Chronic Kidney Disease Prediction using Artificial Neural Networks

Artificial Intelligence Project Report

Submitted by
Sudeepthi Rebbalapalli (17BCE0955)
Aishwarya Gajagowni (17BCE2054)

Semester: FALL 2019-2020

Chronic Kidney Disease Prediction using Artificial Neural Networks

Sudeepthi Rebbalapalli, *17BCE0955*, Aishwarya Gajagowni, *17BCE2054*

***Abstract*— Early detection and characterization are considered to be critical factors in the management and control of chronic kidney disease. The use of efficient data mining and machine learning techniques is shown to reveal and extract hidden information from clinical and laboratory patient data, which can be helpful to assist physicians in maximizing accuracy for identification of disease severity stage. In this project, we make use of Artificial Neural Network approach to detect kidney disease. We show that feature selection approach which filters out less important attributes is well suited for chronic kidney disease prediction. We will also perform a prediction on an unclassified set of data with the classification model developed. The dataset has been collected from UCI Machine Learning Repository.**

***Index Terms*—Artificial Neural Networks, Chronic Kidney Disease, Classification**

I. Introduction

Chronic kidney disease includes conditions that damage your kidneys and decrease their ability to keep you healthy. If kidney disease gets worse, wastes can build to high levels in your blood and make you feel sick. You may develop complications like high blood pressure, anemia (low blood count), weak bones, poor nutritional health and nerve damage. Also, kidney disease increases your risk of having heart and blood vessel disease. Therefore, early diagnosis and detection of this disease can help the patients in recovery on the right time. Data mining and machine learning can be used as an informative tool to extract the useful information which helps pathologists and doctors in prompt decisions making. Today's some researchers are working on CKD by applying different computational techniques for the prediction and diagnosis of this disease. We are making use of classification which is a machine learning technique used to predict group membership for data instances. Classification is comparable to clustering in that it also segment information retrieval into distinct segment called classes. In order to predict the outcome, the algorithm processes a training set containing a set of attributes and their respective outcome, usually called goal or prediction attribute. Here we are making use of ANN algorithm for classification.

II. Literature Survey

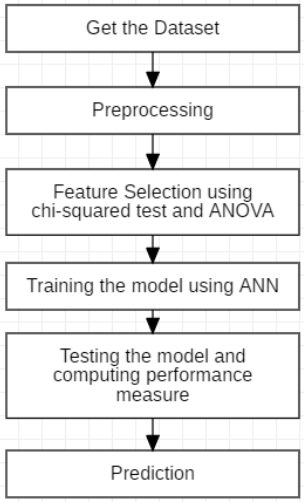
Mohamed Elhoseny, et al. (2019) introduce an intelligent prediction and classification system for healthcare, namely Density based Feature Selection (DFS) with Ant Colony based Optimization (D-ACO) algorithm for chronic kidney disease (CKD). The proposed intelligent system eliminates irrelevant or redundant features by DFS in prior to the ACO based classifier construction. The proposed D-ACO framework three phases namely: Pre-processing: The pre-processing stage is the primary process since the database may contain redundant and noise data. By examining the data, different processes take place such as data cleaning, filling missing values, removing excessive data because the missing values and excessive data degrade the performance. Feature Selection (FS): Here, DFS is used which selects a group of features in every iteration. The DFS method is a heuristic approach used to evaluate the merits of features. Classification: For classification task, the ACO algorithm is employed for the extraction of classification rules, using the behaviour of ant colonies and data mining techniques. Furthermore, the D-ACO algorithm is tested using benchmark CKD dataset from the UCI repository is used and the performance are investigated based on different evaluation factors. The improved results of D-ACO algorithm are due to the addition of DFS which removes the undesirable features to enhance the classifier results. Comparing the D-ACO algorithm with existing methods, the presented intelligent system outperformed the other methodologies with a significant improvisation in classification accuracy using fewer features.

Ahmad, Mubarik, et al. (2017) talk about a system was developed based on machine learning technique, Support Vector Machine (SVM). The methodology of this study consists of two main phases: Classification Modelling and System Development. Classification modelling consists of: Data Collection: Dataset was collected from UCI Machine Learning entitled "Chronic Kidney Disease". Data Preparation: In this stage, there are two sub-stages: attribute selection and data cleansing. Data Grouping: In this stage, the dataset is divided into two groups: training and testing. Classification and Rules Extraction are done with the help of packages in R. System development was based on the extracted rules before. Rules are implemented into a system using programming language R. They made use of Random Forest package in order to calculate the error rate of the system. This study resulted in a system that can detect a chronic condition of kidney disease based on several factors with an accuracy of 98.34%.

Chakrapani, et al. (2019) propose a method based on deep neural network, with back propagation algorithm. Datasets are firstly pre-processed by data mining statistical techniques to fill the missing values of the dataset using mean, mode and median of attributes. After pre-processing of the dataset, data is divided into two sections i.e. training and testing. Further, training data with their known target classes is used for the training of the classifier, and after the training of classifier separate test data is fed into trained classifier. This trained classifier detects the class of sample query. For the performance analysis of different classifiers same process are repeated. From the comparative analysis with other variants of classifiers like SVM, K-NN, Classification and Regression tree it is found that the recognition accuracy of ANN is significantly encouraging. Therefore, we can use this framework for the better prediction of chronic kidney disease. This study makes use of the dataset from the UCI Machine Learning Repository named Chronic Kidney Disease uploaded in 2015.

III. Methodology

The literature would be studied in detail on Artificial Neural Networks in order to understand the workflow of classification using Artificial neural networks. The proposed methodology will be implemented using Jupyter Notebook in the Anaconda Distribution.

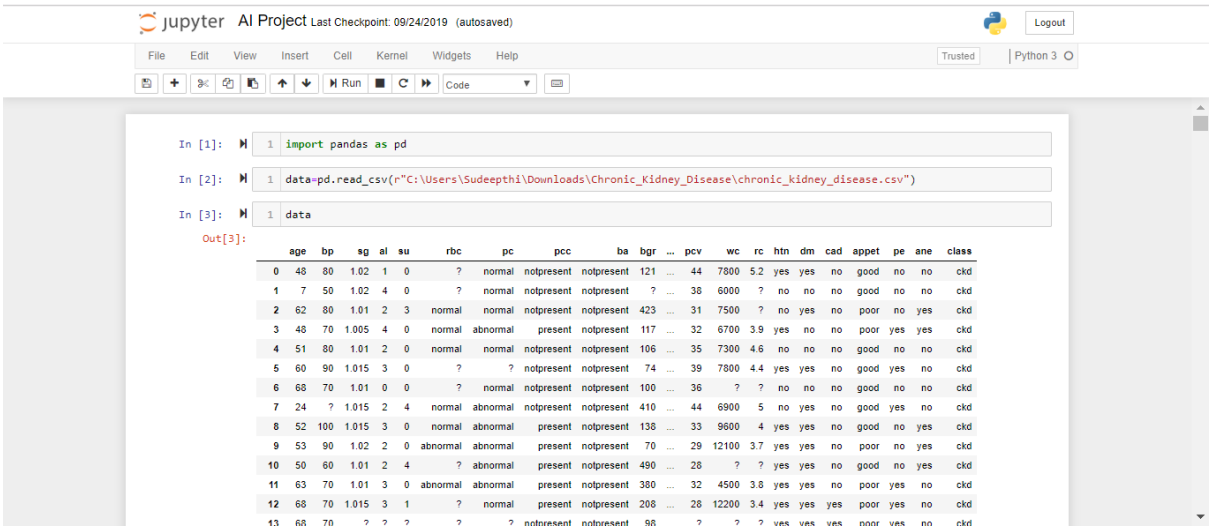


Flowchat of proposed methodology

The dataset from the UCI repository will first be preprocessed to remove the null values by replacing with the most suitable measures of centrality for each feature. The data types of the data will also be adjusted for prediction. Then the feature selection will take place by using chi-squared test on the categorical data with the categorical output and ANOVA test on the continuous data with the categorical output. After feature selection, the remaining features will be used to train and test the classification model using ANN algorithm and the performance measures are computed to determine the quality of the model. Finally, we will use the model on some unclassified data to perform a prediction.

IV. Experimentations and Results

Experimentation



	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	class
0	48	80	1.02	1	0	?	normal	notpresent	notpresent	121	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	7	50	1.02	4	0	?	normal	notpresent	notpresent	?	...	38	6000	?	no	no	no	good	no	no	ckd
2	62	80	1.01	2	3	normal	normal	notpresent	notpresent	423	...	31	7500	?	no	yes	no	poor	no	yes	ckd
3	48	70	1.005	4	0	normal	abnormal	present	notpresent	117	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	51	80	1.01	2	0	normal	normal	notpresent	notpresent	106	...	35	7300	4.6	no	no	no	good	no	no	ckd
5	60	90	1.015	3	0	?	?	notpresent	notpresent	74	...	39	7800	4.4	yes	yes	no	good	yes	no	ckd
6	68	70	1.01	0	0	?	normal	notpresent	notpresent	100	...	36	?	?	no	no	no	good	no	no	ckd
7	24	?	1.015	2	4	normal	abnormal	notpresent	notpresent	410	...	44	6900	5	no	yes	no	good	yes	no	ckd
8	52	100	1.015	3	0	normal	abnormal	present	notpresent	138	...	33	9600	4	yes	yes	no	good	no	yes	ckd
9	53	90	1.02	2	0	abnormal	abnormal	present	notpresent	70	...	29	12100	3.7	yes	yes	no	poor	no	yes	ckd
10	50	60	1.01	2	4	?	abnormal	present	notpresent	490	...	28	?	?	yes	yes	no	good	no	yes	ckd
11	63	70	1.01	3	0	abnormal	abnormal	present	notpresent	380	...	32	4500	3.8	yes	yes	no	poor	yes	no	ckd
12	68	70	1.015	3	1	?	normal	present	notpresent	208	...	28	12200	3.4	yes	yes	yes	poor	yes	no	ckd
13	68	70	?	?	?	?	?	notpresent	notpresent	98	...	?	?	?	yes	yes	yes	poor	yes	no	ckd

Figure 1: Dataset imported successfully

```
In [4]:
1 data.size
```

```
Out[4]: 10000
```

```
In [5]:
1 data.shape
```

```
Out[5]: (400, 25)
```

Figure 2: Dataset shape and size

```
In [7]:
1 import numpy as np
```

```
In [8]:
1 data=data.replace('?',np.nan)
```

```
In [9]:
1 data
```

Out[9]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	class
0	48	80	1.02	1	0	NaN	normal	notpresent	notpresent	121	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	7	50	1.02	4	0	NaN	normal	notpresent	notpresent	NaN	...	38	6000	NaN	no	no	no	good	no	no	ckd
2	62	80	1.01	2	3	normal	normal	notpresent	notpresent	423	...	31	7500	NaN	no	yes	no	poor	no	yes	ckd
3	48	70	1.005	4	0	normal	abnormal	present	notpresent	117	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	51	80	1.01	2	0	normal	normal	notpresent	notpresent	106	...	35	7300	4.6	no	no	no	good	no	no	ckd
5	60	90	1.015	3	0	NaN	NaN	notpresent	notpresent	74	...	39	7800	4.4	yes	yes	no	good	yes	no	ckd
6	68	70	1.01	0	0	NaN	normal	notpresent	notpresent	100	...	36	NaN	NaN	no	no	no	good	no	no	ckd
7	24	NaN	1.015	2	4	normal	abnormal	notpresent	notpresent	410	...	44	6900	5	no	yes	no	good	yes	no	ckd
8	52	100	1.015	3	0	normal	abnormal	present	notpresent	138	...	33	9600	4	yes	yes	no	good	no	yes	ckd
9	53	90	1.02	2	0	abnormal	abnormal	present	notpresent	70	...	29	12100	3.7	yes	yes	no	poor	no	yes	ckd

Figure 3: Replacing the garbage values with null values

```
In [10]:
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
age      391 non-null object
bp       388 non-null object
sg       353 non-null object
al       354 non-null object
su       351 non-null object
rbc      248 non-null object
pc       335 non-null object
pcc      396 non-null object
ba       396 non-null object
bgr      356 non-null object
bu       381 non-null object
sc       383 non-null object
sod      313 non-null object
pot      312 non-null object
hamo     348 non-null object
pcv      330 non-null object
wc       295 non-null object
rc       270 non-null object
htn      398 non-null object
dm       398 non-null object
cad      398 non-null object
appet    399 non-null object
pe       399 non-null object
ane      399 non-null object
class    400 non-null object
dtypes: object(25)
```

Figure 4: Null values of each column in the dataset

Preprocessing

```
In [11]:
```

```
#Replacing the null values in each column with mean, median and mode of the columns
#Replacing categorical values with mode of the column
print(data['rbc'].mode())
print(data['pc'].mode())
print(data['ba'].mode())
print(data['htn'].mode())
print(data['dm'].mode())
print(data['cad'].mode())
print(data['appet'].mode())
print(data['pe'].mode())
print(data['ane'].mode())
```

Figure 5: Finding the mode of the categorical data columns

In [12]:
```
data['rbc'].replace(np.nan,'normal',inplace=True)
data['pc'].replace(np.nan,'normal',inplace=True)
data['pcc'].replace(np.nan,'normal',inplace=True)
data['ba'].replace(np.nan,'notpresent',inplace=True)
data['htn'].replace(np.nan,'no',inplace=True)
data['dm'].replace(np.nan,'no',inplace=True)
data['cad'].replace(np.nan,'no',inplace=True)
data['appet'].replace(np.nan,'good',inplace=True)
data['pe'].replace(np.nan,'no',inplace=True)
data['ane'].replace(np.nan,'no',inplace=True)
```

In [13]:
```
#Replacing numerical values with thier median and mean
data['age'].replace(np.nan,46,inplace=True)
data['bp'].replace(np.nan,70,inplace=True)
data['sg'].replace(np.nan,1.025,inplace=True)
data['al'].replace(np.nan,1,inplace=True)
data['su'].replace(np.nan,0,inplace=True)
data['bgr'].replace(np.nan,148,inplace=True)
data['bu'].replace(np.nan,57,inplace=True)
data['sc'].replace(np.nan,2.1,inplace=True)
data['sod'].replace(np.nan,138,inplace=True)
data['pot'].replace(np.nan,4.6,inplace=True)
data['hamo'].replace(np.nan,12.5,inplace=True)
data['pcv'].replace(np.nan,38,inplace=True)
data['wc'].replace(np.nan,8414,inplace=True)
data['rc'].replace(np.nan,4.7,inplace=True)
```

Figure 6: Replacing the null values with mode for categorical data and mean for continuous data

In [15]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
age      400 non-null object
bp       400 non-null object
sg       400 non-null object
al       400 non-null object
su       400 non-null object
rbc      400 non-null object
pc       400 non-null object
pcc      400 non-null object
ba       400 non-null object
bgr      400 non-null object
bu       400 non-null object
sc       400 non-null object
sod      400 non-null object
pot      400 non-null object
hamo     400 non-null object
pcv      400 non-null object
wc       400 non-null object
rc       400 non-null object
htn      400 non-null object
dm       400 non-null object
cad      400 non-null object
appet    400 non-null object
pe       400 non-null object
ane      400 non-null object
class    400 non-null object
dtypes: object(25)
```

Figure 7: The above figure shows that all the null values have been filled

In [17]:
```
#Changing data type from object to float for the numerical data
data['age']=data['age'].astype('float64')
data['bp']=data['bp'].astype('float64')
data['bgr']=data['bgr'].astype('float64')
data['bu']=data['bu'].astype('float64')
data['sg']=data['sg'].astype('float64')
data['sc']=data['sc'].astype('float64')
```

In [18]:
```
data['rc']= data['rc'].replace('\t?',0, regex=True)
data['wc']= data['wc'].replace('\t?',0, regex=True)
data['pcv']= data['pcv'].replace('\t?',0, regex=True)
```

In [19]:
```
data['sod']=data['sod'].astype('float64')
data['pot']=data['pot'].astype('float64')
data['hamo']=data['hamo'].astype('float64')
data['pcv']=data['pcv'].astype('float64')
data['wc']=data['wc'].astype('float64')
data['rc']=data['rc'].astype('float64')
```

Figure 8: Typecasting the numerical data to float data type

```
In [20]:
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
age      400 non-null float64
bp       400 non-null float64
sg       400 non-null float64
al       400 non-null object
su       400 non-null object
rbc      400 non-null object
pc       400 non-null object
pcc      400 non-null object
ba       400 non-null object
bgr      400 non-null float64
bu       400 non-null float64
sc       400 non-null float64
sod      400 non-null float64
pot      400 non-null float64
hamo     400 non-null float64
pcv      400 non-null float64
wc       400 non-null float64
rc       400 non-null float64
htn      400 non-null object
dm       400 non-null object
cad      400 non-null object
appet    400 non-null object
pe       400 non-null object
ane      400 non-null object
class    400 non-null object
dtypes: float64(12), object(13)
```

Figure 9: The above figure shows the information about the preprocessed dataset

```
In [49]:
1 x=data.drop('class',axis=1)

In [50]:
1 y=data['class']
2 y=pd.DataFrame(y)
```

Figure 10: Seperating the features from the output class

Feature Selection

```
In [51]:
1 #Since our output is categorical, we use ANOVA and chi-squared test for feature selection

In [52]:
1 #Using chi-squared test we can decide whether output is dependent or independet of a feature

In [53]:
1 #Numerical Data
2 cont=['age', 'bp', 'sg','bgr', 'bu','sc','sod', 'pot', 'hamo', 'pcv', 'wc', 'rc']
3 #Categorical Data
4 cat=['al','su','rbc','pc','pcc','ba','htn', 'dm','cad','appet','pe','ane']

In [54]:
1 from sklearn.preprocessing import LabelEncoder
2 lel1=LabelEncoder()

In [55]:
1 categorical=list(x.select_dtypes(include=['object']).columns.values)
2 for cat in categorical:
3     x[cat] = lel1.fit_transform(x[cat].astype(str))
```

Figure 11: Seperating the continuous and categorical data and Label Encoding the categorical data

```
In [56]:
1 from sklearn.feature_selection import chi2
2 from sklearn.feature_selection import f_classif
```

Figure 12: Importing chi2 and Anova functions (as output is categorical)

In [57]:

```
chivalue,pvalue=chi2(x[['al']],y)
print(pvalue)
chivalue,pvalue=chi2(x[['su']],y)
print(pvalue)
chivalue,pvalue=chi2(x[['rbc']],y)
print(pvalue)
chivalue,pvalue=chi2(x[['pc']],y)
print(pvalue)
chivalue,pvalue=chi2(x[['pcc']],y)
print(pvalue)
chivalue,pvalue=chi2(x[['ba']],y)
print(pvalue)
```

```
[1.74363558e-51]
[2.10625642e-22]
[0.05266006]
[0.0010735]
[0.07166598]
[0.00027995]
```

Figure 13: The above figure shows the result of chi2 test on the categorical data w.r.t output

In [58]:

```
chivalue,pvalue=chi2(x[['htn']],y)
print(pvalue)
chivalue,pvalue=chi2(x[['dm']],y)
print(pvalue)
chivalue,pvalue=chi2(x[['cad']],y)
print(pvalue)
chivalue,pvalue=chi2(x[['appet']],y)
print(pvalue)
chivalue,pvalue=chi2(x[['pe']],y)
print(pvalue)
chivalue,pvalue=chi2(x[['ane']],y)
print(pvalue)
```

```
[5.91599066e-21]
[1.22989874e-19]
[6.28297953e-06]
[2.31150706e-12]
[1.4504158e-11]
[1.97317529e-09]
```

In [59]:

```
#Significance value=0.01
x=x.drop('rbc',axis=1)
x=x.drop('dm',axis=1)
x=x.drop('cad',axis=1)
x=x.drop('pcc',axis=1)
```

Figure 14: Dropping features whose p value is less that significance value (0.01)

In [60]:

```
#ANOVA test
```

In [61]:

```
cont=['age', 'bp', 'sg','bgr', 'bu','sc', 'sod', 'pot', 'hamo', 'pcv', 'wc', 'rc']
```

In [62]:

```
fvalue,pvalue=f_classif(data[cont],y)
```

```
C:\Users\Sudeepthi\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [63]:

```
print([(cont[i],pvalue[i]) for i in range(len(cont))])
```

```
[('age', 7.996860620950618e-06), ('bp', 7.932301593523614e-09), ('sg', 2.8101939781831477e-33), ('bgr', 6.503884438854131e-17), ('bu', 1.423864828579862e-14), ('sc', 2.4282684350088497e-09), ('sod', 6.129139402676853e-12), ('pot', 0.13158786074826355), ('hamo', 7.373587221552905 5e-68), ('pcv', 6.657698680490167e-10), ('wc', 3.4674273251197027e-15), ('rc', 2.2717519145343 49e-22)]
```

Figure 15: The above figure shows the result of ANOVA test on the continuous data w.r.t output

```
In [64]:  1  x=x.drop('pot',axis=1)

In [65]:  1  x
```

Out[65]:

	age	bp	sg	al	su	pc	ba	bgr	bu	sc	sod	hamo	pcv	wc	rc	htn	appet	pe	ane
0	48.0	80.0	1.020	1	0	1	0	121.0	36.0	1.2	138.0	15.4	0.0	0.0	0.0	1	0	0	0
1	7.0	50.0	1.020	4	0	1	0	148.0	18.0	0.8	138.0	11.3	0.0	0.0	4.7	0	0	0	0
2	62.0	80.0	1.010	2	3	1	0	423.0	53.0	1.8	138.0	9.6	0.0	0.0	4.7	0	1	0	1
3	48.0	70.0	1.005	4	0	0	0	117.0	56.0	3.8	111.0	11.2	0.0	0.0	0.0	1	1	1	1
4	51.0	80.0	1.010	2	0	1	0	106.0	26.0	1.4	138.0	11.6	0.0	0.0	0.0	0	0	0	0
5	60.0	90.0	1.015	3	0	1	0	74.0	25.0	1.1	142.0	12.2	0.0	0.0	0.0	1	0	1	0
6	68.0	70.0	1.010	0	0	1	0	100.0	54.0	24.0	104.0	12.4	0.0	8414.0	4.7	0	0	0	0
7	24.0	70.0	1.015	2	4	0	0	410.0	31.0	1.1	138.0	12.4	0.0	0.0	0.0	0	0	1	0
8	52.0	100.0	1.015	3	0	0	0	138.0	60.0	1.9	138.0	10.8	0.0	0.0	0.0	1	0	0	1
9	53.0	90.0	1.020	2	0	0	0	70.0	107.0	7.2	114.0	9.5	0.0	0.0	0.0	1	1	0	1
10	50.0	60.0	1.010	2	4	0	0	490.0	55.0	4.0	138.0	9.4	0.0	8414.0	4.7	1	0	0	1
11	63.0	70.0	1.010	3	0	0	0	380.0	60.0	2.7	131.0	10.8	0.0	0.0	0.0	1	1	1	0
12	68.0	70.0	1.015	3	1	1	0	208.0	72.0	2.1	138.0	9.7	0.0	0.0	0.0	1	1	1	0
13	68.0	70.0	1.025	1	0	1	0	98.0	86.0	4.6	135.0	9.8	38.0	8414.0	4.7	1	1	1	0
14	68.0	80.0	1.010	3	2	0	1	157.0	90.0	4.1	130.0	5.6	0.0	0.0	0.0	1	1	1	0
15	40.0	80.0	1.015	3	0	1	0	76.0	162.0	9.6	141.0	7.6	0.0	0.0	0.0				

Anaconda Powershell Prompt

Figure 16: Dropping features whose p value is less that significance value (0.01)

```
In [66]:  1  x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 19 columns):
age      400 non-null float64
bp       400 non-null float64
sg       400 non-null float64
al       400 non-null int32
su       400 non-null int32
pc       400 non-null int32
ba       400 non-null int32
bgr      400 non-null float64
bu       400 non-null float64
sc       400 non-null float64
sod      400 non-null float64
hamo     400 non-null float64
pcv      400 non-null float64
wc       400 non-null float64
rc       400 non-null float64
htn      400 non-null int32
appet    400 non-null int32
pe       400 non-null int32
ane      400 non-null int32
dtypes: float64(11), int32(8)
memory usage: 47.0 KB
```

Figure 17: Selected features dataset is ready for training and prediction

Training and Testing

```
In [209]:  1  #training
           2  from sklearn.model_selection import train_test_split
           3  xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)

In [210]:  1  xtrain.shape
```

Out[210]: (320, 19)

Figure 18: Splitting the dataset into training and testing set

```
In [211]:
from sklearn.neural_network import MLPClassifier
algo=MLPClassifier(hidden_layer_sizes=(40,40,40,40),verbose=True)

In [212]:
algo.fit(xtrain,ytrain)
```

```
Iteration 1, loss = 6.40403982
Iteration 2, loss = 6.32707501
Iteration 3, loss = 6.65958322
Iteration 4, loss = 1.95766571
Iteration 5, loss = 1.68118827
Iteration 6, loss = 1.25561781
Iteration 7, loss = 0.91794481
Iteration 8, loss = 0.90924536
Iteration 9, loss = 1.01559334
Iteration 10, loss = 1.00845537
Iteration 11, loss = 0.90076375
Iteration 12, loss = 0.79481843
Iteration 13, loss = 0.75932431
Iteration 14, loss = 0.76618224
Iteration 15, loss = 0.75836905
Iteration 16, loss = 0.72802448
Iteration 17, loss = 0.70704807
Iteration 18, loss = 0.71424929
Iteration 19, loss = 0.66500772
Iteration 20, loss = 0.35289481
Iteration 21, loss = 0.36232305
Iteration 22, loss = 0.38274444
Iteration 23, loss = 0.34588363
Iteration 24, loss = 0.29191010
Iteration 25, loss = 0.30919306
Iteration 26, loss = 0.33793362
Iteration 27, loss = 0.34202285
Iteration 28, loss = 0.41848715
Iteration 29, loss = 0.59135794
Iteration 30, loss = 0.57607039
Iteration 31, loss = 0.41370095
Iteration 32, loss = 0.30901107
Iteration 33, loss = 0.34559367
Iteration 34, loss = 0.34648344
Iteration 35, loss = 0.42133542
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.

C:\Users\Sudeepthi\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:916: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
Out[212]: MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(40, 40, 40, 40), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=None, shuffle=True, solver='adam', tol=0.0001,
       validation_fraction=0.1, verbose=True, warm_start=False)
```

Figure 18: Setting up the Nueral Networks algorithm and training the model using training dataset

```
In [72]:
ypred=algo.predict(xtest)

In [73]:
from sklearn.metrics import confusion_matrix

In [74]:
confusion_matrix(ytest,ypred)
```

```
Out[74]: array([[49,  4],
       [ 0, 27]], dtype=int64)
```

```
In [75]:
from sklearn.metrics import accuracy_score

In [76]:
accuracy_score(ytest,ypred)
```

```
Out[76]: 0.95
```

Figure 19: Testing the model and computing performance measures

Results (Prediction)

```
In [78]:  1 testdata=pd.read_csv(r"C:\Users\Sudeepthi\Downloads\Chronic_Kidney_Disease\ckdpreddata.csv")

In [79]:  1 testdata
```

Out[79]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	hamo	pcv	wc	rc	htn	dm	cad	appet	pe	ane
0	48	70	1.005	4	0	normal	abnormal	present	notpresent	117	...	11.2	32	6700	3.9	yes	no	no	poor	yes	yes
1	63	70	1.010	3	0	abnormal	abnormal	present	notpresent	380	...	10.8	32	4500	3.8	yes	yes	no	poor	yes	no
2	45	70	1.025	2	0	normal	abnormal	present	notpresent	117	...	10.0	30	19100	3.7	no	no	no	good	no	no
3	30	80	1.020	0	0	normal	normal	notpresent	notpresent	131	...	14.1	45	9400	5.3	no	no	no	good	no	no

4 rows × 24 columns

Figure 20: The dataset of examples that have not been assigned a class to use for prediction

```
In [87]:  1 testdata=testdata.drop('rbc',axis=1)
          2 testdata=testdata.drop('dm',axis=1)
          3 testdata=testdata.drop('cad',axis=1)
          4 testdata=testdata.drop('pcc',axis=1)
          5 testdata=testdata.drop('pot',axis=1)
```

Figure 21: Matching the number of features to the model

```
In [88]:  1 testpred=algo.predict(testdata)

In [89]:  1 testpred

Out[89]: array(['ckd', 'ckd', 'notckd', 'ckd'], dtype='<U6')
```

Figure 22: Prediction values of the classes

V. Conclusions

CKD classification and detection can be used as an analysing tool for providing a second opinion to the doctors. We trained the Neural Network model to predict the existence of chronic kidney disease. It makes use of a list of attributes like age, blood pressure, specific gravity, albumin, sugar, red blood cells, pus cell, pus cell clumps, bacteria, blood glucose random, blood urea, serum creatinine, sodium, potassium, haemoglobin, packed cell volume, white blood cell count, red blood cell count, hypertension, diabetes mellitus, coronary artery disease, appetite, pedal edema, anaemia which are collected from the patient to make the prediction. This system can detect a chronic condition of kidney disease based on several factors with an accuracy of 95% and can therefore be used for the better prediction of chronic kidney detection.

References

Mohamed Elhoseny, K Shankar, J Uthayakumar. (2019). “Intelligent Diagnostic Prediction and Classification System for Chronic Kidney Disease”. JO - Scientific Reports, SP-9583, VL- 9 , IS- 1, SN - 2045-2322, DO - 10.1038/s41598-019-46074-2

Chakrapani, Sumit Raj, VibhavPrakash Singh, DhrubJyoti Kalita. (2019). “Detection of Chronic Kidney Disease Using Artificial Neural Network”. International Journal of Applied Engineering Research ISSN 0973-4562 Volume 14, Number 10, 2019 (Special Issue)

Ahmad, Mubarik & Tundjungsari, Vitri & Widianti, Dini & Amalia, Peny & Rachmawati, Ummi. (2017). “Diagnostic decision support system of chronic kidney disease using support vector machine”. 1-4. 10.1109/IAC.2017.8280576.