Intelligent Systems with Applications 16 (2022) 200144

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Chi^2-MI: A hybrid feature selection based machine learning approach in diagnosis of chronic kidney disease

Samrat Kumar Dey [a,*], Khandaker Mohammad Mohi Uddin [b], Hafiz Md. Hasan Babu [c], Md. Mahbubur Rahman [d], Arpita Howlader [e], K.M. Aslam Uddin [f]

[a] School of Science and Technology, Bangladesh Open University, Gazipur 1705, Bangladesh
[b] Department of Computer Science and Engineering, Dhaka International University, Dhaka 1205, Bangladesh
[c] Department of Computer Science and Engineering, University of Dhaka, Dhaka 1000, Bangladesh
[d] Department of Computer Science and Engineering, Military Institute of Science and Technology, Dhaka 1216, Bangladesh
[e] Department of Computer and Communication Engineering, Patuakhali Science and Technology University, Dumki 8602, Bangladesh
[f] Department of Information and Communication Engineering, Noakhali Science and Technology University, Noakhali 3814, Bangladesh

ARTICLE INFO

Keywords:
Machine learning
CKD
Classification
Feature selection
Healthcare informatics

ABSTRACT

Early detection and characterization are considered crucial in treating and controlling the chronic renal disease. Because of the rising number of patients, the high risk of progression to end-stage renal disease, and the poor prognosis of morbidity and mortality, chronic kidney disease (CKD) is a significant burden on the healthcare system. Detecting CKD in its early stages is critical for saving millions of lives. The uniqueness of this study lies in developing a diagnosis system to detect chronic kidney disease using different Machine Learning (ML) algorithms with the support of a hybrid feature selection approach. This study exploited the 400 clinical data of CKD patients based on the dataset supplied by the University of California Irvine (UCI) available at their Machine Learning repository. Different data preparation techniques like encoding categorical features, missing values imputation, removing outlier factors, handling data imbalance, scaling data at the same level, and selecting relevant features are adopted to prepare the dataset for the prediction model. A hybrid Chi-squared test (Chi2) and Mutual Information (MI) based feature selection approach is proposed to remove redundant features, and a Pearson correlation matrix is also computed to consider the top important features for the prediction. Lastly, the Extra tress classifier can diagnose CKD with 98% accuracy and a 2% true negative rate without data leakage out of 14 machine learning models. On the other hand, the Bagging classifier performed worst with only 60% accuracy.

1. Introduction

The kidney is an essential organ in the body because it filters or removes metabolic waste products from blood plasma and excretes them in the urine. The kidney likewise performs the following critical functions: generating and releasing hormones that regulate blood pressure, regulating the body's fluid or electrolyte balance, controlling red blood cell synthesis via managing PH, and generating an active form of vitamin D that supports healthy and strong bones. Chronic Kidney Disease (CKD) is a kidney disease that can be treated in its early stages but leads to kidney failure in its last stages. Because of its high mortality rate, CKD has gotten much attention. According to the World Health Organization (WHO), chronic diseases have become a significant hazard to emerging countries (World Health Organization, 2005).

CKD caused the death of 753 million people globally in 2016, with 336 million men and 417 million women (Bikbov, Perico & Remuzzi, 2018). It is considered a "chronic" disease because kidney disease develops gradually and lasts long, compromising the urinary system's function. Diabetes, high blood pressure, and cardiovascular disease (CVD) are risk factors for CKD patients (Chen, Zhang & Zhang, 2016). Patients with CKD experience side effects that harm the neurological and immunological systems, particularly in the late stages. Patients in developing countries may reach the last stages, requiring dialysis or a kidney transplant.

* Corresponding author at: School of Science and Technology, Bangladesh Open University, Gazipur 1705, Bangladesh.
E-mail addresses: samrat.sst@bou.ac.bd (S.K. Dey), mohiuddin.cse@diu.ac (K.M.M. Uddin), hafizbabu@du.ac.bd (H.Md.H. Babu), mahbub@cse.mist.ac.bd (Md.M. Rahman), arpita.cse@pstu.ac.bd (A. Howlader), aslam@nstu.edu.bd (K.M.A. Uddin).

https://doi.org/10.1016/j.iswa.2022.200144
Received 15 August 2022; Received in revised form 25 September 2022; Accepted 21 October 2022
Available online 28 October 2022

The glomerular filtration rate (GFR), which represents kidney function, is used by doctors to diagnose renal disease. GFR is calculated using the patient's age, blood test results, gender, and other parameters related to the patient (Anon, 2015). Doctors can divide CKD into five stages based on the GFR number. Table 1 depicts the stages of renal disease as measured by GFR. If diagnosed and treated early, chronic renal disease can be prevented from progressing to kidney failure. The best method to address chronic kidney disease is to detect it earlier, but delaying until it's too late might lead to kidney failure, necessitating dialysis or kidney transplantation to live a normal life. A blood test to assess the glomerular filtrate or a urine test to check albumin are utilized in the medical diagnosis of chronic kidney disease. Because of the growing number of chronic renal patients, the lack of specialized physicians, and the high costs of diagnosis and treatment, particularly in developing countries, computer-assisted diagnostics are needed to aid physicians and radiologists in making diagnostic decisions.

Machine Learning (ML) and Deep Learning (DL) are the latest methods for analyzing data and predicting consequences. An ML model is trained to generate classifications and predict future events using statistical methods, algorithms, and data. If it is predicted that a patient may suffer CKD in advance, it may be able to take preventative steps and offer better therapy, avoiding undesirable outcomes such as dialysis or transplantation. Diabetes, heart disease, blood pressure, certain foods, and family history are all factors that impair kidney function and cause CKD. Fig. 1 depicts some of the factors that influence chronic kidney disease.

This article contributes to developing a replicable hybrid feature section approach (Chi2-MI) by applying 14 distinct machine learning algorithms to detect CKD and non-CKD successfully. The proposed method can also be utilized with another similar health dataset to diagnose any similar ailment. The Extra Trees classifier with hybrid feature section technique outperformed others in nearly all dataset models, according to testing results and comparisons with relevant studies.

2. Related works

A plethora of research has been carried out to predict CKD stages utilizing various classification algorithms. Table 2 summarizes current research methods for diagnosing CKD using Machine Learning algorithms and performance measures. A fuzzy-based smart system has been developed by Ahmed et al. (Ahmed, Kabir, Mahmood & Rahman, 2014) to test the urinary system. The researchers utilized a dataset that included 817 patient clinical test results from Birdem Hospital in Dhaka for two years (2011–2013) and obtained an accuracy of 86.7%.

Khamparia et al. (Khamparia et al., 2020) proposed a model to detect CKD using a deep-stacked autoencoder model. The proposed model used UCI's machine learning database (CKD= 250, Not CKD=150) and obtained an Accuracy of 100% with a Specificity of 100%.

A genetic algorithm (GA) has been proposed by Kim et al. (Kim & Ye, 2021) that is based on neural networks (NN) for performing CKD diagnosis. Their proposed GA optimized the weight vectors to train the NN. The researchers utilized 741 ultrasound images, including 251 images of normal kidneys, 328 images of mild and moderate CKD, and 162 images of severe CKD. Using the artificial neural network (ANN), the authors obtained a 95.4% classification rate.

Machine learning techniques were proposed by Almansour et al. (Almansour et al., 2019) to diagnose CKD at an early stage. The authors used a dataset of 400 patients and 24 parameters linked to the diagnosis of chronic renal disease in the classification methods. The researchers utilized ANN and SVM to diagnose CKD and obtained the accuracy of ANN and SVM was 99.75% and 97.75%, respectively.

To diagnose a CKD dataset, Rady et al. (Rady & Anwar, 2019) used SVM, Multilayer perceptron (MLP) and radial basis function (RBF), and probabilistic neural networks (PNN) methods. The authors observed that SVM, MLP, and RBF algorithms outperformed the PNN method with an accuracy of 96.7%.

Kunwar et al. (Kunwar, Chandel, Sabitha & Bansal, 2016) utilized naive Bayes and artificial neural networks (ANN) to diagnose CKD in a UCI dataset. The experiment was carried out using the Rapidminer tool, and the accuracy of naive Bayes and ANN were found to be 100% and 72.73%, respectively. In this research, the authors considered four main factors: age, diabetes, blood pressure, and RBC count, to detect CDK.

Wibawa et al. (Wibawa, Maysanjaya & Putra, 2017) utilized correlation-based feature selection (CFS) for feature selection and AdaBoost for ensemble learning to enhance CKD diagnosis. Their system had the highest accuracy of 98.1% when using a hybrid of KNN with CFS and AdaBoost. The researchers also used the UCI machine learning repository with 400 instances.

Another work has been conducted by Avci et al. (Avci, Karakus, Ozmen & Avci, 2018) using WEKA software to diagnose the UCI dataset for CKD. The authors used K-Star, SVM, J48, and NB classifiers to evaluate the dataset and observed that with a 99% accuracy rate, the J48 algorithm outperformed the other algorithms.

Chiu et al. (Chiu, Chen, Wang, Chang & Chen, 2013) developed intelligence models utilizing neural network methods to classify CKD. For the early diagnosis of CKD, the models comprised generalized feed-forward neural networks (GRNN), back-propagation network (BPN), and modular neural network (MNN). The authors presented hybrid models that combined the GA with the previously stated models.

Sara et al. (Sara & Kalaiselvi, 2018) used two strategies such as Feature Selection (FS) and Hybrid Wrapper and Filter-Based FS (HWFFS), to minimize the dimensionality of the dataset and strongly select the features linked with CKD. The hybrid characteristics were identified using an SVM classifier after the features derived from the two approaches were blended.

Table 1
The development stages of CKD (Anon, 2015).

Stage	Description	Glomerular filtration rate (mL/min/1.73 m^2)
1	The kidneys are working normally	>=90
2	Damage to the kidneys is minor	60–89
3	Damage to the kidneys is moderate	30–59
4	Damage to the kidneys is sever	12–29
5	Kidney failure has already occurred	<=15

3. Materials and methods

This section extensively discussed the various methods and materials employed in this research. Dataset preparation techniques, hybrid feature selection approach, and the different machine learning algorithms applied to diagnose are presented and analyzed. The primary phase of this study was to examine the chronic kidney disease (CKD) dataset; therefore, several experiments were performed utilizing 14 machine learning algorithms: Ada Boost Classifier, Bagging Classifier, CatBoost Classifier, Decision Tree Classifier, Extra Trees Classifier, Gaussian Naive Bayes (Gaussian NB), Gradient Boosting Classifier, K-nearest neighbors, LightGBM Classifier, Multi-Layer Perceptron (MLP), Random Forest Classifier, Stochastic Gradient Boosting, Support vector machines (SVM), and XGBoost classifier. The Min-Max Scaler and Imputed K-Nearest Neighbor technique estimated missing numerical values. At the same time, the Local Outlier Factor (LOF) method was utilized to remove the outliers, and class balancing was conducted using the Synthetic Minority Oversampling Technique (SMOTE). After that, a novel Chi-Squared test (Chi^2) and Mutual Information (MI) based hybrid feature selection approach is applied to identify the most significant features. After that, the dataset values are standardized using the Standard Scalar method, and then trained the Machine learning (ML) Models

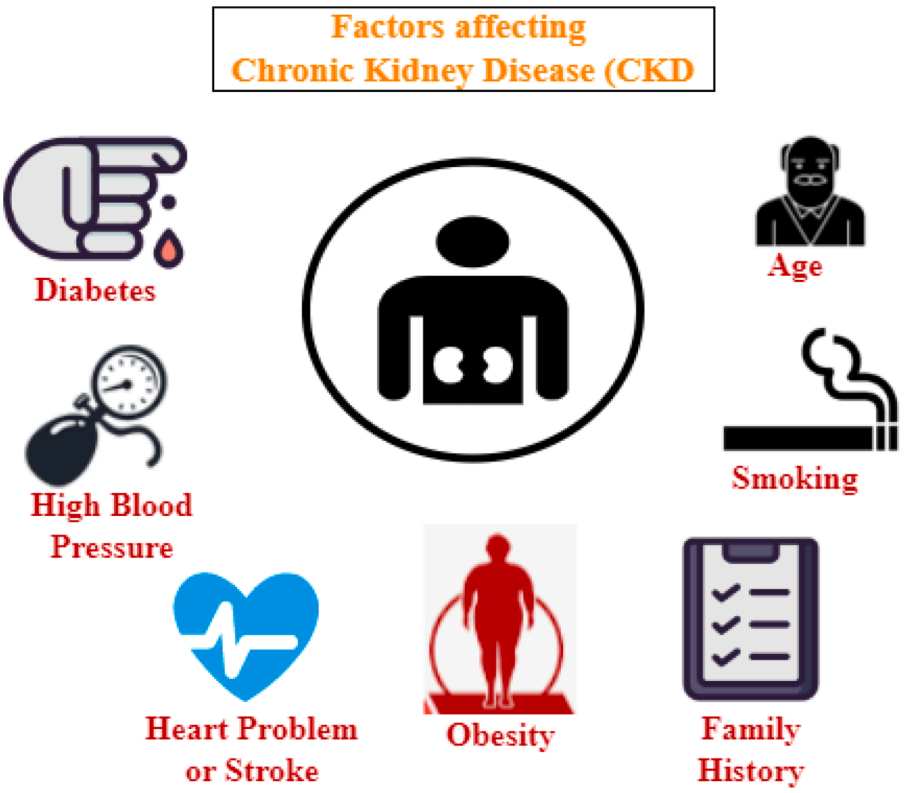

Fig. 1. Chronic Kidney Disease (CKD) affecting factors.

Table 2
Existing Chronic Kidney Disease prediction method and their performance metrics.

Existing work	Dataset	Modality	Method	Validation	Performance Metrics
Ahmed et al. (Ahmed et al., 2014)	817 patient clinical test results	Determine whether the urinary system is good or bad	Fuzzy Logic	Random Patient level	ACC: 86.7%
Khamparia et al. (Khamparia et al., 2020)	UCI's machine learning database. CKD= 250 Not CKD=150	CKD or not- CKD detection	Deep Stacked Autoencoder	Random Patient level	ACC: 100% PREC: 100%
Kim et al. (Kim & Ye, 2021)	Total 741 ultrasound images. 251 images of normal kidneys, 328 images of mild and moderate CKD, and 162 images of severe CKD	Normal, mild and moderate CKD, severe CKD	Artificial Neural Network (ANN) classification	Random Patient level	ACC: 95.4%
Almansour et al. (Almansour et al., 2019)	UCI's machine learning database. CKD= 250 Not CKD=150	CKD or not CKD Detection	ANN, SVM	Random Patient level	ACC: 99.75% (ANN) ACC: 97.75% (SVM)
Rady *et al.* (Rady & Anwar, 2019)	361 Indian CKD patient clinical test results and contained 25 variables (11 numerical, 14 categorical)	Diagnose of CKD	Probabilistic Neural Networks (PNN), Multi-layer perceptron (MLP), SVM, and Radial Basis Function (RBF)	Random Patient level	ACC: 96.7% (PNN)
Kunwar *et al.* (Kunwar et al., 2016)	UCI's machine learning database. CKD= 250 Not CKD=150	Diagnose of CKD	Naive Bayes algorithm, and ANN	Random Patient level	ACC: 100% (Naive Bayes algorithm) ACC: 72.73% (ANN)
Wibawa *et al.* (Wibawa et al., 2017)	UCI's machine learning database. CKD= 250 Not CKD=150	Diagnose of CKD	KNN, CFS, AdaBoost	Random Patient level	ACC: 98.1% (a hybrid of KNN with CFS and AdaBoost)
Avci et al. (Avci et al., 2018)	UCI's machine learning database. CKD= 250 Not CKD=150	Diagnose of CKD	K-Star, SVM, J48 and NB classifier	Random Patient level	ACC: 99% (J48)
Chiu et al. (Chiu et al., 2013)	430 patient clinical test results CKD= 285 Not CKD=145	Diagnose of CKD	BPN + GA	Random Patient level	ACC: 91.71%
Sara et al. (Sara & Kalaiselvi, 2018)	UCI's machine learning database. CKD= 250 Not CKD=150	Diagnose of CKD	SVM+HWFFS (Hybrid Wrapper and Filter-based approach)	Random Patient level	ACC: 90%

using that standardized data. For the validation, the K-fold cross-validation approach is adopted in this study, and finally, in the training data, the training data are encoded using the One-hot-encoding technique. Fig. 2 illustrates the workflow of the proposed methodology for CKD diagnosis.

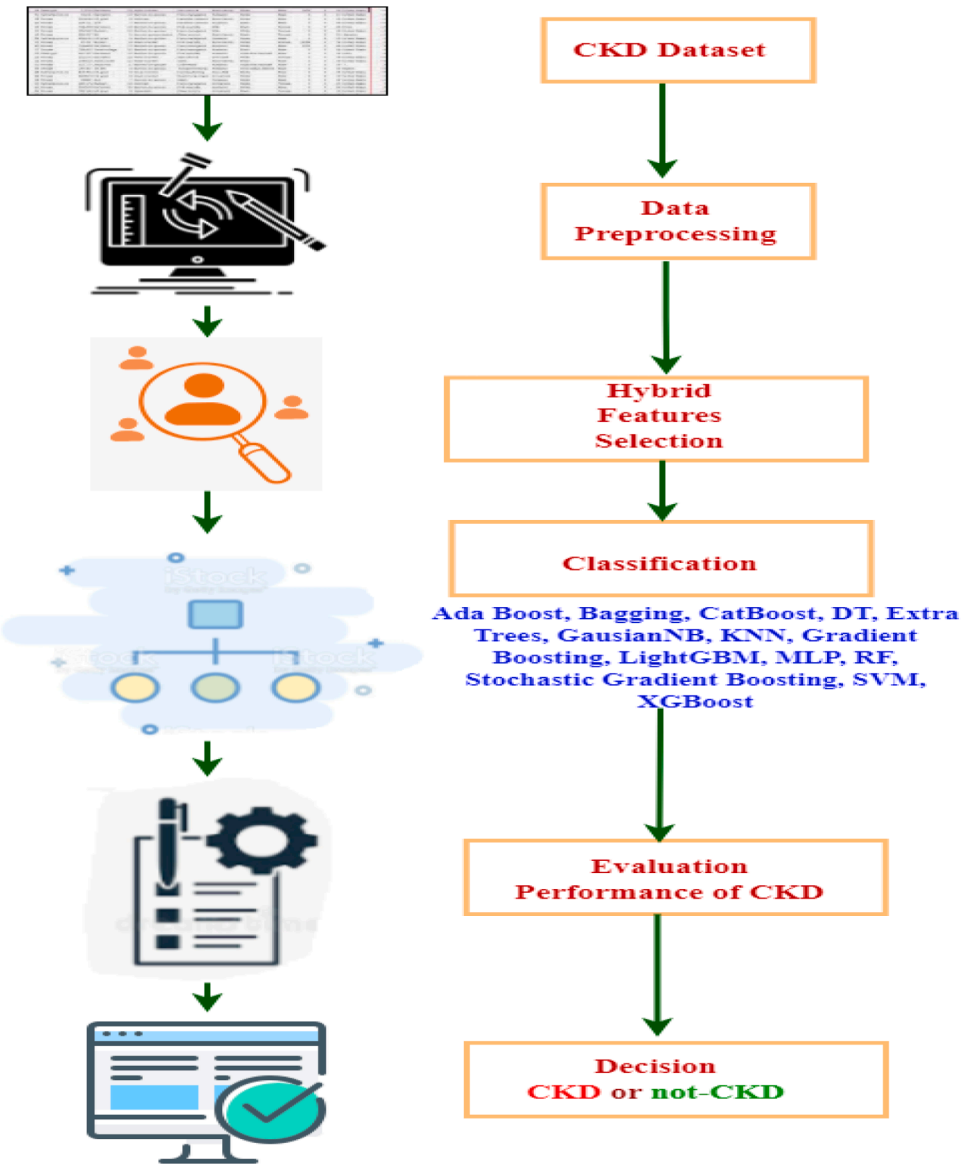

Fig. 2. Workflow of the proposed methodology.

3.1. Dataset preparation

The CKD dataset utilized in this study was collected from the University of California, Irvine Machine Learning Repository (Anon, 2022a), which included the chronic kidney disease data of 400 patients. In addition to the class features, such as "CKD" and "not-CKD," for classification, the dataset has 24 features separated into 11 numeric characteristics and 13 categorical features. There are two values in the diagnostic class: "CKD" and "not-CKD". The overview of the dataset attributes is illustrated in Table 3. Different data preprocessing approaches have been followed to prepare the dataset for the model.

In the beginning, the duplicate sample cheeked was utilized to avoid data leakage, and no single duplicate sample was found in the dataset. Then, the cross-validation and testing approach was followed to separate the 70% data for the training and 30% for testing. Data preparation procedures based solely on the training set were developed and used in the next phase for both partitions. While cleaning the dataset, 10 categorical (nominal) features have been observed, and with the support of the One-hot-encoding technique, these features have been encoded into 0′s or 1′s. The following Eq. (1) highlights the working procedure of the One-hot-encoding method. For e_i it is a vector of the standard base,

Table 3
General overview of the chronic kidney disease dataset available at the UCI machine learning repository (Anon, 2022a).

Dataset Characteristics	Multivariate
Attribute Characteristics	Real
The number of features (Col in general)	26
Number of instances	400
Number of features with numeric values	14
Number of categorical features	12
Number of features we've considered as independent variables	16
Target column Name	Classification
Instances are classified as	CKD/Not CKD
Number of classes defined as CKD	250
Number of classes defined as NOT CKD	150
Data Source	UCI ML Repo

where e_i denotes the vector with a 1 in the i^{th} coordinate and 0′s elsewhere.

$$e_i = 1_A(x) := \begin{cases} 1 \ if \ x \ \in A \\ 0 \ if \ x \ \in A \end{cases} \quad (1)$$

The dataset description of the available features and their label that influence CKD classification is explained in Table 4. Based on the data, the current form of the dataset is imbalanced as it comprises 250 examples of the "CKD" class (62.5%) and 150 cases of "not-CKD" (37.5%). Features or characteristics of the dataset include 24 different features and 1 binary classification attribute.

Normalization is another method for improving machine learning models' ability to find patterns in data by comparing and separating samples (Aljaaf et al., 2018). Surprisingly, the dataset contains several missing values which need to be imputed. This study employed KNN imputation techniques for the imputation because of its working principle that focuses on working through measuring distances. However, direct imputation based on raw data may be wrong. The features have been further rescaled using the Min-Max Scaller approach (Anon, 2022b). By using the following Eq. (2). It rescales the features range between 0 and 1. In the equation, χ denotes the real values whereas χ_{min} and χ_{max} are the minimum and maximum values of the feature f_i. The rescaled values are χ_ν and $\chi, \chi_{min}, \chi_{max} \in f_i$.

$$\chi_\nu = \frac{\chi - \chi_{min}}{\chi_{max} - \chi_{min}} \tag{2}$$

In the next phase, to handle the missing values, this research employed KNN imputation techniques as the dataset contained 1008 missing values from 24 features and 400 records. The missing values of each sample are imputed using the mean value of the k-neighbors found in the data set. For the KNN Imputation approach, the value of $k = 8$ is considered and assumed to be uniform weight. Table 5 highlights the statistical insights and the numerical features obtained from the dataset that supported this study to remove the missing values using different available data.

Furthermore, visualization of the distribution of 14 features with their missing values percentages is depicted in Fig. 3. From the distribution, some of the features show very distant outliers. Here, 3 features are treated as continuous values as these are measures of biological variables that act as continuous in reality. However, some features have high proportions of missing values; thus, they cannot be imputed with measures of central tendency and would distort their distributions. On the other hand, Fig. 4 highlights the distribution of the nominal features with the percentages of attributes present in the dataset. The visualization shows that the dataset has class imbalance problems, and Synthetic over-sampling techniques have been employed to resolve that issue.

Table 4
Description of features and attributes with the different datatype of the dataset (Anon, 2022a).

SL#	Features	Description	Data Type
1	age	Individual's ages in year	Numerical
2	bp	Blood Pressure measured in mm/Hg	Numerical
3	sg	Specific Gravity	Nominal
4	al	Nominal Albumin Value in Urine	Nominal
5	su	Nominal Sugar Value in blood	Nominal
6	rbc	Red Blood Cells: Normal or Abnormal	Nominal
7	pc	Pus cell: Normal or abnormal	Nominal
8	pcc	Pus cell clumps: present or not present	Nominal
9	ba	Bacteria: present or not present	Nominal
10	bgr	Blood Glucose Random measured in mgs/dl	Numerical
11	bu	Blood urea (mgs/dl)	Numerical
12	sc	Serum Creatinine (mgs/dl) value in blood	Numerical
13	sod	Sodium (mEq/L)	Numerical
14	pot	Potassium (mEq/L)	Numerical
15	hemo	Hemoglobin (gms)	Numerical
16	pcv	Packed Cell Volume	Numerical
17	wbcc	White blood Cells Count (Cells/cumm)	Numerical
18	rbcc	Red Blood Cells Count (millions/cumm)	Numerical
19	htm	Hypertension: Yes or No	Nominal
20	dm	Diabetes Mellitus: Yes or No	Nominal
21	cad	Coronary Artery Disease: Yes or No	Nominal
22	appet	Appetite: Good or Poor	Nominal
23	pe	Pedal Edema: Yes or No	Nominal
24	ane	Anemia: Yes or No	Nominal
25	Classification	Target Column: Classified as ckd or notckd	Binary

3.2. Features selection

In Qin et al. (2020), the authors stated that identifying the most critical risk variables in healthcare informatics helps eliminate redundant features, improve data consistency, minimize ML algorithm training time, and increase prediction performance. As a result, the use of various ways to choose characteristics or eliminate less important features has grown in favor over time. In recent years, researchers have used a variety of strategies to pick relevant features, including Principal Component Analysis (PCA), Chi-squared (Chi^2) Test, Mutual Information (MI), and Recursive Feature Elimination (RFE) techniques. This study describes a novel Chi^2-MI hybrid strategy derived from the Wrapper feature selection method. Eqs. (3) and (4) illustrates the working principle of the Chi^2 test and Mutual Information methods. In the Chi-square test, c is the degrees of freedom, O is the observed value, and E is the expected value. On the other hand, $p(x, y)$is the joint probability density function of X and Y, and where $p(x)$ and $p(y)$are the marginal density functions. The mutual information determines how similar the joint distribution $p(x, y);$ is to the products of the factored marginal distributions.

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{3}$$

$$I(X;Y) = \int_X \int_Y P(x,\ y) \log \frac{P(x,y)}{P(x)P(y)} dxdy \tag{4}$$

The approach eliminates the redundant features and imparts a subset λ_1 with important features where $\lambda_1\ \varepsilon$ *Features* (F). The mathematical explanation is as follows.

$$\lambda_1 = \{\varepsilon_w\} - \{\varepsilon_{correlation}\} \tag{5}$$

$$\varepsilon_w = \{x | x \in Chi^2 \cap MI\} \tag{6}$$

$$\varepsilon_{correlation} = \{x | x\, \varepsilon\, High\ correlation\ in\ features\} \tag{7}$$

$$Chi^2 = \{i | i \in imporatnt\ features\ found\ form\ the\ Chi - Squared\ Test\} \tag{8}$$

$$MI = \{i | i \in imporatnt\ features\ found\ form\ the\ Mutual\ Information\} \tag{9}$$

In the next phase, the top 80% of features have been selected for further analysis, and these features were stored in the two different sets of Chi^2 and MI. Identifying correlated features is also essential for reducing redundancy. Further pearson correlation check is performed and chose the highly correlative features which have over 85% correlation and stored them in $\varepsilon_{correlation}$. Lastly, the considering features for this study $\varepsilon_{correlation} = \{sg, al, su, sc, hemo, pcv, rbc, pc, pcc, htn, dm, appet, and\ pe\}$. Fig. 5 shows the correlation observed from the pearson correlation matrix check. The mathematical formula for the pearson correlation matrix is shown in Eq. (10).

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}} \tag{10}$$

The last step for data preprocessing is standardization. It is evident from the outcomes of the research work that various researchers claimed that it significantly impacts ML algorithm performance (Imran, Amin & Johora, 2018). Therefore, before going into the predictive model, this study standardizes the data using the standard scalar method with scikit-learn. The following formula (Eq. (11)) is used to compute the standard value of sample μ, where ν is considered as samples mean and ϕ

Table 5
Statistical data analysis of the 11 numerical features of the dataset.

Features	Mean	Std.	Min	Max	25%	50%	75%
Age (ag)	51.4833	16.9749	2	90	42	54	64
Blood Pressure (bp)	76.4691	13.7560	50	180	70	78	80
Blood glucose random (bgr)	148.0370	76.5830	22	490	101	126	150
Blood Urea (bu)	57.4260	49.9870	1.5	391	27	44	61.750
Hemoglobin (hemo)	12.5260	2.8150	3.1	17.8	10.87	12.53	14.625
Packed cell volume (pcv)	38.8840	8.7620	9	54	34	38.77	44
Potassium (pot)	4.6270	2.9200	2.5	47	4	4.63	4.8
Serum creatinine (sc)	3.0720	4.5120	0.4	76	0.9	1.4	3.070
Sodium (sod)	137.5260	9.9080	4.5	163	135	137.53	141
Red blood cell count (rc)	4.7070	0.8900	8	2.1	4.69	5.1	8
White blood cell count (wc)	8406.1200	2823.3000	2200	26,400	6975	8377.63	9400

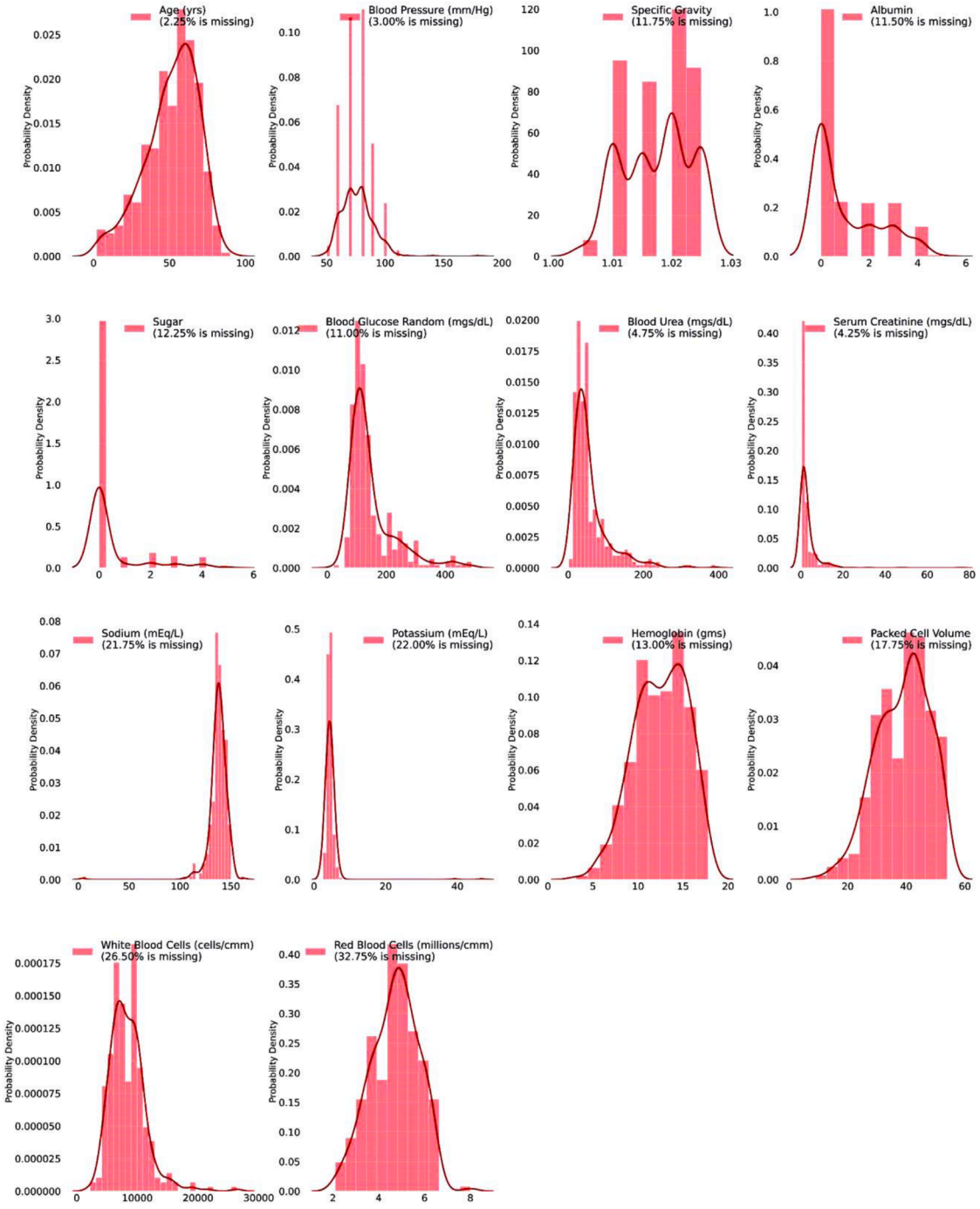

Fig. 3. Visualizing the features distribution of numerical features of the dataset with their missing percentage in the dataset.

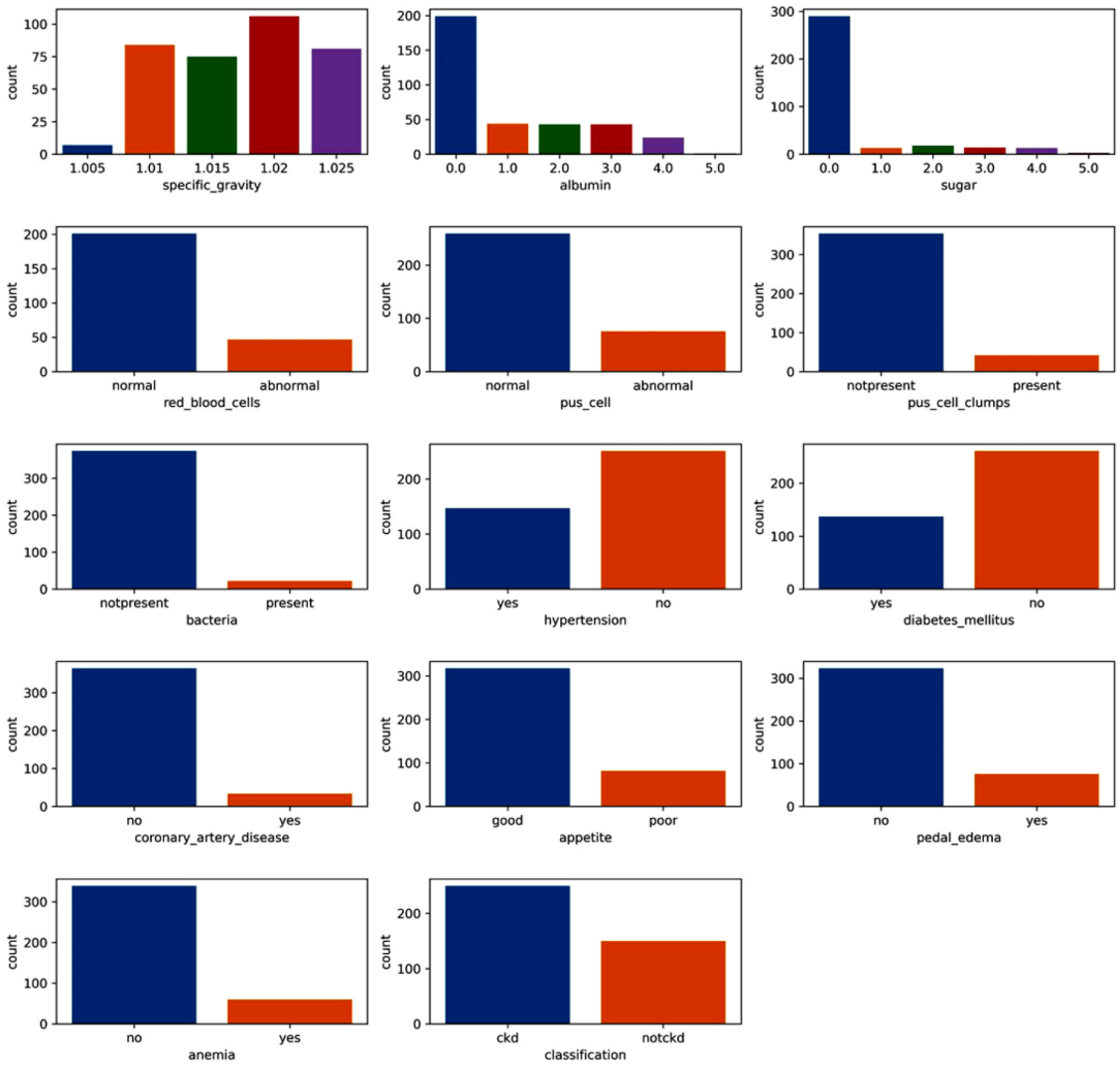

Fig. 4. Visualizing the features distribution of nominal features of the dataset.

as the training samples standard deviation.

$$\omega = \frac{(\mu - \nu)}{\phi} \tag{11}$$

3.3. Machine learning models

Data mining techniques were employed to create classification templates to develop novel and intelligible patterns (Aldhyani, Alshebami & Alzahrani, 2020). The creation of models based on past analysis is required for both the supervised and unsupervised learning approaches employed in clinical and medical diagnostics for regression and classification. In this section, the classification techniques explained are used in this research.

3.3.1. K-nearest neighbor classifier

This algorithm aims to keep objects that are similarly close together. The class labels and feature vectors of a data set are used in this model (Chatzigeorgakidis, Karagiorgou, Athanasiou & Skiadopoulos, 2018). KNN keeps track of all the cases and uses a similarity metric to help categorize new ones. Text is represented in K-nearest neighbors by a spatial vector $S = S(T_1,\ W_1; T_2,\ W_2; \ldots\ldots\ldots\ldots T_n,\ W_{n2};)$. The training text is used to find and compute similarity for any text, and the texts with the highest similarity are chosen. Finally, K neighbors are used to determine the classes.

The following formula compares the feature vectors of the incoming text with the feature vectors of each training text.

$$sim\ (Vi,\ Vj) = \frac{\sum_{k=1}^{N} TikTjk}{\sqrt{\sum_{k=1}^{N} Tik^2}\ \sqrt{\sum_{k=1}^{N} Tjk^2}} \tag{12}$$

Here, V_i is the feature vector of incoming text, and V_j is the feature vector of the training text. The dimension of the feature vector is defined by N. The k^{th} elements of vectors V_i and V_j are T_{ik} and T_{jk}, respectively. The KNN's actual formulae are as follows:

$$\mathrm{Q(Vi,\ Cm)} = \sum_{j=1}^{k} sim(Vi, Vj)\ \delta(Vi, Cm) \tag{13}$$

$$\delta(Vi, Cm) = \begin{cases} 1, & if\ Vi\ \in Cm \\ 0, & if\ Vi\ \in Cm \end{cases} \tag{14}$$

3.3.2. Random forest classifier

This algorithm creates a vast number of decision trees that work together. In this method, decision trees serve as pillars. The term "random forest" refers to a collection of decision trees whose nodes are established at the preprocessing stage (Anon, 2022c). The best feature is chosen from a random selection of features after numerous trees have been constructed. Another idea produced by the decision tree algorithm

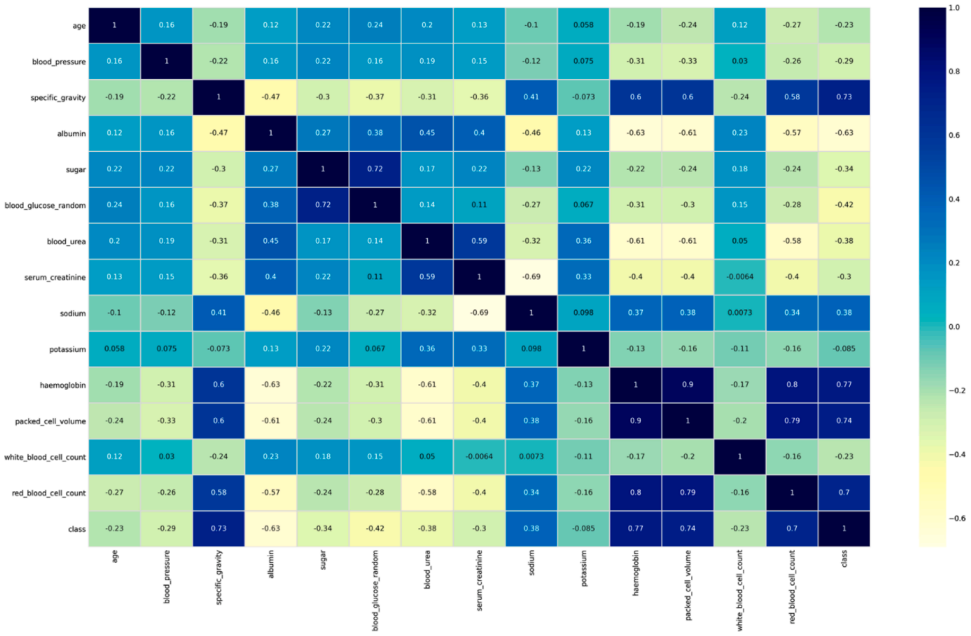

Fig. 5. Pearson correlation matrix for identifying relevant features from the dataset.

is generating a decision tree. As a result, a random forest comprises these trees, which are used to categorize new objects from a vector of inputs. Each decision tree constructed is used to classify data. If votes are assigned to that class, the random forest will select the classification with the most votes from all the trees in the forest.

The random forest classifier's mathematical formula is as follows:

$$P_{ij} = m_i T_j \; - \; m_{left(j)} T_{left(j)} \; - \; m_{right(j)} T_{right(j)} \tag{15}$$

$Pi\ sub\ (j) = node\ j^{'}s\ significance$

$m\ sub\ (j) = samples\ arriving\ at\ node\ j$

$T\ sub\ (j) = the\ value\ of\ node\ j^{'}s\ impurity$

$left\ (j) = node\ j\ splits\ a\ child\ node\ from\ the\ left$

$right\ (j) = node\ j\ splits\ a\ child\ node\ from\ the\ right$

3.3.3. Support vector machine (SVM)

Support vector machines for classification are supervised learning models that evaluate data and detect patterns in machine learning. The basic SVM is a non-probabilistic binary linear classifier that takes input data and predicts which of two possible classes will form the output for each input (Zhang, 2012). An SVM training method creates a model that allocates incoming instances to one of two categories based on a series of training examples individually labeled as belonging to one of two categories. In addition to linear classification, SVMs can efficiently conduct non-linear classification by implicitly translating their inputs into high-dimensional feature spaces, known as the kernel technique.

Mathematically, SVM can be defined where the training data for the two classes are first stacked into an n a $p \times q$ a matrix X. Here, the number of observations is denoted by p, and the number of variables is defined by q. The ith row of X is represented using x_i. Another diagonal $p \times p$ matrix Y with -1 and +1 indicates if each xi belongs to class +1 or 1. The main challenge in SVM is to partition the collection of training vectors into two distinct groups with a hyperplane.

$$D = \left\{\left(x^1, y^1\right), \cdots, \left(x^l,\ y^l\right)\right\}, x \in \mathbb{R}, y \in \{-1, 1\} \tag{16}$$

The hyperplane is considered to have separated the set of vectors optimally if it did so without mistake, and the distance between the nearest vectors to the hyperplane was maximum. Because Eq. (16) has some duplication, it is acceptable to examine a canonical hyperplane, where the parameters w and b are limited by Eq. (17).

$$(w, x) + \mathrm{b} = 0 \tag{17}$$

$$min_i \left|\left(w, x^i\right) + \mathrm{b}\right| = 1 \tag{18}$$

Eq. (19) calculates the distance d (w, b; x) between a point x and the hyperplane (w, b).

$$d(w, b; x) = \frac{\left|(w, x^i)\ + b\right|}{||w||} \tag{19}$$

According to the limitations of Eq. (18), maximizing the margin yields the ideal hyperplane. The margin is calculated as follows:

$$\begin{aligned} \rho\ (w, b) \quad &= \quad \min_{x^i : y^i = -1} d(w,\ b; x^i) + \quad \min_{x^i : y^i = 1} d(w,\ b; x^i) \\ &= \quad \min_{x^i : y^i = -1} \frac{\left|(w, x^i)\ + \mathrm{b}\right|}{||w||} + \quad \min_{x^i : y^i = 1} \frac{\left|(w, x^i)\ + \mathrm{b}\right|}{||w||} \\ &= \quad \frac{1}{||w||} \left(\min_{x^i : y^i = -1} \frac{\left|(w, x^i)\ + \mathrm{b}\right|}{||w||} + \min_{x^i : y^i = 1} \frac{\left|(w, x^i)\ + \mathrm{b}\right|}{||w||} \right) \\ &= \quad \frac{|2|}{||w||} \end{aligned} \tag{20}$$

As a result, the best hyperplane that separates the data is the one that minimizes using Eq. (21).

$$\varphi(w) = \frac{1}{2}||w||^2 \tag{21}$$

Assume that the following limit holds to see how minimizing Eq. (21) is comparable to adopting the Structural risk minimization (SRM) principle.

$$||w|| \ < A \tag{22}$$

Then, create a new equation using Eqs. (18) and (19).

$$d(w, b; x) \geq \frac{1}{A} \tag{23}$$

3.3.4. GaussianNB classifier. In the absence of identical characteristics, Naive Bayes (NBs) outperforms the complicated classification approach assuming the presence of a certain class. The NBs classifier is based on conditional probability and is built using Bayes' theorem (Puga, Krzywinski & Altman, 2015). This classifier is used as a supervised machine learning approach in medical statistical data analysis since it is very efficient on large datasets and generates useful conclusions. Eq. (24) is used to map this classifier, where the probability input hypothesis is $P(M|T)$ for a given set of data, based on the probabilities of $P(T|M)$, $P(M)$, and $P(T)$.

$$P(M|T) = \frac{P(T|M) \times P(M)}{P(T)} \tag{24}$$

$P(T|M)$ focuses on the conditional probability distribution for each input instance $T = T1,\ T2, \ldots,\ Tn$ for the response variable M. The response variable's marginal probability is denoted by P (M), and the marginal probability of an input instance is represented by $P(T)$. Eq. (25) assigns a function to the NBs classifier's class label prediction.

$$M = \arg max_M P(M) \prod_{k=1}^{n} P(T_k|M) \tag{25}$$

3.3.5. Decision tree classifier

A decision tree (DT) is where the nodes represent the input characteristics. The output is then projected based on the height information gained for that specific feature. The subtree is created by combining the characteristics not used in the over steps. In DT, Input variables are represented by internal nodes, branches represent outcomes, and classes are represented by leaf nodes (Zhao, Lee & Jeong, 2021). It comprises numerous layers of nodes, with the root node being the highest level and the leaves being the lowest.

Eq. (26) calculates the entropy of each class, while Eq. (27) calculates the entropy based on two features. Eq. (28) is used to calculate the feature-level gain G. The data features are separated sequentially, and the entropy of each branch is computed using Eq. (29) to calculate the overall proportional entropy. Here, G/I calculates the feature level's height gain ratio.

$$H(S) = \sum_{i=1}^{c} -P_i\ log_2\ P_i \tag{26}$$

$$H(S,D) = \sum_{c \in D} P(c).E(c) \tag{27}$$

$$G(S,D) = H(S) - H(S,D) \tag{28}$$

$$I(S,D) = -\sum_{j=1}^{v} \frac{D_i}{D}\ log_2 \frac{D_i}{D} \tag{29}$$

3.3.6. Multi-layer perceptron (MLP)

The Multi-layer perceptron (MLP) is a neural network where each set of inputs produces a specific vector. An output, input, and hidden layer form the MLP multilayered perceptron structure. Eq. (30) represents the input layer of MLP.

$$a^x = x \tag{30}$$

$$a^{(l+1)} = \Psi\left(w^{(l)} a^{(l)} + b^{(l)}\right) \tag{31}$$

x is the input and a^1 is the first layer of the network in Eq. (30), and the input of each layer is the weighted output of the previous layer (Pacifici, Chini & Emery, 2009) that corresponds to Eq. (31).

If l is a particular layer, $w^{(l)}$ denotes layer weight, and $b^{(l)}$ denotes the bias symbol in layer l, and Ψ denotes a non-linear function employed in this network. This function might be a sigmoid, hyperbolic tangent, or something else. Eq. (32) shows the output layer of the MLP.

$$h_{w,b}(x) = a^{(n)} \tag{32}$$

The network levels are n, the weights are w, and the bias is b. As indicated in Eq. (33), the objective function is a function that minimizes the difference between output and intended output.

$$\mathrm{J(w, b; x, y)} = \frac{1}{2}\left|\left|h_{(w,b)}(x) - y\right|\right|^2 \tag{33}$$

3.3.7. Bagging classifier

Bagging is a technique for enhancing machine learning classification algorithms' performance. Leo Breiman created this approach, which gets its name from "bootstrap aggregating" (Breiman, 1996). A classification algorithm develops a classifier $H:\ D \rightarrow \{-1, 1\}$ on the basis of a training set of example descriptions D in the case of classification into two potential classes. In response to changes in the training set, the bagging approach generates a series of classifiers H_m, $m = 1, \ldots, M$. A compound classifier is created by combining these classifiers. In Eq. (34), the compound classifier prediction is given as a weighted aggregate of individual classifier predictions.

$$H(d_i) = \sin\left(\sum_{m=1}^{M} \alpha_m\ H_m(d_i)\right) \tag{34}$$

Eq. (34)'s meaning might be interpreted as a voting process. An example d_i is assigned to the class that receives the greatest number of votes from the classifiers. The parameters $\alpha_m,\ m = 1, \ldots, M$ are chosen so that more precise classifiers have a greater influence on the result prediction than less accurate classifiers.

3.3.8. Gradient boosting classifier

The key concept behind gradient boosting is considering the boosting process to optimize a set of cost functions. Consequently, the loss function minimization gradient descent routes may be translated into decision trees acquired sequentially to enhance the classifier (Friedman, 2001). The goal of gradient boosting is to learn a function $\widehat{F}$ from all viable hypotheses H while reducing the expectation of loss over the distribution D, given a training dataset with each data point $(x,\ y) \sim D$. Here, Eq. (35) helps to calculate $\widehat{F}$ (Arcing the Edge | Department of Statistics, 2022)

$$\widehat{F} = \underset{F \in \mathscr{H}}{\operatorname{argmin}}\ \underset{(x,Y)\ \sim D}{\mathrm{E}} L(y, F(x)) \tag{35}$$

In Eq. (36), $L(y, F(x))$ defines the prediction loss of $F(x)$ to the label y. $F_k\ (x)$ denotes the learned model in the k^{th} iteration, whereas $h_k\ (x)$ denotes the decision tree acquired as the descent direction in the k^{th}iteration based on the model already obtained $F_k\ (x)$.

$$F_{k+1}\ (x) \leftarrow F_k(x) + \alpha_k.h_k(x) \tag{36}$$

3.3.9. AdaBoost classifier

The AdaBoost method was employed in this study to create a robust classifier using a training strategy (sometimes called weak learning). Weak learning aimed to find the weak classifier that could best distinguish between positive and negative samples. Weak learning established the ideal threshold value for each feature, ensuring that the least number of examples are misclassified Wyner, Olson, Bleich & Mease, 2017). Weighting vector w_n is calculated using the Eq. (39). Eqs. (37) and ((38) are required to calculate value of w_n. In Eq. (37), $G_i\ (x_i)$ denotes the weak classifier and w_n is initiated as the observation weights, where $i = 1,\ 2,\ \ldots,\ N$. In Eq. (40), $f_i(x)$ is calculated using the value of weak classifier and Eq. (38). The final classifier output is f shown in Eq. (41) is a sign of weighted linear combination of classifiers generated at each stage of the procedure.

$$err_m = \frac{\sum_{i=1}^{N} w_i I\ (y_i \neq G_i\ (x_i))}{\sum_{i=1}^{N} w_i} \tag{37}$$

$$a_m = \log\left(\frac{1 - err_t}{err_t}\right) \tag{38}$$

$$w_i \leftarrow w_i\ .\ \exp(a_t .\ I\ (y_i \neq G_t\ (x_i))) \tag{39}$$

$$f_i(x) = \sum_{m=1}^{M} a_m\ G_m\ (x) \tag{40}$$

$$f = sign(f_M(x)) \tag{41}$$

3.3.10. CatBoost classifier

The CatBoost classifier is another machine learning technique that predicts category features. CatBoost is a gradient boosting method that uses binary decision trees as basic predictors (Prokhorenkova, Gusev, Vorobev, Dorogush & Gulin, 2018). Assume data with samples $D = (Xj, yj)\ j = 1, \ldots, m$, where $Xj = (x1\ j, x2\ j, .., xn\ j)$ is a vector of n characteristics and response feature $y_j \in \mathbb{R}$ is a binary (yes or no) or numerical feature (0 or 1). The samples (Xj, yj) are dispersed independently and identically according to an unknown distribution p (.,.). The learning task's purpose is to develop a function $H : \mathbb{R}^n \rightarrow \mathbb{R}$ that minimizes the predicted loss provided in Eq. (42).

$$\mathscr{L}(H) := \psi L(y,\ H(X)) \tag{42}$$

Here, the smooth loss function is denoted by $L(.,.)$, and from the training data D, the testing data sampled is represented by$(X,\ y)$.

3.3.11. Extra trees classifier

According to the standard top-down process, the Extra-Trees technique creates an ensemble of unpruned decisions or regression trees. Its two primary distinctions from previous tree-based ensemble approaches are dividing nodes at random and growing trees using the whole learning sample (rather than a bootstrap replica) (Geurts, Ernst & Wehenkel, 2006). Algorithm 1 shows the algorithm of extra-trees splitting for the numerical attributes. It contains two parameters: T, which is the number of randomly picked characteristics at each node, and p_{min} is the minimum sample size for splitting a node. The trees' predictions are combined to produce the final prediction, selected by a majority vote in classification tasks and an arithmetic average in regression problems. The splitting node operation is performed by ***Split_node (E)*** function, and a split value is obtained by the function ***pick_random_split* (E, *b*).** The final result is obtained by the ***Stop_split (E)*** function.

3.3.12. LightGBM classifier

Chen et al. (Chen & Guestrin, 2016) presented XGBoost, a scalable tree boosting engine. Although XGBoost achieved great accuracy, the LightGBM ensemble was more resilient, computationally efficient, and time-efficient (Daoud, 2019). Let, $X = (x_i, y_i)$ is a given dataset with features x and label y. Considering Γ as a loss function and F_0 as an initial fit optimization goal is obtained by Eq. (43).

$$\widehat{F} = \arg\min_{F} minE_{x,y}[\Gamma(y, F(x)] \tag{43}$$

Eq. (44) gives the pseudo residuals or gradient a_m for the m^{th} iteration, to which the decision tree $hm(x)$ is fitted.

$$a_m = -\frac{\partial \Gamma(y_i,\ \widehat{F})}{\partial F} \tag{44}$$

The iterative criterion for GBDT to obtain a new boosted fit to minimize the loss function is given in Eq. (45), Where, λ_m is a multiplier that operates as step size and is optimized using a linear search algorithm. λ_m can be obtained using Eq. (46).

$$F_m(x) = F_{m-1}(x) + \lambda_m h_m\ (x) \tag{45}$$

$$\lambda_m = arg_\lambda \min \sum_{i=i}^{N} \Gamma(y_i,\ F_{m-1}(x_i) + \lambda h_m\ (x_i)) \tag{46}$$

3.3.13. XGBoost classifier

Extreme Gradient Boosting (XGBoost) is a boosted tree technique that uses the gradient boosting idea (Friedman, 2001). Compared to previous gradient boosting methods, XGBoost employs a more regularized model formalisation to prevent data over-fitting, resulting in improved performance (Punnoose & Ajit, 2016). Let, D a tree ensemble model uses L additive functions to predict the output given data with m-samples and n-features, as shown in Eq. (47). H is a space of regression trees that obtains using Eq. (48).

$$\widehat{y_j} = \varnothing\left(X_j\right) = \sum_{l=1}^{L} h_l\ \left(X_j\right),\ h_l\ \in H \tag{47}$$

$$H = \left\{h(X) = \omega_q(X)\right\}\left(q : \mathbb{R}^n \rightarrow U,\ \omega \in \mathbb{R}^U\right) \tag{48}$$

The regularized objective is minimized by Eq. (49) to learn the set of functions employed in the model. Where l is a differentiable variable to measure the difference between the targets $\widehat{y_j}$ and anticipatedy_j. To avoid over-fitting, Ω penalizes the model's complexity. The model is taught in an additive method. As shown in Eq. (50), a score to quantify the quality of a particular tree structure q is calculated. To obtain $\Gamma^{(u)}(q)$, f_i and g_i are calculated using Eqs. (51) and Eq. (52), respectively.

$$\Gamma(\phi) = \sum_{j} l(\widehat{y_j},\ y_j)\ + \sum_{l} \Omega(h_l),\ \Omega(h) = \gamma U + \frac{1}{2}\lambda||\omega||^2 \tag{49}$$

$$\Gamma^{(u)}(q) = -\frac{1}{2} \sum_{j=1}^{U} \frac{\left(\sum_{i=I_j} f_i\right)^2}{\sum_{i=I_j} g_i + \lambda} + \gamma U \tag{50}$$

$$f_i = \partial_{\widehat{y}^{(u-1)}}\ l\left(y_i,\ \widehat{y}^{(u-1)}\right) \tag{51}$$

$$g_j = \partial^2_{\widehat{y}^{(u-1)}} i\left(y_i,\ \widehat{y}^{(u-1)}\right) \tag{52}$$

4. Experimental results

This section highlights the relevant results in detail obtained from the experiment. After collecting it, the dataset has been shuffled and split into training and testing sets. Various data preparation strategies were fitted only on the training set, then applied to both the training and testing sets to avoid data leaking and over-fitting. For obtaining the results, different environments have been utilized in this research. Table 6 highlights the environment setup required for the experiment. All the techniques have been performed in the local machine using Jupyter Notebook in Python, including libraries or modules such as Scikit Learn, NumPy, Pandas, Seaborn, and Matplotlib.

Furthermore, different evaluation metrics were employed to evaluate the performance of the available ML classifier. The following Eqs.

Table 6
System Specifications.

System Specifications for the Machine Learning Model Development	
RAM	16GB
CPU	1 × single core hyper threaded; Xeon processors @2.3GHz
Cache	46MB
GPU	NVidia K80 GPU
GPU Memory	16GB
Session Limit	10 h
Disk Space	100 GB

(53)–(56) present the procedure of calculating other evaluation metrics of the Confusion Matrix. Accuracy (AC), Precision (P), Recall (R), and F1-score are extracted by computing correctly classified samples (True positive (TP) and True Negative (TN)) and the incorrectly classified samples (False positive (FP) and False Negative (FN)).

$$AC = \frac{TN + TP}{TN + TP + FN + FP} \times 100\% \tag{53}$$

$$P = \frac{TP}{TP + FP} \times 100\% \tag{54}$$

$$R = \frac{TP}{TN + TP + FN + FP} \times 100\% \tag{55}$$

$$F1\ score = 2 \times \frac{P \times R}{P + R} \times 100\% \tag{56}$$

Fig. 6 illustrates the accuracy obtained in detecting the CKD and non-CKD classes based on the dataset. The dataset is split into 70% training data and 30% test data for experimenting. Precision, Recall, and F1-score are also computed and presented in Table 7. Among 14 machine learning classifiers, the Extra Trees model produces the highest accuracy of 98% with 99% precision, 98% recall, and 98% F1-score. From 120 (30%) test data, the Extra Trees class classified 60% of data as non-CKD and 40% as CKD. While the Extra Tress classifier classified the True positive samples as 99% with only a 1% of error rate Ada Boost, Random Forest, and SVM identified the positive samples as 98%, 97%, and 98% with 2%, 3%, and 2% error rate respectively. CatBoost and XGBoost achieved an accuracy of 96% after Ada Boost, Random Forest, and SVM with true negative values of 5% each. After that, 95% and 94% accuracy were produced in the exploration by the XGBoost and Decision Tree classifier with an actual positive sample detection rate of 95% and 94%.

With 93% accuracy, Gaussian NB and Stochastic Gradient Boosting come in fifth place in the highest accuracy achievement. However, Gaussian NB produced 93% precision, recall, and F1-score, while SGB obtained 94%, 92%, and 93%, respectively. The dataset is also tested with the Back propagation-based multi-layer perceptron architecture, and less than 90% accuracy is obtained from the MLP with the same precision, recall, and F1-score. Surprisingly, KNN and Bagging classifier produced the lowest accuracy of 67% and 60%, respectively, and 23% and 50% error rates.

Table 7
Performance metrics of the machine learning model across the dataset.

ML Classifier	Precision	Recall	F1-score
Ada Boost Classifier	98%	96%	97%
Bagging Classifier	30.00%	50.00%	37.00%
CatBoost Classifier	96%	95%	96%
Decision Tree Classifier	94.00%	93.00%	94.00%
Extra Trees Classifier	99%	98%	98%
Gaussian Naive Bayes (Gaussian NB)	93.00%	93.00%	93.00%
Gradient Boosting Classifier	95.00%	88.00%	93.00%
K-nearest neighbors	66.00%	67.00%	66.00%
LightGBM Classifier	96%	95%	96%
Multi-Layer Perceptron	89.00%	89.00%	89.00%
Random Forest Classifier	97.00%	96.00%	96.00%
Stochastic Gradient Boosting	94%	92%	93%
Support vector machines (SVM)	98.00%	97.00%	97.00%
XGBoost	95%	94%	95%

Finally, based on the results, the Extra Trees Classifier obtained the highest accuracy (98%), and the Bagging classifier obtained the lowest accuracy (60%). Chronic Kidney Disease (CKD) is a life-long disease characterized by impaired kidney function and reduced quality of life for patients. Patients, healthcare providers, and the government all bear a significant financial burden from CKD. Renal replacement therapy (RRT) for end-stage renal disease (ESRD) is expensive or complicated. Patients with such disease undergo hemodialysis, and peritoneal dialysis or sometimes transplantation is needed. From a public health standpoint, it is critical to predict CKD prevalence trends so that decision-makers (ministries, insurers, and hospital administrators) may make appropriate measures to prevent a potential increase in the number of patients. Machine-learning techniques are increasingly being used in healthcare to construct disease prediction models, thanks to the availability of biomedical laboratory data. In this study, among the 14 conventional machine learning models, the Extra Trees classifier outperformed other models in terms of accuracy and various performance metrics on the Chronic Kidney Dataset available at the UCI Machine Learning repository.

Furthermore, this study also explored the complexity values of the

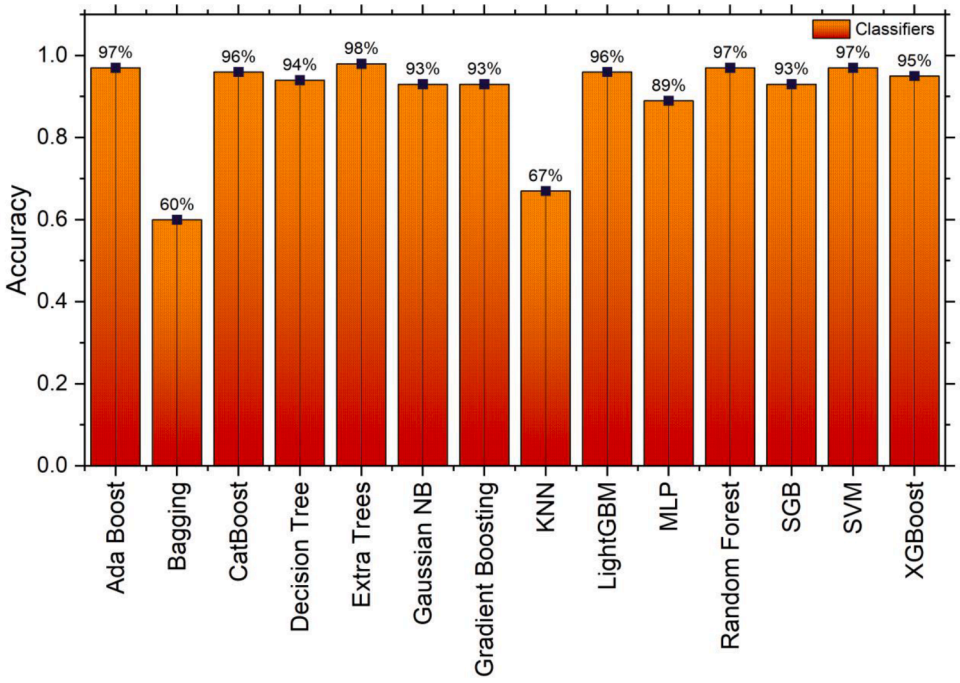

Fig. 6. Overall classification accuracy percentage for all used algorithms.

Extra Trees classifier and the time complexity of building an Extra-Trees is $O(v * n\ log(n))$, where the number of attributes is denoted by v, and the number of records is denoted by n. The outcome of this study provides a significant decision as the Extra Trees classifier usually is not a well-known predictor in health informatics. According to the available literature on CKD diagnosis, KNN, Decision Tree, SVM, and Random Forest have surpassed others in terms of performance metrics in most scenarios. One of the most notable findings from this study was adopting a hybrid feature selection strategy. Previously, the Recursive Feature Elimination (RFE) technique was used in most related studies.

Moreover, stepwise data preparation procedures were also used in this proposed work with the hybrid (Chi^2-MI) feature selection method to reconstruct the dataset for use in the models. Encoding, normalization, missing value imputation, outlier removal, imbalance handling, feature selection, and standardizations are all steps in the data preparation process that assist this exploration to achieve over 90% accuracy from 12 different ML models and over 60% accuracy from the other two models with top 85% (13) correlated features. Therefore, the accuracy range of this study is between 98% and 60%. The proposed model is tested with only a single dataset with 24 features from 400 patients. In Table 8, a comparison of the proposed system with existing approaches is presented and analyzed. According to the comparison table, some earlier studies obtained marginally higher accuracy than the proposed approach, but only a few studies adopted a feature section approach (mostly Recursive Feature Elimination) while evaluating CKD data. However, no previous research proposed a reproducible hybrid wrapper feature selection algorithm that can be applied to another similar health dataset to diagnose any similar disease. In addition, selecting appropriate and meaningful features can lead to a higher diagnostic accuracy with a minimum error rate.

Table 8
Comparative analysis of the performance of the proposed model with previous studies.

Reference	Accuracy	Precision	Recall	F-1 Score	Applied Feature Selection
Ahmed et al. (Ahmed et al., 2014)	86.7%	–	–	–	No
Khamparia et al. (Khamparia et al., 2020)	100%	100%	100%	100%	No
Kim et al. (Kim & Ye, 2021)	95.4%	–	–	–	No
Almansour et al. (Almansour et al., 2019)	99.75%	100%	99.6%	99.7%	No
Rady et al. (Rady & Anwar, 2019)	96.70%	98.75%	100.0%	99.37%	No
Kunwar et al. (Kunwar et al., 2016)	100%	–	–	–	No
Wibawa et al. (Wibawa et al., 2017)	98.10%	98.10%	98.0%	98.0%	Yes
Avci et al. (Avci et al., 2018)	99.00%	98%	99%	98%	No
Chiu et al. (Chiu et al., 2013)	91.71%	–	–	–	No
Sara *et al.* (Sara & Kalaiselvi, 2018)	90%	95.24%	90.91%	93.02%	Yes
Senan et al. (Senan et al., 2021)	100%	100%	100%	100%	Yes
Krishnamurthy et al. (Krishnamurthy et al., 2021)	90%	70%	88.2%	78.1%	No
Our proposed model (Chi^2-MI + Extra Trees)	**98%**	**99%**	**98%**	**99%**	**Yes**

Table 9
Extra-Trees splitting algorithm .

Split_node(E)
Input: E, the local learning subset corresponding to the split node
Output: nothing or a split $[b < b_c]$
- if stop_split(E) is true
  return nothing
- else
  Select T attributes $\{b_1, b_2, \ldots, b_T\}$ among all candidate attributes that are not constant (in E);
- Draw T splits $\{e_1, e_2, \ldots, e_T\}$, e_i = pick_random_split(E, b_i), $i = 1, 2, \ldots, T$
- Return e_* when $Score(e_*, E) = max_{i=1,2\ldots,T} Score(e_i, E)$

pick_random_split(E, b)
Inputs: attribute b and subset (E)
Output: a split
- Let the minimal and maximum value of b in E are b^E_{min}, b^E_{max}, respectively.
- Uniformly in $[b^E_{min}, b^E_{max}]$ a random cut-point (b_c) is drawn.
- Return the split $[b < b_c]$

Stop_split (E)
Input: a subset (E)
Output: 0 or 1 (Boolean)
- If $|E| < p_{min}$
  Return True
- If all of E's attributes are constant,
  Return True
- If the output in E is constant,
  Return True
- Else
  Return False

5. Conclusion

This study emphasizes the capacity of machine learning algorithms to detect CKD using the fewest possible tests or features. For detecting CKD at an early stage, this research employed Machine Learning algorithms like Ada Boost, Bagging classifier, CatBoost, Decision Tree, Extra Trees, Gaussian Nave Bayes, K-nearest neighbor, Gradient Boosting, LightGBM, Multi-Layer Perceptron, Random Forest, Stochastic Gradient Boosting, Support Vector Machine, and XGBoost. A hybrid wrapper feature selection method (Chi2-MI) has been applied to the whole dataset, and the most impactful features based on correlation scores are selected to predict the CKD. The proposed hybrid feature selection approach performed well with the Extra Tress classifier and produced a higher accuracy score by outperforming other algorithms. Based on the outcome of this research, these ML-based models can be effectively employed in resource development and undertaking public health efforts such as close monitoring of patients and early identification of CKD. The future scope of this research would be to develop a web application to assist the clinician in diagnosing kidney failure patients in real time. Also, it is intended to build a dataset containing the images of ultrasound of CKD patients and apply deep learning methods to detect CKD.

ORCiD information

Author Name	ORCiD Information
Samrat Kumar Dey	0000–0002–7999–8576
Khandaker Mohammad Mohi Uddin	0000–0002–5401–0437
Hafiz Md. Hasan Babu	0000–0002–2325–2986
Md. Mahbubur Rahman	0000–0001–6525–2274
Arpita Howlader	0000–0001–5413–2277
K. M. Aslam Uddin	N/A

Data availability

The data supporting this study's findings are available from the corresponding author upon reasonable request.

Funding

None.

Ethical approval

Not required.

Consent to participate

Not required.

CRediT authorship contribution statement

Samrat Kumar Dey: Conceptualization, Methodology, Software, Resources, Writing – review & editing, Visualization. **Khandaker Mohammad Mohi Uddin:** Data curation, Writing – original draft. **Hafiz Md. Hasan Babu:** Validation, Supervision. **Md. Mahbubur Rahman:** Supervision. **Arpita Howlader:** Writing – review & editing, Visualization. **K.M. Aslam Uddin:** Writing – review & editing.

Declaration of Competing Interests

The authors declare that there are no conflicts of interest.

Data Availability

Data will be made available on request.

References

Short Text Classification Using Semantic Random Forest, Springerprofessional.De. (n.d.). https://www.springerprofessional.de/en/short-text-classification-using-semantic-random-forest/2233656 (accessed February 2, 2022).

sklearn.preprocessing.MinMaxScaler, Scikit-Learn. (n.d.). https://scikit-learn/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html (accessed February 4, 2022).

UCI Machine Learning Repository: Chronic_Kidney_Disease Data Set, (n.d.). https://archive.ics.uci.edu/ml/datasets/chronic_kidney_disease (accessed February 2, 2022).

Ahmed, S., Kabir, M. T., Mahmood, N. T., & Rahman, R. M. (2014). Diagnosis of kidney disease using fuzzy expert system. In *The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014)* (pp. 1–8). https://doi.org/10.1109/SKIMA.2014.7083522

Aldhyani, T. H. H., Alshebami, A. S., & Alzahrani, M. Y. (2020). Soft clustering for enhancing the diagnosis of chronic diseases over machine learning algorithms. *Journal of Healthcare Engineering, 2020*, Article e4984967. https://doi.org/10.1155/2020/4984967

Aljaaf, A. J., Al-Jumeily, D., Haglan, H. M., Alloghani, M., Baker, T., Hussain, A. J., et al. (2018). Early prediction of chronic kidney disease using machine learning supported by predictive analytics. In *2018 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1–9). https://doi.org/10.1109/CEC.2018.8477876

Almansour, N. A.., Syed, H. F., Khayat, N. R., Altheeb, R. K., Juri, R. E., Alhiyafi, J., … Olatunji, S. O. (2019). Neural network and support vector machine for the prediction of chronic kidney disease: A comparative study. *Computers in Biology and Medicine, 109*, 101–111. https://doi.org/10.1016/j.compbiomed.2019.04.017

Anon, Estimated Glomerular Filtration Rate (eGFR), National Kidney Foundation. (2015). https://www.kidney.org/atoz/content/gfr (accessed February 4, 2022).

Arcing the Edge | Department of Statistics, (n.d.). https://statistics.berkeley.edu/tech-reports/486 (accessed February 4, 2022).

Avci, E., Karakus, S., Ozmen, O., & Avci, D. (2018). Performance comparison of some classifiers on Chronic Kidney Disease data. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)* (pp. 1–4). https://doi.org/10.1109/ISDFS.2018.8355392

Bikbov, B., Perico, N., Remuzzi, G., & O. behalf of the GBD Genitourinary Diseases Expert Group. (2018). Disparities in chronic kidney disease prevalence among males and females in 195 countries: Analysis of the Global Burden of Disease 2016 Study. *NEF, 139*, 313–318. https://doi.org/10.1159/000489897

Breiman, L. (1996). Bagging predictors. *Machine learning, 24*, 123–140. https://doi.org/10.1007/BF00058655

Chatzigeorgakidis, G., Karagiorgou, S., Athanasiou, S., & Skiadopoulos, S. (2018). FML-kNN: Scalable machine learning on Big Data using k-nearest neighbor joins. *Journal of Big Data, 5*, 4. https://doi.org/10.1186/s40537-018-0115-x

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). Association for Computing Machinery. https://doi.org/10.1145/2939672.2939785.

Chen, Z., Zhang, X., & Zhang, Z. (2016). Clinical risk assessment of patients with chronic kidney disease by using clinical data and multivariate models. *International urology and nephrology, 48*, 2069–2075. https://doi.org/10.1007/s11255-016-1346-4

Chiu, R. K., Chen, R. Y., Wang, S.-A., Chang, Y.-C., & Chen, L.-C. (2013). Intelligent systems developed for the early detection of chronic kidney disease. *Advances in Artificial Neural Systems, 2013*, Article e539570. https://doi.org/10.1155/2013/539570

Daoud, E. A. (2019). Comparison between XGBoost, LightGBM and CatBoost using a home credit dataset. *International Journal of Computer and Information Engineering, 13*, 6–10.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics, 29*, 1189–1232.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning, 63*, 3–42. https://doi.org/10.1007/s10994-006-6226-1

Imran, A. A., Amin, M. N., & Johora, F. T. (2018). Classification of Chronic Kidney Disease using Logistic Regression, Feedforward Neural Network and Wide amp; Deep Learning. In *2018 International Conference on Innovation in Engineering and Technology (ICIET)* (pp. 1–6). https://doi.org/10.1109/CIET.2018.8660844

Khamparia, A., Saini, G., Pandey, B., Tiwari, S., Gupta, D., & Khanna, A. (2020). KDSAE: Chronic kidney disease classification with multimedia data learning using deep stacked autoencoder network. *Multimedia Tools and Applications, 79*, 35425–35440. https://doi.org/10.1007/s11042-019-07839-z

Kim, D.-H., & Ye, S.-Y. (2021). Classification of chronic kidney disease in sonography using the GLCM and artificial neural network. *Diagnostics, 11*, 864. https://doi.org/10.3390/diagnostics11050864

Krishnamurthy, S., Ks, K., Dovgan, E., Luštrek, M., Gradišek Piletič, B., Srinivasan, K., … Syed-Abdul, S. (2021). Machine learning prediction models for chronic kidney disease using national health insurance claim data in Taiwan. *Healthcare, 9*, 546. https://doi.org/10.3390/healthcare9050546

Kunwar, V., Chandel, K., Sabitha, A. S., & Bansal, A. (2016). Chronic Kidney Disease analysis using data mining classification techniques. In *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)* (pp. 300–305). https://doi.org/10.1109/CONFLUENCE.2016.7508132

Pacifici, F., Chini, M., & Emery, W. J. (2009). A neural network approach using multi-scale textural metrics from very high-resolution panchromatic imagery for urban land-use classification. *Remote Sensing of Environment, 113*, 1276–1292. https://doi.org/10.1016/j.rse.2009.02.014

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased boosting with categorical features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (pp. 6639–6649). Curran Associates Inc.

Puga, J. L., Krzywinski, M., & Altman, N. (2015). Bayes' theorem. *Nature Methods, 12*, 277–278. https://doi.org/10.1038/nmeth.3335

Punnoose, R., & Ajit, P. (2016). Prediction of employee turnover in organizations using machine learning algorithms. *International Journal of Advanced Research in Artificial Intelligence (IJARAI), 5*. https://doi.org/10.14569/IJARAI.2016.050904

Qin, J., Chen, L., Liu, Y., Liu, C., Feng, C., & Chen, B. (2020). A machine learning methodology for diagnosing chronic kidney disease. *IEEE access : practical innovations, open solutions, 8*, 20991–21002. https://doi.org/10.1109/ACCESS.2019.2963053

Rady, E.-H. A., & Anwar, A. S. (2019). Prediction of kidney disease stages using data mining algorithms. *Informatics in Medicine Unlocked, 15*, Article 100178. https://doi.org/10.1016/j.imu.2019.100178

Sar, S. A. B. V. J., & Kalaiselvi, K. (2018). Ensemble swarm behaviour based feature selection and support vector machine classifier for chronic kidney disease prediction. *International Journal of Engineering & Technology, 7*, 190–195. https://doi.org/10.14419/ijet.v7i2.31.13438

Sena, E. M. N., Al-Adhaileh, M. H., Alsaade, F. W., Aldhyani, T. H. H., Alqarni, A. A., Alsharif, N., … Alzahrani, M. Y. (2021). Diagnosis of chronic kidney disease using effective classification algorithms and recursive feature elimination techniques. *Journal of Healthcare Engineering, 2021*, Article e1004767. https://doi.org/10.1155/2021/1004767

Wibawa, M. S., Maysanjaya, I. M. D., & Putra, I. M. A. W. (2017). Boosted classifier and features selection for enhancing chronic kidney disease diagnose. In *2017 5th International Conference on Cyber and IT Service Management (CITSM)* (pp. 1–6). https://doi.org/10.1109/CITSM.2017.8089245

World Health Organization. (2005). *Preventing chronic diseases : A vital investment : Who global report.* World Health Organization. https://apps.who.int/iris/handle/10665/43314 accessed February 4, 2022.

Wyner, A. J., Olson, M., Bleich, J., & Mease, D. (2017). Explaining the success of adaboost and random forests as interpolating classifiers. *The Journal of Machine Learning Research, 18*, 1558–1590.

Zhang, Y. (2012). Support vector machine classification algorithm and its application. In C. Liu, L. Wang, & A. Yang (Eds.), *Information computing and applications* (pp. 179–186). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-34041-3_27.

Zhao, L., Lee, S., & Jeong, S.-P. (2021). Decision tree application to classification problems with boosting algorithm. *Electronics, 10*, 1903. https://doi.org/10.3390/electronics10161903