applied sciences

MDPI

Article

Predicting the Risk of Chronic Kidney Disease (CKD) Using Machine Learning Algorithm

Weilun Wang [1,*], Goutam Chakraborty [1,2] and Basabi Chakraborty [1]

[1] Faculty of Software Information Science, Iwate Prefectural University, Iwate 020-0693, Japan; goutam@iwate-pu.ac.jp (G.C.); basabi@iwate-pu.ac.jp (B.C.)
[2] Sendai Foundation for Applied Information Sciences, Sendai 980-0012, Japan
[*] Correspondence: g236q201@s.iwate-pu.ac.jp; Tel.: +81-070-4295-0361

Abstract: Background: Creatinine is a type of metabolite of blood that is strongly correlated to glomerular filtration rate (GFR). As measuring GFR is difficult, creatinine value is used for indirectly determining GFR and then the stage of chronic kidney disease (CKD). Adding a creatinine test into routine health examination could detect CKD. As more items for comprehensive examination means higher cost, creatinine testing is not included in the routine health examination in many countries. An algorithm based on common test results, without creatinine test, to evaluate the risk of CKD will increase the chance of its early detection and treatment. **Methods:** In this study, we used open source data containing 1 million samples. These data contain 23 health-related features, including common diagnostic test results provided by National Health Insurance Sharing Service (NHISS). A low GFR indicates possible chronic kidney disease (CKD). As is commonly accepted in the medical community, a GFR of 60 mL/min is used as the threshold, below which is considered to have CKD. In this study, the first step aims to build a regression model to predict the value of creatinine from 23 features, and then combine the predicted value of creatinine with the original 23 features to evaluate the risk of CKD. We will show by simulation that by the proposed method we can achieve better prediction results compared to direct prediction from 23 features. The data is extremely unbalanced for predicting the target variable creatinine. We used undersampling method and proposed a new cost-sensitive mean-squared error (MSE) loss function to deal with the problem. Regrading model selection, this work used three machine learning models: a bagging tree model named Random Forest, a boosting tree model named XGBoost, and a neural network based model named ResNet. To improve the result of the creatinine predictor, we averaged results from eight predictors, a method known as ensemble learning. Finally, the predicted creatinine and the original 23 features is used to predict the risk of CKD. **Results:** We optimized results of R-Squared (R2) value to select the appropriate undersampling strategy and the regression model for the regression stage of creatinine prediction. Ensembled model achieved the best performance of R2 of 0.5590. The six factors from 23 are selected from the top of the list of how strongly they affect the creatinine value. They are sex, age, hemoglobin, the level of urine protein, waist circumference, and habit of smoking. Using the predicted value of creatinine, an area under Receiver Operating Characteristic curve (AUC) of 0.76 is achieved while classifying samples for CKD. **Conclusions:** Using commonly available health parameters, the proposed system can assess the risk of CKD for public health. High-risk subjects can be screened and advised to take a creatinine test for further confirmation. In this way, we can reduce the impact of CKD on public health and facilitate early detection for many, where a blanket test of creatinine is not available for all.

Keywords: chronic kidney disease; creatine; ensemble learning; regression; unbalanced data

check for updates

Citation: Wang, W.; Chakraborty, G.; Chakraborty, B. Predicting the Risk of Chronic Kidney Disease (CKD) using Machine Learning Algorithm. *Appl. Sci.* **2021**, *11*, 202. https://dx.doi.org/10.3390/app11010202

Received: 11 November 2020
Accepted: 23 December 2020
Published: 28 December 2020

1. Introduction

Chronic kidney disease (CKD) is a type of kidney disease in which there is a gradual loss of glomerular filtration rate (GFR) over a period of more than 3 months [1]. It is a silent killer as there are no physical symptoms in the early stage. CKD affected 753 million people globally in 2016, 417 million females and 336 million males [2]. Over 1 million people in 112 poor countries die from renal failure every year, as they cannot afford the huge financial burden of regular dialysis or kidney replacement surgery [3]. Thus, early detection and effective intervention are important to reduce the impact of CKD on public health. Due to different economic conditions in different countries, the schedule for routine health examinations is different. Even in the same country, different groups get different levels of health examinations. A comprehensive routine health examination even for detection of common fatal diseases, like cancer and heart disease, is rare in most countries. Tests related to CKD are initiated only when there is a symptomatic problem, and then it is too late.

For screening of kidney function, a urine test and a blood test are needed [4]. Creatinine is a type of metabolite in blood, which reflects the value of glomerular filtration rate (GFR) indirectly. Direct measurement of GFR is difficult. GFR is estimated by a simple function whose parameters are creatinine value, sex, age, and race. Disease control agencies of some countries recommend that the whole population over a certain age should be screened for creatinine. Meanwhile, in many countries people with diabetes or hypertension (high blood pressure) are been screened for regular renal check [5]. Prediction of CKD through ultrasound imaging is also considered desirable in clinical practice [6].

Recently, researches on the prediction of CKD using machining learning methods were reported [7–13]. All of these works used a dataset from University of California Irvine (UCI) [14] which contains 400 samples with 24 features (age, blood pressure, creatinine, etc.) to measure CKD, and it achieved good classification results with over 97% accuracy. Although the result looks good, it cannot be applied to practice. The first problem is that there is a bias in the UCI dataset. There are 250 CKD samples and 150 non-CKD samples in the dataset: the ratio of CKD and non-CKD is different from reality. In addition, in the 250 CKD samples, there are nearly 140 samples with creatinine values exceeding 10 mL/min, which are meaningless to classify CKD. On this account, the proposed model will fail to classify, and the classification result will not be acceptable when we consider actual data. How the composition of CKD samples and non-CKD samples affects the classification results will be explained in Section 4.6. The second problem is that the ground truth of CKD is determined by the value of GFR, and the value of GFR is calculated by Equation (1). In Equation (1), the value of creatinine is the main contributing parameter with three other features: age, race, and sex. In other words, if the value of creatinine is already known, the value of GFR can be calculated directly using Equation (1) appears in Section 2.1, and we know the status of CKD from the value of GFR. Therefore, using machining learning algorithms to predict the result of CKD on condition that value of creatinine is already known and the ground truth is calculated using Equation (1), is meaningless. Thus, the premise of the previous published works [7–13] is flawed.

In addition, we found that there is another way to measure CKD as described in the work [5]. Features of age, gender, the existence of diabetes, hypertension, anemia, and cardiovascular disease are used to measure the risk score of CKD using a simple grid search method. From this work, we got a cue that the state of other common diseases could possibly be used to measure the risk of CKD where the parameters do not include creatinine. The existing method treats the existence of related diseases, like diabetes or hypertension, as binary variables, 0/1, to predict CKD risk. Details about diagnostic tests like blood sugar, blood pressure, hemoglobin are usually included in items of a regular health check. Other common physical measures (waist, BMI, vision, etc.) may possibly be related to the risk of CKD. Therefore, we got an idea to try whether we can predict the risk of CKD using these possibly related and commonly available data.

In this work, we proposed a two-stage method to evaluate the risk of CKD under the condition that creatinine data is not available. In the first stage, a machine learning regression model is used to predict the value of creatinine using a supervised data. As the values of creatinine in the data, which is the target variable, are extremely unbalanced, we used an undersampling method and proposed a cost-sensitive mean squared error

(MSE) loss function to deal with the problem. With respect to model selection, in this work we used three machine learning models: a bagging tree model known as Random Forest [15], a boosting tree model called XGBoost [16], a neural network based model known as ResNet [17]. To improve the result of creatinine prediction, we averaged results from eight predictors as our ensembling learning. Finally, the predicted creatinine and the original 23 features are used to predict the risk of CKD in a binary way.

The paper is organized as follows. In Section 2, we describe the dataset and elaborate highlights of the experiment. Section 3 describes the proposed method. The experimental details and results are discussed in Section 4. The paper is concluded in Section 5 with some ideas about the future direction of the work.

2. Materials and Methods

2.1. Dataset

In this work, we used the open source data provided by National Health Insurance Sharing Service (NHISS), available from the site: https://nhiss.nhis.or.kr/. The dataset contains the regular heath check information for 1 million subjects between 25 and 90 years of age, data collected in 2017. There are 24 collected attributes; we added another three attributes, namely, GFR, Stage, and CKD, and their description is listed in Table 1.

Table 1. Description of variables used in the analysis.

Ser	Variable	Type	Class	Description
1	Sex	Categorical	Predictor	
2	Age	Numerical	Predictor	
3	Waist	Numerical	Predictor	
4	Listen_left	Categorical	Predictor	hearing impairment or not
5	Listen_right	Categorical	Predictor	hearing impairment or not
6	Vision_left	Numerical	Predictor	
7	Vision_right	Numerical	Predictor	
8	BP_HIGH	Numerical	Predictor	systolic pressure
9	BP_LWST	Numerical	Predictor	diastolic pressure
10	BLDS	Numerical	Predictor	fasting blood sugar
11	TOT_CHOLE	Numerical	Predictor	total cholesterol
12	TRIGLYCERIDE	Numerical	Predictor	triglycerides
13	HDL_CHOLE	Numerical	Predictor	high-density lipoprotein cholesterol
14	LDL_CHOLE	Numerical	Predictor	low-density lipoprotein cholesterol
15	HMG	Numerical	Predictor	hemoglobin
16	OLIG_PROTE_CD	Categorical	Predictor	the level of urine protein
17	SGOT_AST	Numerical	Predictor	aspartate amino-transferase
18	SGPT_ALT	Numerical	Predictor	alanine amino-transferase
19	GAMMA_GTP	Numerical	Predictor	gamma glutamyl transpeptidas
20	SMK_STATE	Categorical	Predictor	smoking status
21	DRINK_OR_NOT	Categorical	Predictor	drink habit
22	MOUTH_CHECK	Categorical	Predictor	decayed teeth or not

Table 1. *Cont.*

Ser	Variable	Type	Class	Description
23	BMI	Numerical	Predictor	calculated using height and weight
24	CREATININE	Numerical	Target_1	serum creatinine
25	GFR	Numerical		calculated using Equation (1)
26	Stage	Categorical		determined from GFR
27	CKD	Binary	Target_2	GFR < stage 2 (class 1) or not (class 0, normal)

In the first stage of the experiment, the 24th entry in Table 1 is the target variable, where 1 to 23 are input variables to the regression model. The attributes 25–27 are not contained in the original collected data. Those added attributes are derived as follows. The 25th attribute of glomerular filtration rate (GFR) is calculated using the formula shown below.

$$GFR = 186 \times [Creatinine]^{-1.154} \times [Age]^{-0.203} \times [1.212 \quad if\_Black] \times [0.742 \quad if\_Female] \tag{1}$$

GFR is divided into 6 stages from normal level to renal failure, according to flow rate shown in Table 2. This is the 26th attribute in Table 1. Stages 1 and 2 are regarded as normal non-CKD (class 0), and below that flow rate is regarded as CKD (class 1). This is the 27th entry in Table 1 and is our final classification target (Target_2).

Table 2. Stage of glomerular filtration rate (GFR) [1].

Stage	GFR (mL/min)	Description
Stage 1	90 or higher	normal
Stage 2	89 to 60	mild loss
Stage 3a	59 to 45	mild to moderate
Stage 3b	44 to 30	moderate to severe
Stage 4	29 to 15	severe
Stage 5	less than 15	failure

2.2. *Regression Methods*

2.2.1. Random Forest

Random forest (RF) is an ensemble learning method that constructs a multitude of decision trees at training time and outputting mean prediction from individual trees [15]. A basic structure of Random Forest is shown in Figure 1.

Each sub-tree model does random sampling with replacement from training data and finally average results from all sub-models. Every sub-model runs in parallel without any dependency. In addition to constructing each tree using a different subset of the data, random forests differ in the way of how trees are constructed. In standard decision trees, each node is branched using the optimum decision for division among all variables, so as to minimize entropy due to the splitting of the data set represented by the parent node. In a random forest, split points of each node are randomly chosen from the best split point among a subset of predictors. Random forest thus avoids overfitting, which is common with a single deep decision tree.

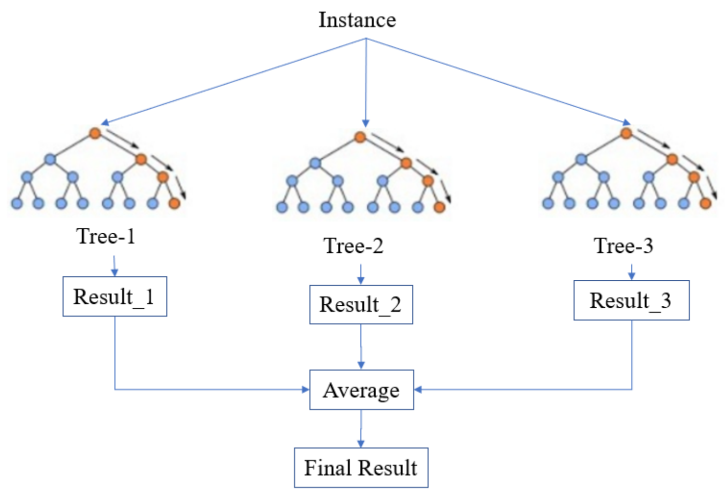

Figure 1. Simplified structure of random forest.

2.2.2. XGBoost

XGBoost is an optimized distributed gradient boosting library of algorithms [16]. It implements machine learning algorithms under the Gradient Boosting Decision Tree (GBDT) framework [18]. A basic structure of XGBoost is shown in Figure 2. It is to be noted that the residual from tree-1 is fed to tree-2 so as to reduce the residual and this continues.

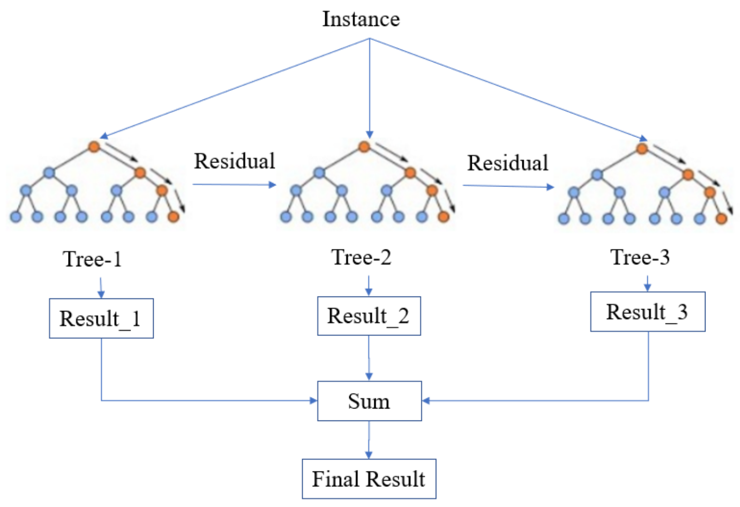

Figure 2. Simplified structure of XGBoost.

Different from Random Forest, each tree model in XGBoost minimizes the residual from its previous tree model. The traditional GBDT uses only the first derivative of error information. XGBoost performs the second-order Taylor expansion of the cost function, and uses both the first and second derivatives. In addition, the XGBoost tool supports customized cost function.

2.2.3. RestNet

ResNet is a deep residual neural network used to learn the regression of nonlinear functions [17]. It replaces convolutional layers and pooling layers by fully connected layers to ensure that deep residual learning can be achieved for nonlinear regression. The basic structure of ResNet is shown in Figure 3. The model begins with an input layer and is followed by dense blocks and identity blocks. There are three hidden dense layers in both the dense block and identity block. In the dense block, the input is also connected to output via another dense layer, whereas in the identity block it is directly connected. In ResNet, output layer is the end layer. In this work, every dense block is followed by two identity blocks, and this set of three blocks are repeated 3 times. The last identity block is followed by the output layer.

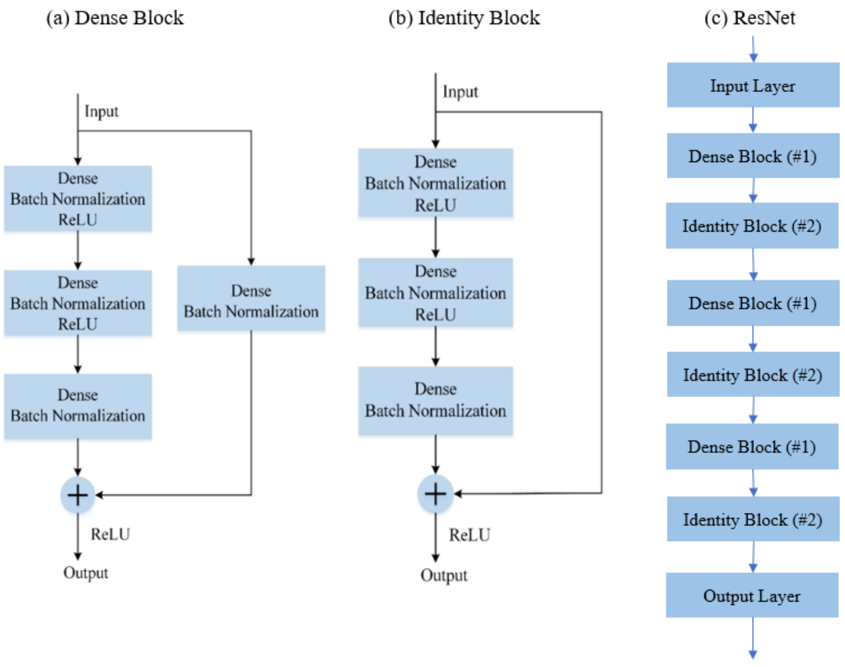

Figure 3. Simplified structure of ResNet.

3. Proposed Methodology

Figure 4 shows two methods of predicting CKD Risk. In Figure 4a, we predict CKD risk directly, and in Figure 4b, we predict creatinine first and then combine its value with 23 features to predict the risk of CKD. The complicated nonlinear prediction of creatinine in model (b), make the input to CKD classifier richer in information. Compared to model (a), model (b) can achieve better classification. In this work, we used model (b) for CDK risk prediction.

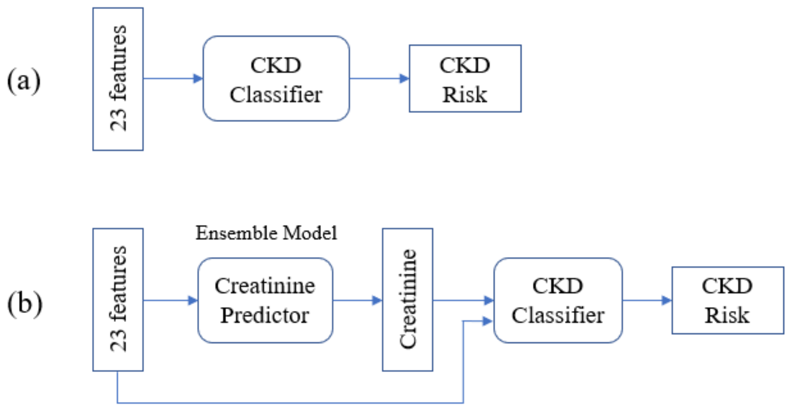

Figure 4. Two methods of predicting Chronic Kidney Disease (CKD) risk. (**a**) directly training a CKD classifier to predict CKD risk; (**b**) predict creatinine first and then combine its value with 23 features to predict the risk of CKD.

In Section 3.1, the preprocessing method of the data is described. To overcome the problem of imbalance in data, we used undersampling, which is described in Section 3.2.

We introduce a new cost-sensitive loss function in Section 3.3. Finally, the model ensemble strategy is explained in Section 3.4.

3.1. Preprocessing Method

In the preprocessing part, we did data cleaning as described in item (i) and item (ii) below. We also modified coding of some attributes as described in item (iii).

(i) Some samples have missing values on some attributes. As we already have a large data, 8654 samples with one or more missing attributes are removed.
(ii) There are some data with very large values of creatinine. Those samples are from patients at a late stage of renal failure. These subjects are not targets of this work, and therefore such data are classed as outlier data as far as training our target model is concerned. We removed 1234 samples with a value of creatinine higher than 2.5 mL/min.
(iii) In the original data, the attribute of Sex and SMK_STATE are not suitable for numerical coding. We changed them into one-hot coding format. We remove original variables and replace them with new binary variables where 0 is the value when the category is false and 1 when it is true. For example, sex is replaced by Male and Female, two attributes. For a Male subject, Male attribute is assigned a value "1" and female as "0".

After preprocessing, 990,112 samples remained. We split it into a training set with 900,000 samples and a test set with 90,112 samples.

3.2. Undersampling for Data Balancing

3.2.1. Extremely Unbalanced Data

The distribution of target variable of creatinine is shown in Figure 5. Most of the samples are concentrated near the median, and the number of samples away from the median is very less. A machine learning model will fit better on the region with more samples and perform worse around the region where data is less. In another words, the confidence interval of prediction accuracy will be wider where the samples are less.

For general regression tasks, the problem of unbalanced data can be ignored. If an interval contains less samples, we can say that the samples in that range are outliers. However, for this task, it is the opposite. Our target is to correctly predict those people who have a high creatinine value, though we have a few data for that range. From Figure 5, we observe that most samples are between 0.5 and 1.4. If we use the whole dataset to train a machine learning model, as the training algorithm will try to minimize the sum error for the whole dataset, samples with high creatinine value where data is less will be ignored.

In order to show the impact of imbalance problem on regression more explicitly, we performed a simple experiment using XGBoost with Minimum Square Error (MSE) as loss function. The result is shown in Figure 6, where the y-axis represents the ground truth and the x-axis is the predicted value. We observed that the model failed to predict for the interval with less training samples.

Generally speaking, the problem occurs because the number of samples with a high creatinine value is not enough for the machine learning model to learn from them. The data imbalance problem usually can be solved by data level methods and model level methods [19]. Oversampling is the most common data level method. However, this data is obviously not suitable for oversampling. Over-sampling refers to balance data distribution by resampling or generating new data from low number of available ones. In this problem, high creatinine data is extremely low. Oversampling will cause noise in the rare data to be amplified countless times. We used undersampling method and proposed a cost-sensitive mean-squared error (MSE) loss function to deal with this problem.

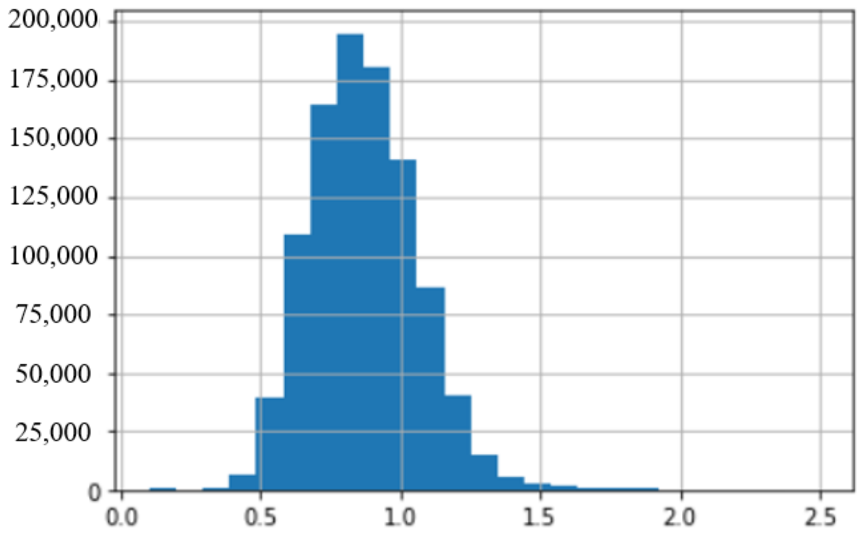

Figure 5. Distribution of Creatinine.

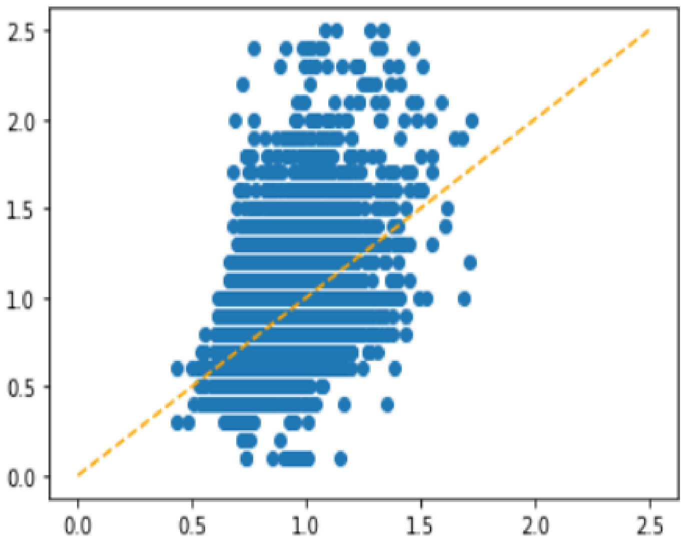

Figure 6. Impact of imbalance problem.

3.2.2. Details about Undersampling

The target of the prediction, the creatinine value, is highly imbalanced in the data set. To alleviate the imbalance problem, undersampling is done on a range of values where a large amount of data is available. Table 3 shows the details about undersampling. Six types of undersampling strategies are considered with different levels of undersampling as shown in Table 3. Training data is more balanced as we go from sampling-1 to sampling-6. Total training data, shown in the last row of Table 3, were 191,312, 107,439, 61,757, 34,220, 18,896, and 11,069 samples, respectively. As we opt for a more balanced set, the number of training data decreases. We performed experiments with all 6 sample sets and compare the results.

3.3. *Cost-Sensitive MSE Loss Function*

Mean absolute error (MAE) and mean squared error (MSE) are the two basic evaluation methods for regression tasks. Their formulas are shown below,

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_t^i - y_p^i\right| \tag{2}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_t^i - y_p^i)^2 \quad (3)$$

where y_t^i is the target value of the *ith* labeled data, and y_p^i is the predicted value of the same from the regression algorithm. Errors in MSE are squared; it makes samples with small error less important (due to squaring) and creates a stronger incentive to train data with larger error. This happens for attribute values where the available data are rare. However, it does not mean that cost function with further higher-order on errors will achieve better results. Especially for data with noise which will be amplified too. In order to strike a balance, MSE is considered as a suitable loss function for this task.

Table 3. Undersampling.

Creatinine	Original	Sampling-1	Sampling-2	Sampling-3	Sampling-4	Sampling-5	Sampling-6
0.1	395	395	395	395	395	395	395
0.2	93	93	93	93	93	93	93
0.3	549	549	549	549	549	549	549
0.4	5503	5503	5503	5000	2500	1200	600
0.5	35,395	20,000	10,000	5000	2500	1200	600
0.6	99,185	20,000	10,000	5000	2500	1200	600
0.7	149,238	20,000	10,000	5000	2500	1200	600
0.8	177,224	20,000	10,000	5000	2500	1200	600
0.9	164,153	20,000	10,000	5000	2500	1200	600
1.0	127,881	20,000	10,000	5000	2500	1200	600
1.1	78,432	20,000	10,000	5000	2500	1200	600
1.2	37,180	20,000	10,000	5000	2500	1200	600
1.3	13,873	13,873	10,000	5000	2500	1200	600
1.4	5216	5216	5216	5000	2500	1200	600
1.5	2224	2224	2224	2224	2224	1200	600
1.6	1160	1160	1160	1160	1160	1160	600
1.7	687	687	687	687	687	687	600
1.8	523	523	523	523	523	523	523
1.9	309	309	309	309	309	309	309
2.0	235	235	235	235	235	235	235
2.1	144	144	144	144	144	144	144
2.2	137	137	137	137	137	137	137
2.3	92	92	92	92	92	92	92
2.4	96	96	96	96	96	96	96
2.5	76	76	76	76	76	76	76
Sum	900,000	191,312	107,439	61,757	34,220	18,896	11,049

Although MSE makes less frequent data more important, due to the imbalance of data, the model is trained to reduce error for more abundant data. This causes the error of the rare data to be larger than the common data. To alleviate this problem, we split the data into k subsets. Then, calculate the mean error of each subset named as $average\_error\_\sigma_k$ and their average named as $average\_error\_\sigma$. The ratio between $average\_error\_\sigma_k$ and $average\_error\_\sigma$ is used as a weight for the error of each sample from different subset.

As the weight of error is sensitive to the cost of each subset, we called the loss function as cost-sensitive MSE, and its formula is shown as below.

$$Cost\_Sensitive\_MSE = \frac{1}{N}\sum_{i=1}^{N} \frac{average\_error\_\sigma_k}{average\_error\_\sigma} \times (y_t^i - y_p^i)^2 \tag{4}$$

Compared to the original MSE cost, we added a weight to the squared error for each sample. The weight is the ratio of $average\_error\_\sigma_k$ and $average\_error\_\sigma$. This new loss function is implemented as follows.

(a) Calculate the range of target variable (Range) and minimum value of target variable (Min).
(b) Split the data into 10 subsets depending on Range. For example, samples with target variable larger than (0.2 × Range + Min) and smaller than (0.3 × Range + Min) belong to subset_3.
(c) After the first training epoch, calculate the mean error of each subset, which is named the $average\_error\_\sigma_k$.
(d) Calculate the mean value of $average\_error\_\sigma_k$ for 10 subsets, which is the $average\_error\_\sigma$.

For the subsets with fewer samples, $average\_error\_\sigma_k$ will be larger than $average\_error\_\sigma$ and weight coefficient will be larger than 1. For the subsets with more samples, $average\_error\_\sigma_k$ will be smaller than $average\_error\_\sigma$ and weight coefficient will be smaller. Unlike using higher order exponential on errors, which is applied on all samples, the method we proposed is tuned for samples in specific intervals. The advantage of this method is that it can not only increase the importance of rare data, but also avoids the negative effects from high-order exponential errors applied to all samples in the training data set.

3.4. Model Ensemble Strategy

As undersampling method is used for data balancing, only a small part of samples are used for training. In order to better utilize the training samples, the model ensemble strategy is used as shown in Figure 7. It is a bagging method that uses different sets of undersampled data and trains multiple predictors. Finally, the results from multiple predictors are averaged to improve generalization performance of the predictor.

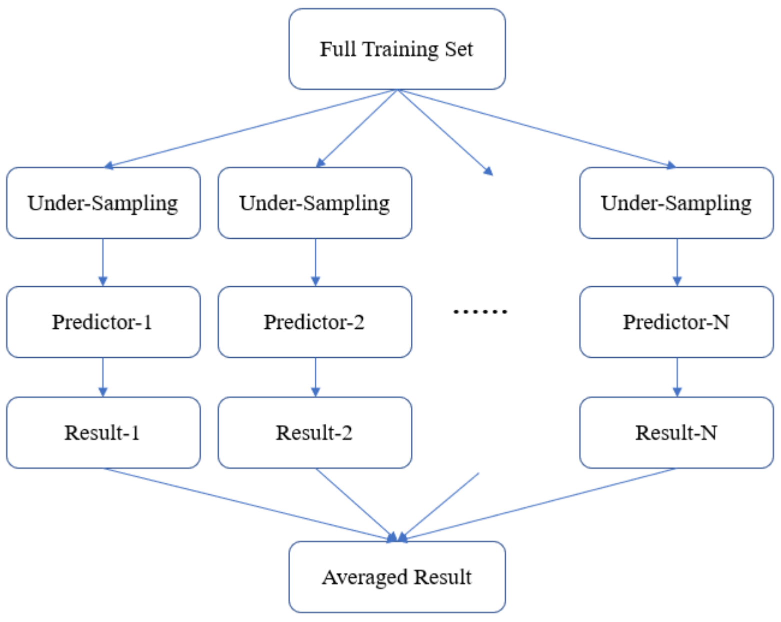

Figure 7. Process of model ensemble.

4. Experiments and Results

4.1. Evaluation Method

Like the training set, for the test set too we need to attend to the imbalance problem. If the whole test set, which contains 90,112 samples, is used for evaluation, even if the mean error of the whole test set is very low, for less frequent samples the prediction could be poor. In order to effectively evaluate the results, undersampling is used on the test set too. We randomly take only 100 samples for those range of target values which contains over 100 samples, and all remaining (not used during training) samples for less frequent segments of the range from the test sample set. We finally took 1592 samples for testing. To improve the reliability of the evaluation results, 10 times test experiments are performed and the averaged results are used for comparison.

R-square score (R2) calculated using Equation (5) is used to evaluate the prediction results. The greater the value of R2, the better the regression result.

$$R2 = 1 - \frac{\sum_{i=1}^{N}(y_t^i - y_p^i)^2}{\sum_{i=1}^{N}(y_m^i - y_p^i)^2} \tag{5}$$

4.2. Experiments with Different Undersampling Strategies

Experimental results with different undersampling subsets are presented in Table 4. Different sampling strategies are listed in Table 3. Test-1 to Test-10 are results with different test samples. The entries of Table 4 are the R2 scores. In this experiment, XGBoost is used because it is the fastest algorithm among the 3. RF and ResNet are the other two, and are much slower. Sampling-5 strategy achieved the best R2 score of 0.5591 with 18,896 samples, and sampling-4 strategy achieved the second-best R2 score of 0.5523 with 34,220 samples. Our goal is to achieve a good result while keeping as many samples as possible. Even though sampling-5 strategy achieved a better R2 score result than sampling-4 strategy, sampling-4 strategy used almost 2 times the number of training samples. On this account, sampling-4 strategy which achieved an average R2 score of 0.5523, and use 34,220 training samples, is selected as the best under-sampling strategy for this problem. This strategy is used in all following experiments.

Table 4. Experiments for different undersampling strategies used XGBoost.

Test Set	Original	Sampling-1	Sampling-2	Sampling-3	Sampling-4	Sampling-5	Sampling-6
1	0.2507	0.4586	0.5065	0.5389	**0.5581**	0.5667	0.5532
2	0.2488	0.4540	0.5005	0.5389	**0.5562**	0.5622	0.5473
3	0.2507	0.4521	0.4967	0.5309	**0.5455**	0.5543	0.5384
4	0.2484	0.4538	0.5017	0.5341	**0.5574**	0.5633	0.5482
5	0.2498	0.4537	0.5026	0.5348	**0.5526**	0.5604	0.5426
6	0.2468	0.4507	0.4977	0.5313	**0.5453**	0.5585	0.5407
7	0.2447	0.4484	0.4997	0.5358	**0.5545**	0.5644	0.5492
8	0.2461	0.4499	0.4948	0.5328	**0.5471**	0.5544	0.5359
9	0.2449	0.4494	0.4997	0.5361	**0.5561**	0.5649	0.5523
10	0.2423	0.4472	0.4973	0.5326	**0.5499**	0.5618	0.5499
Average	0.2473	0.4518	0.4997	0.5346	**0.5523**	0.5591	0.5458

4.3. Experiments with Different Regression Algorithms

The next step is to test the efficacy of different regression algorithm. Experimental results using three regression algorithms with sampling-4 are presented in Table 5. It shows that XGBoost achieved the best result of 0.5523 of R2 score (also shown in Table 4). Therefore, XGBoost is selected as a standard predictor in the following experiments.

Table 5. Experiments with different regression algorithms.

Test Set	Random Forest	XGBoost	ResNet
1	0.5392	0.5581	0.5322
2	0.5369	0.5562	0.5266
3	0.5289	0.5455	0.5265
4	0.5362	0.5574	0.5338
5	0.5337	0.5526	0.5357
6	0.5289	0.5453	0.5254
7	0.5387	0.5545	0.5353
8	0.5267	0.5417	0.5238
9	0.5389	0.5561	0.5288
10	0.5353	0.5499	0.5289
Average	0.5343	**0.5523**	0.5297

Additionally, we used random forest to evaluate the importance of each variable by calculating the decrease in gini_index, when that variable is used for decision at a higher level node. The top 6 important variables are shown in Table 6.

Table 6. Factors important in predicting Creatinine value.

Variable	Decrease in Gini Index
Sex	0.49
Age	0.11
HMG	0.10
OLIG_PROTE_CD	0.08
Waist	0.04
SMOKE_STATE	0.04

4.4. Experiments with Cost-Sensitive Loss Function

Before using cost-sensitive loss, the average R2 score was 0.5523 as shown in Table 4. After using cost-sensitive loss, the result of R2 score is improved slightly to 0.5546. From the simulations, we observed that the result of R2 score did not improve much by using cost-sensitive loss function. However, the cost-sensitive loss will increase the R2 score of the subset of data in the range where the number of data is small. From Figure 8 and Table 7, we can observe that in the range where there is fewer data, the error decreases, and the range where there is more data, the error increases. As accurately predicting in the region of high creatinine is more important due to higher risk of CKD. Thus, the cost-sensitive loss function achieved a positive goal.

By using sampling-4 strategy of undersampling for data balancing, and cost-sensitive loss function, the impact of data imbalance problem is partially mitigated. The result is shown in Figure 9, where the *y*-axis represents the ground truth value and the *x*-axis is the predicted value. The left part is the result before dealing with imbalance problem; the right part is the result after using undersampling and cost-sensitive loss function. We can observe that the model is able to predict with better accuracy over the whole range of data value.

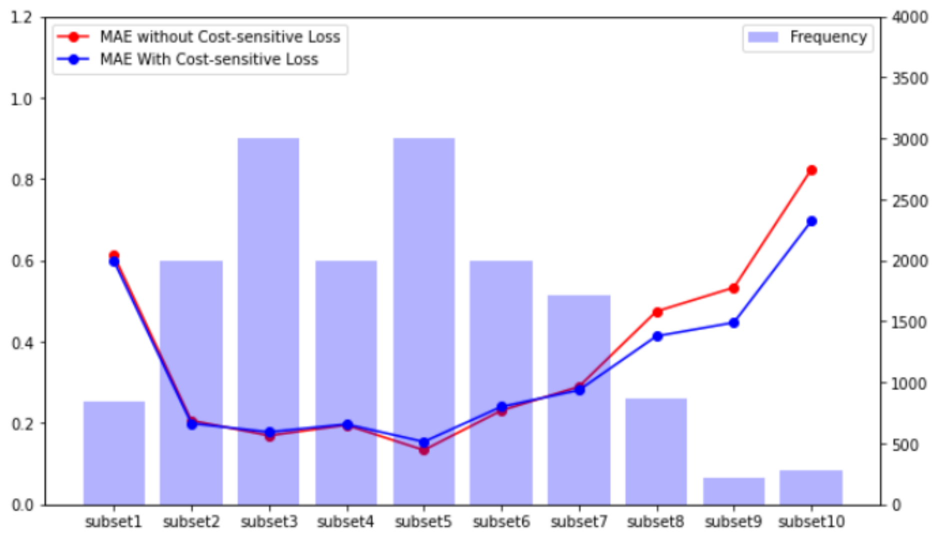

Figure 8. Effect of cost-sensitive loss function.

Table 7. Experiments with different loss function. XGBoost is used for the regression model.

Error Using Data Subset	Number of Data in the Subset	Without Cost-Sensitive Loss	Error Comparison	With Cost-Sensitive Loss
1	840	0.6125	>	0.6036
2	2000	0.2065	>	0.1974
3	3000	0.1693	<	0.1799
4	2000	0.1952	<	0.1972
5	3000	0.1336	<	0.1544
6	2000	0.2309	<	0.2422
7	1710	0.2896	>	0.2831
8	870	0.4746	>	0.4081
9	220	0.5332	>	0.4560
10	280	0.8235	>	0.6965

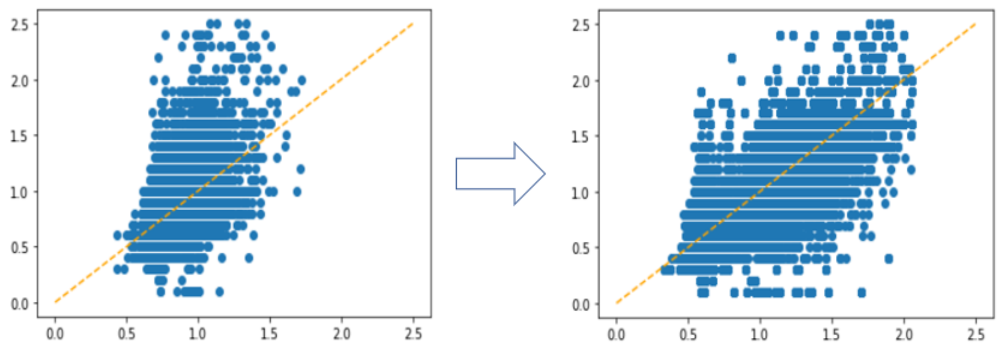

Figure 9. Effect of undersampling and cost-sensitive Loss.

4.5. Experiments with Model Ensemble

As we used undersampling method and selected only 34,220 samples to train a regression model, most of the data is not used for model training. In order to make full use of the data, 8 times undersampling were processed to extract 8 training datasets. P1 to

P8: eight different models were trained by these eight datasets, using XGBoost. Table 8 shows results of 8 predictors and every single predictor achieved almost the same averaged R2 score.

Table 8. Experiments with model ensemble.

Test Set \ Predictor	P1	P2	P3	P4	P5	P6	P7	P8	Average
1	0.5562	0.5574	0.5576	0.5569	0.5591	0.5568	0.5624	0.5599	0.5583
2	0.5537	0.5483	0.5562	0.5551	0.5559	0.5519	0.5542	0.5538	0.5536
3	0.5449	0.5421	0.5516	0.5432	0.5476	0.5433	0.5512	0.5483	0.5465
4	0.5617	0.5605	0.5656	0.5625	0.5589	0.5509	0.5625	0.5614	0.5605
5	0.5498	0.5505	0.5530	0.5502	0.5496	0.5507	0.5526	0.5522	0.5511
6	0.5498	0.5474	0.5503	0.5499	0.5508	0.5483	0.5531	0.5503	0.5500
7	0.5635	0.5650	0.5690	0.5664	0.5707	0.5676	0.5708	0.5680	0.5676
8	0.5520	0.5511	0.5527	0.5503	0.5502	0.5528	0.5560	0.5539	0.5524
9	0.5592	0.5512	0.5585	0.5571	0.5595	0.5533	0.5585	0.5573	0.5568
10	0.5497	0.5458	0.5491	0.5495	0.5483	0.5473	0.5499	0.5459	0.5482
Average	0.5541	0.5519	0.5564	0.5541	0.5551	0.5533	0.5571	0.5551	0.5546

Table 9 shows results of model ensembling. 1P means the results from P1; 2Ps means the averaged result from 1P and 2P; 3Ps means the averaged result from 1P, 2P, and 3P; and so on. We achieve a better result of R2 score of 0.5590 by using model ensembling strategy. More important is that generalization performance is improved by using averaged result from multiple predictors.

Table 9. Experiments with model ensemble.

Test Set	1P	2Ps	3Ps	4Ps	5Ps	6Ps	7Ps	8Ps
1	0.5562	0.5596	0.5605	0.5608	0.5615	0.5615	0.5623	0.5626
2	0.5537	0.5538	0.5562	0.5571	0.5579	0.5577	0.5580	0.5581
3	0.5449	0.5462	0.5497	0.5492	0.5499	0.5496	0.5506	0.5510
4	0.5617	0.5642	0.5663	0.5665	0.5661	0.5661	0.5663	0.5664
5	0.5498	0.5528	0.5545	0.5547	0.5546	0.5548	0.5552	0.5555
6	0.5498	0.5514	0.5526	0.5532	0.5536	0.5536	0.5542	0.5543
7	0.5635	0.5669	0.5692	0.5696	0.5708	0.5711	0.5718	0.5720
8	0.5520	0.5542	0.5553	0.5553	0.5552	0.5556	0.5564	0.5567
9	0.5592	0.5578	0.5596	0.5602	0.5610	0.5605	0.5610	0.5611
10	0.5497	0.5503	0.5516	0.5522	0.5525	0.5524	0.5527	0.5525
Average	0.5541	0.5557	0.5575	0.5579	0.5583	0.5583	0.5588	0.5590

4.6. Prediction the Risk of CKD

The predicted creatinine value and the original 23 features will be combined to predict the risk of CKD. We trained a logistic regression model using XGBoost and sampling-4 method for training.

First, a test set of 1592 samples, which contains 555 CKD samples and 1037 non-CKD samples, is used for testing. The Receiver Operating Characteristic (ROC) curve is drawn in Figure 10, by varying the threshold. The area under the ROC curves (AUC) was 0.90. In order to display the results more precisely, Table 10 shows the results of true positive (TP), false positive (FP), true negative (TN), false negative (FN), true positive rate (TPR), and true negative rate (TNR) by varying the threshold. If the threshold is set at 0.3, we could achieve a TPR of 84% and a TNR of 83% under the condition that creatinine data is not available.

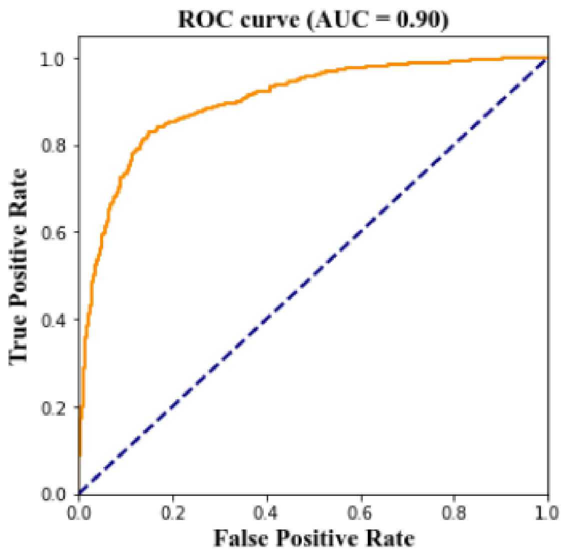

Figure 10. The ROC curve of 1592 testing samples.

Table 10. Performance with varying threshold value on classification of 1592 samples.

Threshold	TP	FN	TN	FP	TPR	TNR
0.0	555	0	0	1037	100%	0%
0.1	523	32	570	467	94%	55%
0.2	488	67	761	276	88%	73%
0.3	464	91	861	176	84%	83%
0.4	435	120	909	128	78%	88%
0.5	401	154	944	93	72%	91%
0.6	348	207	975	62	63%	94%
0.7	290	265	996	41	52%	96%
0.8	218	337	1016	21	39%	98%
0.9	127	428	1027	10	23%	99%
1.0	0	555	1037	0	0%	100%

As the ratio of CKD and non-CKD in 1592 samples is different from reality, next the whole test set of 90,112 samples, which contains 3248 CKD samples and 86,864 non-CKD samples, is used for testing. The Receiver Operating Characteristic (ROC) curve is shown in Figure 11, by varying the threshold. The area under the ROC curves (AUC) was 0.76. Table 11 shows the detailed results with varying threshold. Obviously, when the whole test set of 90,112 samples is used for testing, the overall result is worse. This is because the test data are very unbalanced. If we use the same threshold of 0.3, the TPR decreases from 84% to 56% and the TNR decreases from 83% to 80%. This is because there are many samples that are difficult to clearly define the class. Even so, if the threshold is set at 0.4, 13,540 (1598 + 11,942) samples will be classified as positive. There will be 49.1% (1598/3248) positive samples detected, and only 15.6% (13,540/86,864) of the population need to be checked. In this way, we can reduce the impact of CKD through a compromise of testing creatinine for those identified in the CKD risk class.

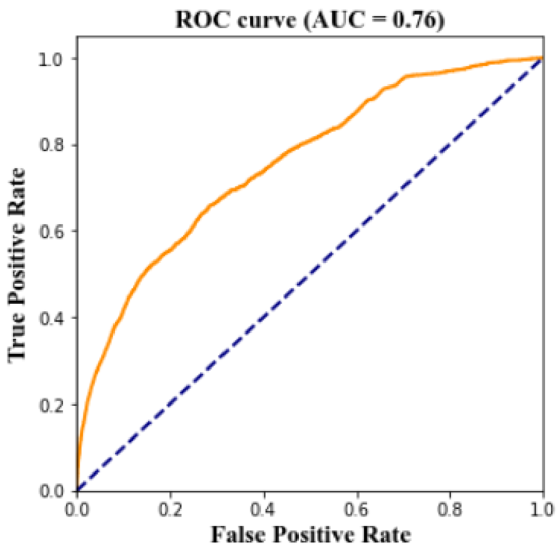

Figure 11. The ROC curve of 90,112 testing samples.

Table 11. Performance with varying threshold value of 90,112 testing samples.

Threshold	TP	FN	TN	FP	TPR	TNR
0.0	3248	0	0	86,864	100%	0%
0.1	2648	600	42,096	44,768	82%	48%
0.2	2171	1077	60,526	26,338	67%	70%
0.3	1805	1443	69,360	17,504	56%	80%
0.4	1598	1650	74,922	11,942	49%	86%
0.5	1245	2003	79,564	7300	38%	92%
0.6	971	2277	82,275	4589	30%	95%
0.7	720	2528	84,422	2422	22%	97%
0.8	476	2772	85,770	1094	15%	99%
0.9	138	3110	86,730	134	4%	99%
1.0	0	3248	86,864	0	0%	100%

5. Conclusions and Future Work

This study aims to build a regression model to predict the value of creatinine, then combine the predicted value of creatine with the common 23 health factors to evaluate the risk of CKD. As the creatinine value, which is the target variable, is extremely unbalanced, we used an undersampling method and proposed a cost-sensitive mean squared error (MSE) loss function to deal with the problem. Regrading model selection, we used three machine learning models: a bagging tree model named Random Forest, a boosting tree model named XGBoost, and a neural network based model named ResNet. To improve the result of creatinine predictor, we averaged results from eight predictors and ensembled their results. The ensembled model showed the best performance of R2 0.5590. The top six factors that influence creatinine are sex, age, hemoglobin, the level of urine protein, waist, and habit of smoking. With the predicted value of creatinine, an area under Receiver Operating Characteristic curve (AUC) of 0.76 is achieved when classifying samples for CKD.

Author Contributions: Formal analysis, W.W. and G.C.; Software, W.W.; Supervision, G.C. and B.C.; Writing—original draft, W.W.; Writing—review and editing, G.C. and B.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data available in a publicly accessible repository that does not issue DOIs Publicly available datasets were analyzed in this study. This data can be found here: [https://nhiss.nhis.or.kr/].

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Stevens, P.E.; Levin, A. Evaluation and management of chronic kidney disease: Synopsis of the kidney disease: Improving global outcomes 2012 clinical practice guideline. *Ann. Intern. Med.* **2013**, *158*, 825–830. [CrossRef] [PubMed]
2. Bikbov, B.; Perico, N.; Remuzzi, G. Disparities in chronic kidney disease prevalence among males and females in 195 countries: Analysis of the Global Burden of Disease 2016 Study. *Nephron* **2018**, *139*, 313–318. [CrossRef] [PubMed]
3. Couser, W.G.; Remuzzi, G.; Mendis, S.; Tonelli, M. The contribution of chronic kidney disease to the global burden of major noncommunicable diseases. *Kidney Int.* **2011**, *80*, 1258–1270. [CrossRef] [PubMed]
4. National Institute of Diabetes and Digestive and Kidney Diseases. *Chronic Kidney Disease Tests and Diagnosis*; National Institute of Diabetes and Digestive and Kidney Diseases: Bethesda, MD, USA, 2016.
5. Yarnoff, B.O.; Hoerger, T.J.; Simpson, S.K.; Leib, A.; Burrows, N.R.; Shrestha, S.S.; Pavkov, M.E. The cost-effectiveness of using chronic kidney disease risk scores to screen for early-stage chronic kidney disease. *BMC Nephrol.* **2017**, *18*, 85. [CrossRef] [PubMed]
6. Kuo, C.C.; Chang, C.M.; Liu, K.T.; Lin, W.K.; Chiang, H.Y.; Chung, C.W.; Ho, M.R.; Sun, P.R.; Yang, R.L.; Chen, K.T. Automation of the kidney function prediction and classification through ultrasound-based kidney imaging using deep learning. *NPJ Digit. Med.* **2019**, *2*, 1–9. [CrossRef] [PubMed]
7. Basak, S.; Alam, M.M.; Rakshit, A.; Al Marouf, A.; Majumder, A. Predicting and Staging Chronic Kidney Disease of Diabetes (Type-2) Patient Using Machine Learning Algorithms. *Int. J. Innov. Technol. Explor. Eng.* **2019**, *8*. [CrossRef]
8. Kumar, M. Prediction of chronic kidney disease using random forest machine learning algorithm. *Int. J. Comput. Sci. Mob. Comput.* **2016**, *5*, 24–33.
9. Rady, E.H.A.; Anwar, A.S. Prediction of kidney disease stages using data mining algorithms. *Inform. Med. Unlocked* **2019**, *15*, 100178. [CrossRef]
10. Chimwayi, K.B.; Haris, N.; Caytiles, R.D.; Iyengar, N.C.S. Risk Level Prediction of Chronic Kidney Disease Using Neuro-Fuzzy and Hierarchical Clustering Algorithm (s). *Int. J. Multimedia Ubiq. Eng.* **2017**, *12*, 23–36. [CrossRef]
11. Almansour, N.A.; Syed, H.F.; Khayat, N.R.; Altheeb, R.K.; Juri, R.E.; Alhiyafi, J.; Alrashed, S.; Olatunji, S.O. Neural network and support vector machine for the prediction of chronic kidney disease: A comparative study. *Comput. Biol. Med.* **2019**, *109*, 101–111. [CrossRef] [PubMed]
12. Salekin, A.; Stankovic, J. Detection of chronic kidney disease and selecting important predictive attributes. In Proceedings of the 2016 IEEE International Conference on Healthcare Informatics (ICHI), Chicago, IL, USA, 4–7 October 2016.
13. Almasoud, M.; Ward, T.E. Detection of chronic kidney disease using machine learning algorithms with least number of predictors. *Int. J. Soft Comput. Its Appl.* **2019**, *10*. [CrossRef]
14. Rubini, L. Jerlin. Department of Computer Science and Engineering, Algappa University: TamilNadu. 2015. Available online: http://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease (accessed on 8 June 2020).
15. Liaw, A.; Wiener, M. Classification and regression by randomForest. *R News* **2002**, *2*, 18–22.
16. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
17. Chen, D.; Hu, F.; Nian, G.; Yang, T. Deep Residual Learning for Nonlinear Regression. *Entropy* **2020**, *22*, 193. [CrossRef] [PubMed]
18. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, 1189–1232. [CrossRef]
19. Buda, M.; Maki, A.; Mazurowski, M.A. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Netw.* **2017**, *106*, 249–259. [CrossRef] [PubMed]