12} (FP)	… (TN)	C_{1k} (TN)
Class 2	C_{21} (FN)	C_{22} (TP)	… (FN)	C_{2k} (FN)
….	… (TN)	… (FP)	(TN)	… (TN)
Class k	C_{k1} (TN)	C_{k2} (FP)	… (TN)	C_{kk} (TN)

$$\text{Recall}_k = \frac{c_{kk}}{\text{total observations in class k}} \quad (5)$$

$$\text{BA} = {}^{1}/_{k} \sum \text{Recall}_k \quad (6)$$

$$\text{FPR}_k = \frac{FP}{FP+TN} \quad (7)$$

$$FPR = {}^{1}/_{k} \sum \text{FPR}_k \quad (8)$$

$$\text{IAM} = 1/k \sum_{i=1}^{k} \frac{c_{ii} - \max\left(\sum_{j\neq i}^{k} c_{ij}, \sum_{j\neq i}^{k} c_{ji}\right)}{\max(c_{.i}, c_{i.})} \quad (9)$$

3. Results and Discussions

3.1 Simulation Study

In simulation and empirical studies, most programs are made using python with the *sci-kit learn* package [19]. For the model with CatBoost, we use the CatBoost package in python. For DRF modeling, we use DRF formulas which can be accessed on the author's GitHub [15] using the R program.

The data exploration stage is carried out first to ensure that the scenario generated on the simulation data is appropriate. Feature reduction is needed to explore data visually on data with high dimension. The multidimensional scaling (MDS) method will be used

DOI: https://doi.org/10.29207/resti.v7i1.4766

to reduce the features of the simulated data with 300 dimensions to three dimensions so that the data distribution in each class can be seen using a three-dimensional plot. In general, MDS is a technique used to analyze data similarity using distances in geometric space.

Figure 5 shows the results of data exploration using MDS. Different colours in the plot represent data from different classes. The results of data exploration in Figure 5(a) illustrate that the data with high centroid distances between classes are separated more clearly than when the distance is halved, as in Figure 5(b), and the distance is reduced by one-quarter, as in Figure 5(c). Data exploration shows that as a centroid distance between classes gets closer to each other, the separation between classes is less visible, as we can see in the three-dimensional plot. The data between classes piling up as the distance between classes gets closer. Hence, the classification becomes more difficult.

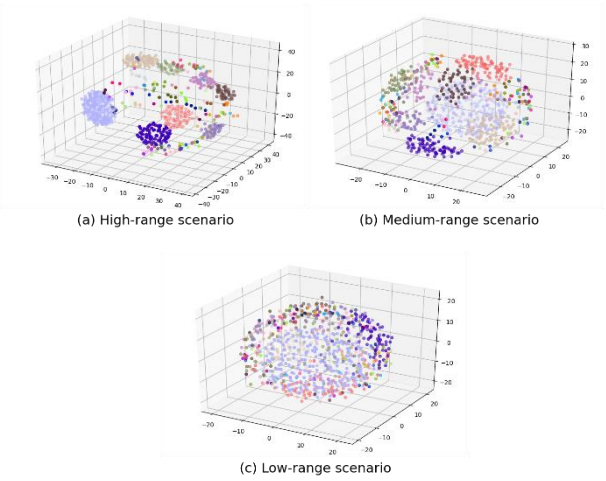

Figure 5. Exploration of Simulated Data with MDS

The relative accuracy test was calculated for the three scenarios. For high distances, the relative accuracy for *nodesize* 0.06*n* to 0.01*n* is 1. *n* is the number of train data used. The results show a value smaller than one for each *nodesize* setting at medium and low distances, as shown in Figure 6. From these results, the model produced by RF on the simulation data medium and low distance has the possibility of underfitting. Under these conditions, DRF can produce better accuracy than RF [15].

Next, we constructed CatBoost and DRF models for each scenario: high, medium, and low distances. So that three CatBoost models and three DRF models will be formed, representing each scenario. Then, we compared the model performance for each scenario based on the balanced accuracy, false positive rate, and IAM values.

Table 6 shows the study results in each scenario and method. In the high-distance scenario, both models can make perfect predictions marked by a balanced accuracy value of 100% even though there is an imbalance in the data. This result is in line with our hypothesis, which, as the data is perfectly separated, even with a simple classification model, one could perform a perfect classification. Whereas in the medium-distance and low-distance scenarios, the average balanced accuracy of both models decreases.

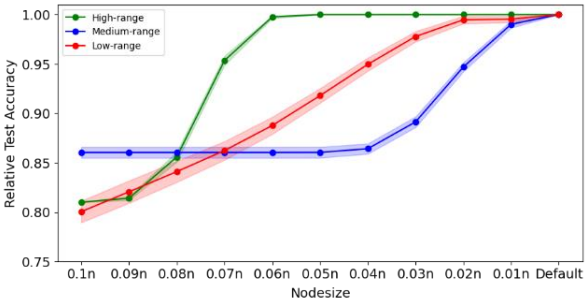

Figure 6. Relative Accuracy of Simulation Data

In medium-distance scenarios, CatBoost produces a higher balanced accuracy than DRF. CatBoost's average balanced accuracy is 99.25%, while DRF's is 97.54%. The false positive rate shows that DRF produces higher with 0.039% while CatBoost is 0.011%. The imbalance accuracy ratio (IAM) on both methods shows a positive value, which means, on average, both methods can make a higher correct prediction than the number it does not. CatBoost has a higher IAM value than DRF, which is 0.0348 in difference.

Table 6. Average of Balanced Accuray, FPR, dan IAM

Scenario	BA (%)	FPR (%)	IAM
High distance			
CatBoost	100.00	0.000	1.000
DRF	100.00	0.000	1.000
Medium distance			
CatBoost	99.25	0.011	0.984
DRF	97.54	0.039	0.949
Low distance			
CatBoost	32.37	0.869	-0.444
DRF	23.97	1.434	-0.560

In the low-distance scenario, the performance of both models drops significantly compared to high and medium distances. CatBoost produces a balanced accuracy of 32.37% and 23.977% with DRF. As the distance between classes getting close, the separation between classes becomes less visible; thus, classification is getting harder. The FPR value for both methods is also higher for DRF, which is 1.434%, while CatBoost is 0.869%. The IAM value for both methods shows a negative value indicating that the number of instances the classifier predicts incorrectly on average is higher than it predicts correctly, which means poor performance.

Hypothesis testing was conducted in medium and low-distance scenarios with our null hypothesis that both methods produce the same performance based on all metrics. The alternative hypothesis is that CatBoost's performance is different from DRF's. The p-value on

DOI: https://doi.org/10.29207/resti.v7i1.4766

the t-test results in each scenario with each evaluation metric produces a value of 0.000 at α= 0.05. Based on these test results, there is sufficient evidence to state that the CatBoost method performs better than DRF in medium and low-distance multi-class simulation scenarios. In addition, simulation studies also show that the distance factor affects classification ability. Even though the data is imbalanced, classification can be done well when the data between classes have a high-distance

3.2 Empirical Study

Text data are used for empirical study. Therefore, several steps are carried out before modeling. In Figure 7, we show the step we use. We will first do cleaning, pre-processing and feature extraction based on the data. The result from feature extraction could later be used in the modeling phase. The cleaning stage is carried out to ensure that the data used does not contain noise.

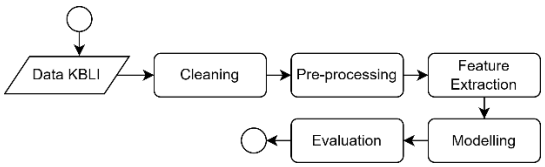

Figure 7. Text Data Analysis Procedure

The preprocessing stage consists of (1) combining details from X1 (main activity description) and X2 (main product description) for input in the model by concatenating. For example, X1: "PENYEDIAAN MINUMAN KELILING", and X2: "POPICE", the new details used as input are "*PENYEDIAAN MINUMAN KELILING POPICE*". These results will enter stage (2) *lowercasing*, changing capital letters to non-capital. In this step, the input becomes: "*sedia minum keliling popice*". Then go to stage (3), which is *stopword removal* by deleting certain words that appear frequently but do not give significant meanings. In Indonesian, those words such as "dan" or "di". The last stage is (4) *stemming* to return the words to their primary form using the *Sastrawi* library in python. This process is done by removing affixes in the Indonesian language. The final input after the preprocessing stage becomes "*sedia minum keliling popice*".

After the preprocessing stage is complete, feature extraction will be done using TF-IDF to transform text data into numeric so that the analysis and modeling stages can be carried out. In empirical data, the relative accuracy test was also calculated and showed a relative accuracy < 1 for each *nodesize* setting, as in Figure 8.

Determining hyperparameters is essential in building a machine-learning model [20]. In this study, hyperparameter tuning will be carried out using a *grid search* method. Using grid search, we will pre-determined lists of hyperparameters to search for their best combinations.

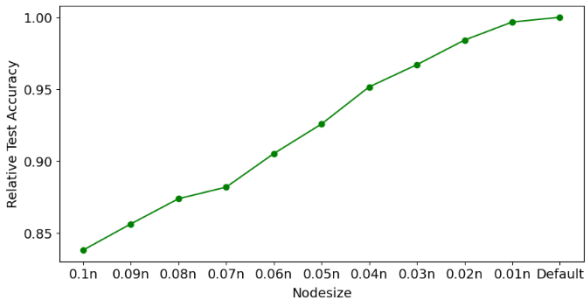

Figure 8. Relative Accuracy of Empirical Data

For CatBoost hyperparameter tuning, we follow [21] without including the *auto_class_weight* parameter. Table 7 shows the hyperparameter and list value used on the grid search. In CatBoost, learning rates use internal adjustments, so the difference between tuned and not tuned is similar [22]. This hyperparameter tuning results are *iterations*: 1000, *depth*: 8, and *l2_leaf_reg*: 1.

Table 7. CatBoost Hyperparameter Tuning Value

Parameter name	Values	Description
depth	4, 5, 6, 7, 8	Control maximum depth of decision tree (6)
l2_leaf_reg	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	Coefficient of regularization term (3)
n_estimators	100, 250, 500, 1000	Number of trees in the ensemble (1000)

We also use hyperparameter tuning for DRF. Based on [15], Han et al. only use *mtry* for tuning DRF. In this study, we will try several parameters that have been tried on the RF, which shows the influence on the model [20]. The hyperparameters tested on DRF are presented in Table 8. The result of this hyperparameter tuning are *mtry*: $2\sqrt{p}$, *nodesize*: 2, and *sampsize*: 0.4*n*.

Table 8. DRF hyperparameter tuning value

Parameter name	Values	Description
mtry	$\sqrt{p}$, $2\sqrt{p}$	Number of variables that randomly sample for candidate split ($\sqrt{p}$)
nodesize	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	Minimum size of terminal nodes (1)
sampsize	0.2n, 0.4n, 0.632n, 0.7n, 0.9n	The number of observations that are drawn for each tree (0.632n)

Model evaluation will be carried out using repeated stratified k-fold. Table 9 shows an average of evaluation metrics on each model. We use the best model from hyperparameter tuning for evaluation. Any other parameter that was not stated will be set by default.

In the empirical study, the average balanced accuracy on CatBoost is higher than DRF, as presented in Table 9. Balanced accuracy on CatBoost on test data is 92.45%, while DRF is 88.18%. The FPR value on CatBoost is lower by 0.031% than DRF. At the same

DOI: https://doi.org/10.29207/resti.v7i1.4766

time, the positive IAM values for both methods indicate the ability to carry out more correct classifications, with CatBoost having a higher IAM value. Based on these three measures, CatBoost and DRF were able to classify the KBLI group, with CatBoost outperforming DRF.

Table 9. Average of Balanced Accuracy, FPR and IAM

Method	BA (%)	FPR (%)	IAM
CatBoost			
Data train	98.25	0.031	0.954
Data test	92.45	0.194	0.807
DRF			
Data train	98,90	0,025	0.969
Data test	88.18	0,225	0.717

A statistical test was then carried out to test the hypothesis that the difference in the value of the metrics was significant. We use the same test procedure as in the simulation data. The t-test yields a p-value = 0.000 at $\alpha = 0.05$, so there is sufficient evidence to state that CatBoost's performance is significantly better than DRF's based on the metrics.

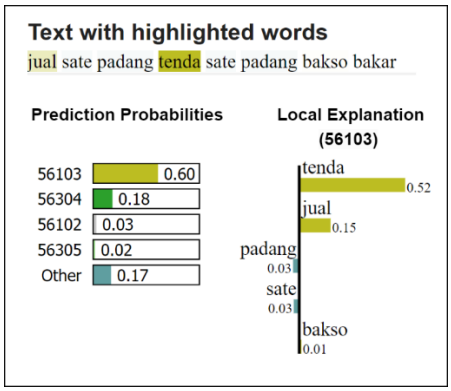

Figure 9. LIME Results on CatBoost Predictions

We also conducted further interpretation from the result of CatBoost using the local model-agnostic method. A local model-agnostic method is a local interpretation method that explains prediction from individuals [23]. One of the implementations of local model-agnostic is using LIME. LIME is a method that can be used in individual interpretation of tabular data, text classification, and image classification. LIME can describe sparse data (lots of zero-value columns) using fewer variables [24]. Text data from basic vectorization such as TF-IDF will produce sparse data to apply this method. LIME [25] works by training local surrogate models to explain predictions from individual data.

Interpretation or explanation of the CatBoost model on individual data carried out using sample data. In Figure 9, the company with the input description "selling satay padang, satay padang, grilled meatballs" ("*jual sate padang tenda sate padang baksor bakar*") can be predicted correctly at KBLI 56103 (diner (*kedai makan)*). The prediction probability shows the predicted probability for every class. The results of the local explanation show that "tent" (*tenda*) and "sell" (*jual*) features contribute the most to this prediction.

4. Conclusion

The simulation study results show that the distance factor between classes affects the predictive ability of the two models—the farther the distance between classes, the better the model performance. CatBoost and DRF can produce perfect predictions in high-range scenarios when the classes are adequately separated, even though there is imbalanced data. Both methods can be said to have the same good performance in high-distance scenarios. However, the closer the distance between classes, the performance of both models decreases. Although the relative accuracy test shows that the DRF model can produce better accuracy than RF, CatBoost's balanced accuracy is significantly better than DRF in medium and low-distance scenarios. The application of CatBoost and DRF to the empirical data from the KBLI text shows good performance, with CatBoost's balanced accuracy on the test data of 92.45% and DRF of 88.18%. CatBoost's performance on empirical data significantly surpasses DRF. Furthermore, the use of CatBoost and DRF in other datasets could be studied with more extensive hyperparameter tuning

Reference

[1] M. Koziarski, M. Woźniak, and B. Krawczyk, "Combined Cleaning and Resampling algorithm for multi-class imbalanced data with label noise," *Knowledge-Based Syst.*, vol. 204, p. 106223, 2020, doi: https://doi.org/10.1016/j.knosys.2020.106223.

[2] H.-Y. Lin, "Efficient classifiers for multi-class classification problems," *Decis. Support Syst.*, vol. 53, no. 3, pp. 473–481, 2012, doi: https://doi.org/10.1016/j.dss.2012.02.014.

[3] R. Chairunisa, Adiwijaya, and W. Astuti, "Perbandingan CART dan Random Forest untuk Deteksi Kanker berbasis Klasifikasi Data Microarray," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 5 SE-Artikel Rekayasa Sistem Informasi, Oct. 2020, doi: 10.29207/resti.v4i5.2083.

[4] I. T. Utami, B. Sartono, and K. Sadik, "Comparison of single and ensemble classifiers of support vector machine and classification tree," *J. Math. Sci. Appl.*, vol. 2, no. 2, pp. 17–20, 2014.

[5] B. Sartono and U. D. Syafitri, "Metode pohon gabungan: Solusi pilihan untuk mengatasi kelemahan pohon regresi dan klasifikasi tunggal," in *Forum Statistika dan Komputasi*, 2010, vol. 15, no. 1.

[6] J. Treboux, D. Genoud, and R. Ingold, "Decision tree ensemble vs. nn deep learning: efficiency comparison for a small image dataset," in *2018 International Workshop on Big Data and Information Security (IWBIS)*, 2018, pp. 25–30.

[7] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. [Online]. Available: https://faculty.marshall.usc.edu/gareth-james/ISL/

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009. [Online]. Available: http://www-stat.stanford.edu/~tibs/ElemStatLearn/

[9] T. Chen *et al.*, "Xgboost: extreme gradient boosting," *R Packag. version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.

[10] G. Ke *et al.*, "Lightgbm: A highly efficient gradient boosting

DOI: https://doi.org/10.29207/resti.v7i1.4766

decision tree," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
[11] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: gradient boosting with categorical features support." arXiv, 2018. doi: 10.48550/ARXIV.1810.11363.
[12] J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, and M. Asadpour, "Boosting methods for multi-class imbalanced data classification: an experimental review," *J. Big Data*, vol. 7, no. 1, p. 70, 2020, doi: 10.1186/s40537-020-00349-y.
[13] X. Zhang and G. X. Wu, "Text classification method of dongba classics based on CatBoost algorithm," in *The 8th International Symposium on Test Automation & Instrumentation (ISTAI 2020)*, 2020, vol. 2020, pp. 133–139. doi: 10.1049/icp.2021.1336.
[14] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[15] S. Han, H. Kim, and Y.-S. Lee, "Double random forest," *Mach. Learn.*, vol. 109, no. 8, pp. 1569–1586, 2020, doi: 10.1007/s10994-020-05889-1.
[16] M. Gösgens, A. Zhiyanov, A. Tikhonov, and L. Prokhorenkova, "Good Classification Measures and How to Find Them." arXiv, 2022. doi: 10.48550/ARXIV.2201.09044.
[17] M. Grandini, E. Bagli, and G. Visani, "Metrics for Multi-Class Classification: an Overview." arXiv, 2020. doi: 10.48550/ARXIV.2008.05756.
[18] E. Mortaz, "Imbalance accuracy metric for model selection in multi-class imbalance classification problems," *Knowledge-Based Syst.*, vol. 210, p. 106490, 2020, doi: https://doi.org/10.1016/j.knosys.2020.106490.
[19] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in {P}ython," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[20] P. Probst, B. Bischl, and A.-L. Boulesteix, "Tunability: Importance of Hyperparameters of Machine Learning Algorithms." arXiv, 2018. doi: 10.48550/ARXIV.1802.09596.
[21] J. Hancock and T. M. Khoshgoftaar, "Impact of Hyperparameter Tuning in Classifying Highly Imbalanced Big Data," in *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*, 2021, pp. 348–354. doi: 10.1109/IRI51335.2021.00054.
[22] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artif. Intell. Rev.*, vol. 54, no. 3, pp. 1937–1967, 2021, doi: 10.1007/s10462-020-09896-5.
[23] C. Molnar, *Interpretable Machine Learning*, 2nd ed. 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book
[24] P. Biecek and T. Burzykowski, *Explanatory Model Analysis*. Chapman and Hall/CRC, New York, 2021. [Online]. Available: https://pbiecek.github.io/ema/
[25] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144. doi: 10.1145/2939672.2939778.

DOI: https://doi.org/10.29207/resti.v7i1.4766