International Journal of Functional Research in Science and Engineering
e-ISSN: 2814-0923
www.ijfrse.com
Volume 3; Issue 4; December 2025; Page No. 89-103.

ARTIFICIAL INTELLIGENCE PREDICTION OF CHRONIC KIDNEY FAILURE

Ernest-Okoye, Ngozi[1]
&
Odogwu, Ugochukwu Chinedu[2]

*[1&2]Anambra State Polytechnic, Mgbakwu. Nigeria

Abstract
Adaptability and survival of the human body is dependent on the ability of separate body organs to work together as a system. The kidney as one of the important organs of the body deserves proper care to avoid failure. This research sort to simulate and postulate the threshold points of kidney failure using SEUCR test results obtained from indigenous hospitals in the Eastern part of Nigeria and synchronized in TensorFlow; an open source library for numerical computation. The result outcome is presented in realtime large-scale and verified with machine learning. A Virtual kidney diagnostic platform at 97% accuracy depicted threshold points of kidney failures

Keywords: Artificial Intelligence, Prediction, Chronic kidney failure, Organ, Hospital.

Introduction

Human bodies consist of a number of biological systems that carry out specific functions necessary for everyday living. These twelve systems carry out distinct and necessary activities for human survival. They are Circulatory system, Digestive system, Endocrine system, Immune system, Lymphatic system, Nervous system, Muscular system, Reproductive system, Skeletal system, Respiratory system, Renal system and Integumentary system. The adaptability and survival of the human body is dependent on the ability of separate body systems to work together. All these systems make use of the five important organs of the body which include Brain, Heart, Kidneys, Liver and Lungs.

According to Rettner, (2016), estimation of thirty trillion cells exists in the human body with at least 10 times as many bacteria as cells. The job of the kidneys as one of the organs of the body is to remove waste and extra fluid from the blood with the help of other organs and the twelve body systems. As such, jobs of the kidneys in the body are not limited to removal of waste products and extra fluid from the blood. The kidneys take urea out of the blood and combine it with water and other substances to make urine, clean blood, control chemicals and fluids in the body, help control the blood pressure and help make red blood cells. Rettner opined that each day, the kidneys process about 200 quarts (50 gallons) of blood to filter out about 2 quarts of waste and water as urea out of the blood and combine it with water and other substances to make urine. When ones' kidneys are damaged, they may not work as well as they should. If the damage to the kidneys continues to get worse, there is tendency of the kidneys being less and less able to do their job. This condition according to Rettner leads to chronic kidney disease. Kidney failure is the last (most severe) stage of chronic kidney disease. This is why kidney failure is also called end-stage renal disease or ESRD for short. When the kidneys fail, it means they have stopped working well enough for one to survive without dialysis or a kidney transplant. Dialysis as one of the means of managing kidney failure can do only some, not all, of the jobs that healthy kidneys do. Therefore, even when one is being treated for kidney failure, one may have some problems that come from having kidneys that don't work well.

Statement of the Problem

Kidney failure refers to temporary or permanent damage to the kidneys that result in loss of normal kidney function. In most cases the symptoms of kidney failure are manifested by the patient and in some other cases they do not, resulting to acute (sudden) kidney failure. Certain conditions poised by some patients may lead to either permanent or acute (reparable) kidney failure. The physical symptoms of kidney failure like the swelling of some parts of the body and the presence of urea plus other substances does not distinctively pronounce kidney failure, hence the need for humanity to know the exact point at which the presence of the above conditions of health will spell doom for the kidney.

This research intends to research on conditions harmful to the normal functioning of the kidney with the view of nibbling kidney failure at the bud by using real-time diagnostic expert system that will simulate and train the four major coefficients of a failed kidney with the following objectives;

- To gather test results of susceptible degraded kidney patients as data input.
- To develop an ANN using tensorflow as the engine for simulation of data.
- To simulate and train the input data for determination of threshold point.
- To postulate exact point/s of kidney failure.

Review of Related Literature

Medical researchers have proved that the major causes of kidney degradation are diabetes, High Blood Pressure, urinary tract infections and polycystic syndrome. Engineering scholars over the years have utilized large data on the above criteria to proffer solutions to the menace of CKD.

Segal, Kalife and Koren (2020) utilized ML algorithm for early detection of End-Stage Renal disease by predicting a model for progression to ERSD based on a large scale multidimensional database. 10,000,000 medical insurance claims from 550,000 patient records under commercial health insurance database for patients over 18 years were utilized for the prediction. At 95% accuracy and with the designed machine, Segal was able to predict the threshold point at 0.715 sensitivity.

Charumathi (2020) developed Deep Learning Algorithm (DLA) to detect CKD using data sourced from retinal images from three – population based multiethnic cross-sectional studies in Singapore and China. 5188 patients from 40years and above were used and result validated with 1297 at 95% accuracy.
Makinno, Yoshimoto and Suzuki (2019) developed intelligent machine to predict the progression of Diabetic Kidney Disease using Big Data Machine Learning. Their virgin model, natural language and longitudinal data were based on electronic medical records of 64,059 diabetic patients over six months. 24 factors were selected to find the serial patterns relating to 6-month DKD aggravation using convolutional autoencoder. Prediction was at 71% accuracy.

Senan et al (2012) in their work titled Diagnosis of CKD using effective classification algorithms and recursive feature elimination techniques evaluated a dataset collected from 400 patients containing 24

features divided into 11 numerical and 13 categorical features, with the dataset divided into 75% for training and 25% for testing and validation, result showed that Random forest algorithm out-performed the rest of the classifiers with accuracy of 100% thus validated the research at 97.3% accuracy.
Koyner, Carey, Edelson and Churpek (2018) developed an acute kidney injury risk prediction model using electronic health record data for longitudinal use in hospitalized patients. Creatinine surrogate from inpatients adults without pre-existing renal failure at admission was used. The researchers deployed demographics, vital signs and diagnostic reports on 121,158 patients. 60% of the total population was used for testing and 40% for validation. At a probability threshold of greater than or equal to 0.022, the algorithm had a sensitivity of 84% and a specificity of 85% for stage 2 acute kidney injuries.

Methodology

The materials for the early detection of CKD are in two forms; software and hardware. The hardware include test samples obtained from indigenous hospitals in the Eastern part of the Nigeria and which are comparable with test samples obtainable from other parts of the world except in creatinine as same is dependent on indigenous food intake. The software is composed with tensorflow, SKLearn, Matplot, Python, Numpy. The two bodies merged by feeding the physical samples into the TensorFlow platform for deep machine learning. The knowledge derived will be compared with a normal kidney to determine normal range, early-stage renal degradation, late stage renal degradation and critical point of kidney failure.
Creation of the tensorflow involves five progressive sections.

Williams (2019) iterated the five steps involved in tensorflow output pipeline to include;
- Data generation or creation
- Placeholder formulation
- Dataset definition
- Pipeline creation
- Program execution

Data creation for the tensorflow input pipeline he continued; entails generation of data either in samples or arrays. The dataset for this paper will be generated from test samples of persons with renal disease and susceptible cum potential CKD patients gathered from local health facilities; Rock Foundation Hospital Awka, Runia Specialist Hospital, Awka, Glanson Laboratories Awka, Nnamdi Azikiwe Teaching Hospital, Nnewi and the online data collected from National Kidney Foundation archives. Data generation consists of learning a mapping function f(X) that takes the set of inputs (X and in this dissertation; the SEUCR quantities; Creatinine, Sodium and Urea) containing features hitherto unknown to the researcher and maps them with minimal error to the outputs (Y) in the form of Normal (N) or Failed kidney (F). The general process of the mapping involves finding signals/patterns in our data after the algorithm has essentially reduced the dispersion in the data to an acceptable minimum. One of the most widely used approach to machine learning is the use of neural networks. The core of neural networks used is the perceptron i.e the machine equivalent of the biological neuron.

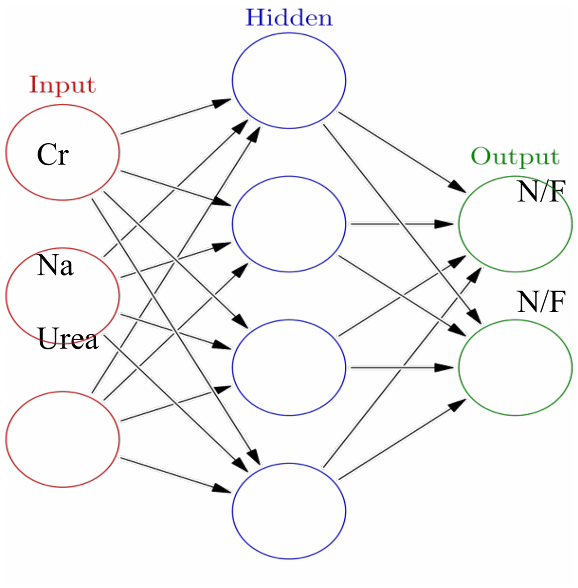


Fig 1: Node connection of ML

As stated earlier, the f(X) for this work is derived from Sodium, Electrolyte, Urea and Creatinine being substances released as waste from the kidneys. These generated data (FX) are prepared by arbitrarily sorting before been fed into the hidden layer for interpolation and actual modelling. The actual data sorting cum preparation for this work was done in Ms Excel environment with sorting IF statement codes. The aim of the sorting is to present unarranged data to the intelligent machine; criteria needed for advanced intelligent selection. Blood and urine tests show how well the kidneys are doing their job which entails how quickly body wastes are being removed. Leakage of abnormal amounts of protein from the kidney can also be viewed from the urine tests which signify kidney damage.

Table 1
Criteria Test Samples

S/N	Patients' Parameters	Amount
1	Blood Urea	61
	Creatinine	3.5
	Potassium	4.4
	Sodium	140
	Chloride	105
	Bicarbonate	25
	Calcium	2.2

2	Blood Urea	21
	Creatinine	1.3
	Potassium	4.0
	Sodium	145
	Chloride	90
	Bicarbonate	22
	Calcium	1.2
3	Blood Urea	17
	Creatinine	0.5
	Potassium	4.4
	Sodium	138
	Chloride	106
	Bicarbonate	15
	Calcium	2.2
4	Blood Urea	50
	Creatinine	2.7
	Potassium	3.4
	Sodium	128
	Chloride	100
	Bicarbonate	26
	Calcium	2.2
5	Blood Urea	27
	Creatinine	1.2
	Potassium	3.4
	Sodium	125
	Chloride	144
	Bicarbonate	23
	Calcium	1.8

The second stage is place holder generation; a tool capable of assigning variable data to a tensorflow pipeline. Place holder generation has the capability of short time memory as it holds data to be assigned on later or future date. Tf.data is the tool cum API used for the purpose of this work to actualize the prediction of kidney degradation by building a node to node interphase forming present state of the organ with the aid of data aggregated from laboratory results which was fed into its pipeline.

The first stage in the development of Tensorflow environment for ML is Installation of Python and libraries and their dependencies. The latest version of python; Python 3 was used on TensorFlow v2.0, as it presents a better API and integration with python machine learning libraries. Other libraries deployed are Keras, Pandas, Matplot Lib, Seaborn, Sklearn, Numpy.

The Python Code was written using the Visual Studio Code Editor because its user friendly with python, a feature enabled by the Python Extension for Visual Studio Code. The Selected Python Server was installed as a Virtual Environment. The Python Keras library is designed to work with a TensorFlow backend as default hence this paper's default mode.

The second stage involves Importation of Data. This was done using the Pandas python library, which can import data from a .csv extension file saved as comma delimited strings. After importation of data, a

review of the imported data was carried out. This review enables the researcher to crosscheck the imported data with the manuscript for the purpose of accuracy.
The code below achieves the data importation;

```
import pandas as pd
df =
pd.read_csv("C:/Users/Sly/Documents/TensorFlow/KidneyFailureModel/seucr_dataset.csv",delimiter=','
)
```

and viewing thus;

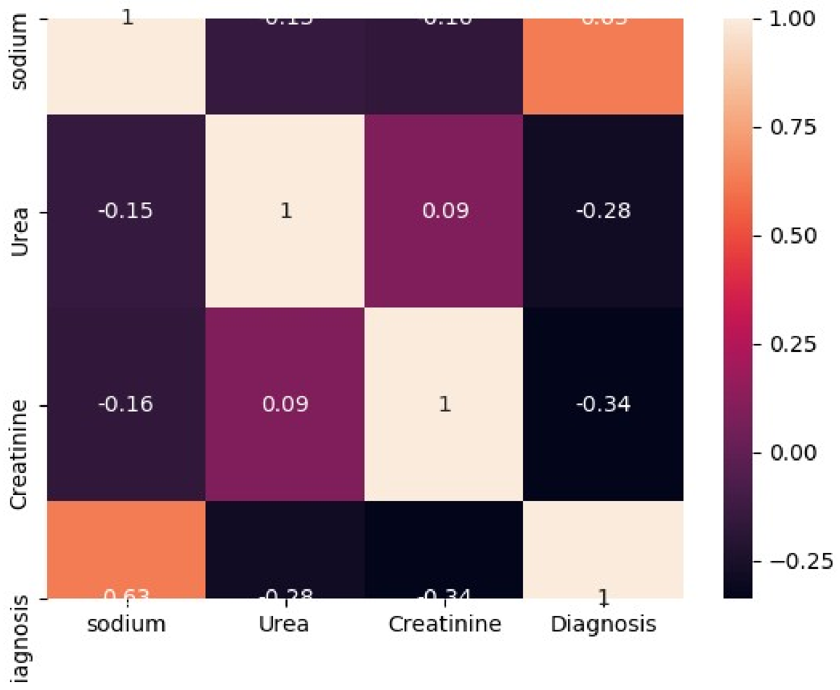
Next is a look into the structure and simple statistical measures of the data.

```
#display first 5 rows of the data
print(df.head())

############ OUTPUT ###############################
sodium Electrolyte  Urea  Creatinine sodium_A Creatinine_A Urea_A  (Diagnosis)
0    145          P              21        1.3       N           N     AB        1
1    140          N        19        1.0       N           N     N         1
2    139          N        18        0.6       N          AB     N         1
3    141          N        20        1.3       N           N     N         1
4    144          N        18        1.2       N           N     N         1

# display various statistical info about our data
print(df.describe())

############ OUTPUT ###############################
          sodium       Urea        Creatinine   Diagnosis
count  300.000000  300.000000  300.000000  300.000000
mean   134.156667   32.166667   1.351667    0.450000
std     6.706368         13.886745    0.569307    0.498325
min    118.000000    7.000000    0.500000    0.000000
25%    129.000000  21.000000    0.900000    0.000000
50%    135.000000  29.000000    1.200000    0.000000
75%    139.000000  40.000000    1.700000    1.000000
max    151.000000  88.000000    3.900000   1.000000
```

The prediction column represents the boolean result of the test with [1] – Normal Kidney Function and [0] – Kidney Failure, as can be seen, there are 300 rows of data to work with.
The third stage is Examining Statistical Correlation. It is always good practice to examine data before diving into model architecture and training. If the output data already has a strong correlation with input variables, then it might be sufficient to apply statistical methods and still obtain largely satisfactory results. It's pertinent to examine the correlation between the data using a Matplot; a plot showing levels of correlation interaction between the data. The Seaborn library aids in this task as demonstrated below:

```
# check if there is correlation in our data before training
import seaborn as sns
import matplotlib.pyplot as plt
```

```
corr = df.corr()
sns.heatmap(corr,annot=True,
xticklabels=corr.columns.values,
yticklabels=corr.columns.values)
plt.show()
```



From the above table, there is only limited correlation between kidney condition and sodium (~60%).Other parameters show minimal to no statistical correlation respectively with the entity. Essentially, correlation is largely insufficient for a diagnosis for this case study and as such, determination of kidney correction to critical determination is a case perfectly suited for machine learning since the neural net will extract more features from the data than simple correlation determination.

The fourth step in building of this Tensorflow model is data preparation which includes:

- ***Shuffling imported data:***

To enable better training accuracy, this work ensured that input data rows do not follow a pattern e.g. NORMAL, NORMAL, NORMAL, NORMAL, KIDNEY FAILURE, KIDNEY FAILURE, KIDNEY FAILURE, KIDNEY FAILURE, etc for this may affect the training / testing data by training on one pattern and test on a different pattern. The above scenario if not checked will lead to poor results and is a specific case of over-fitting problem. To avoid the above anomaly, non-productive approach was utilized by shuffling data rows with the column values unchanged and resetting the array indices.

Dataset Definition

The dataset deployed in this dissertation for the determination of CKD consists of minimum, normal/average and maximum limits. Minimum limit refers to the barest obtainable limit and it transcends to the normal range; a level considered safe for the smooth operation of the kidneys in the renal system as opined by *Papadakis; McPhee; Rabow (2018). The normal range leads into maximum limit which is triggered by either other ailments, excessive intake of toxic substances e.g drugs and / or hereditary factors. At this stage or level, the individual is placed under one or two of the CKD management procedures. A little tilt from this level pushes the kidney to its extreme point of degradation and consequently* kidney failure.

This work avails the human populace opportunity of viewing 5-point scale on a single platform, the various stages of kidney degradation as well as the approximate critical point as applicable to Igbo race. The test samples used for this work was collected from the eastern part of the country and as such the result is domiciled with same location owing to the fact that creatinine level differ from zone to zone; a criterion produced from wears and tears of consumed food by a region and age. Dataset definition was achieved by creating a dynamic realtime response from the input criteria (the three point criteria) in a tensorflow platform which portrays the expected outcome

Pipeline creation

Pipeline was created with the six (6) libraries (Keras, Pandas, MatplotLib, Seaborn, Sklearn, Numpy) imported into Python environment to create a complete Python code base modelling platform as shown below:

```
import pandas as pd
df =
pd.read_csv("C:/Users/Sly/Documents/TensorFlow/KidneyFailureModel/seucr_dataset.csv",delimiter=','
)

# shuffle our data to ensure optimal training results
from sklearn.utils import shuffle
df = shuffle(df)
df.reset_index(inplace=True,drop=True)

#display first 5 rows of the data
print(df.head())

# display various statistical info about our data
print(df.describe())
check if there is correlation in our data before training
import seaborn as sns
import matplotlib.pyplot as plt
corr = df.corr()
sns.heatmap(corr,annot=True,
xticklabels=corr.columns.values,
yticklabels=corr.columns.values)
plt.show() # will require closing dialog to continue...
```

```python
import numpy as np
X = np.array(df[['sodium','Urea','Creatinine']])# use only the sodium, urea and Creatine columns for our inputs
y = np.array(df['Diagnosis'])# use Diagnosis as our output
X = X[:300,]# trim the arrays to our data size.
y = y[:300,]

print("Input Array Shape: ", X.shape)# show the dimensions of our input values array
print("Output Array Shape: ", y.shape)# show the dimensions of our output array

y = np.reshape(y,(-1,1))
print("Output Array After ReShaping: ", y.shape)# show the dimensions of our output array

from sklearn.model_selection import train_test_split
#split our data into training and testing data with a 70-30 ratio
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.3,random_state=5)

print("X_train Shape: ", X_train.shape)
print("X_test Shape: ", X_test.shape)

print("y_train Shape: ", y_train.shape)
print("y_test Shape: ", y_test.shape)

#save original test parameters for display
X_test_unscaled = X_test

# normalize the input variables
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

print("Running Binary Logistic Classification Algorithm...")

from keras.models import Sequential
from keras.layers import Dense

model = Sequential()

model.add(Dense(64,input_dim=3,activation='relu'))
model.add(Dense(16,activation='relu'))
model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(
   X_train,
   y_train,
epochs=100,
batch_size=5,
verbose=1,
validation_data=(X_test, y_test)
)
print("Model Training Complete...")

print("Model Summary:")
model.summary()

print("Accuracy Score: ")
score = model.evaluate(X_test, y_test,verbose=1)
print(score)

# visualize some of our results against true values
y_pred = model.predict_classes(X_test)
print("Sodium-Urea-Creatine: Predictions vs Actual Values: ")
i =0
while i <len(y_pred):
print(
"Na - Urea - Creatine",
      X_test_unscaled[i],
" =>",
"Pred.:",
"NORMAL"if y_pred[i]==1else"KIDNEY FAILURE",
' vs ',
"Actual:",
"NORMAL"if y_test[i]==1else"KIDNEY FAILURE")
    i +=1
```

This paper drew up data made of 300 samples of susceptible patients of CKD ie patients with diabetes or showing signs of diabetes, patients with High Blood Pressure (HBP), patients with family history of chronic kidney diseases (polycystic group) and patients with urinary tract infections

The parameters used for this simulation are Blood urea, creatinine and sodium as drawn from the samples because they are evidently visible in SEUCR tests. The fourth criterion is electrolyte indicated either as positive or negative owing to the presence of certain elements at a time as shown below.

Table 2
Patients' SEUCR results

S/N	Patients' Parameters	Amount	Parameters Used	Electrolyte Status
1	Blood Urea	61	61	Positive
	Creatinine	3.5	3.5	
	Potassium	4.4		
	Sodium	140	140	
	Chloride	105		
	Bicarbonate	25		
	Calcium	2.2		
2	Blood Urea	21	21	Positive
	Creatinine	1.3	1.3	
	Potassium	4.0		
	Sodium	145	145	
	Chloride	90		
	Bicarbonate	22		
	Calcium	1.2		
3	Blood Urea	17	17	Positive
	Creatinine	0.5	0.5	
	Potassium	4.4		
	Sodium	138	138	
	Chloride	106		
	Bicarbonate	15		
	Calcium	2.2		
4	Blood Urea	50	50	Positive
	Creatinine	2.7	2.7	
	Potassium	3.4		
	Sodium	128	128	
	Chloride	100		
	Bicarbonate	26		
	Calcium	2.2		
5	Blood Urea	27	27	Positive
	Creatinine	1.2	1.2	
	Potassium	3.4		
	Sodium	125	125	
	Chloride	144		
	Bicarbonate	23		
	Calcium	1.8		

Program Execution

To execute this tensorflow program, the Ms Excel prepared data(sorted and randomly scattered) was fed into the platform through Sequential modelling designed to the process of fine-tuning the loss of the model until the minimal loss attainable by the model was obtained. At this point, the model is regarded to as converged with the features as; dataset (array of Sodium, Urea & Creatinine levels) and the output dataset (array of diagnoses) were passed to the model trainer. Next in line was specification of the number of epochs and batch sizes (also starting out small) then model training commences. Since the training dataset size is relatively small ie 300, the training started out the test with 100 epochs.

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()

model.add(Dense(8,input_dim=3,activation='relu'))
model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

model.fit(
   X_train,
   y_train,
epochs=100,
batch_size=5,
verbose=1,
)
############ OUTPUT ##############################
Using TensorFlow backend.
Epoch 1/100
210/210 [==============================] - 0s 2ms/step - loss: 0.6855 - accuracy: 0.5714
Epoch 2/100
210/210 [==============================] - 0s 681us/step - loss: 0.6066 - accuracy: 0.6048
Epoch 3/100
210/210 [==============================] - 0s 815us/step - loss: 0.5478 - accuracy: 0.7143
Epoch 4/100
210/210 [==============================] - 0s 691us/step - loss: 0.5043 - accuracy: 0.7952
...

Epoch 98/100
210/210 [==============================] - 0s 605us/step - loss: 0.2604 - accuracy: 0.8952
Epoch 99/100
210/210 [==============================] - 0s 639us/step - loss: 0.2594 - accuracy: 0.8952
Epoch 100/100
210/210 [==============================] - 0s 558us/step - loss: 0.2590 - accuracy: 0.8952
```

Model Training Complete...

Criterion Selection

As stated in Literature review above, the established criteria according to National Kidney Foundation are contained in the SEUCR diagnosis. This work borrowed the existing fact to create a platform that will view the four quantities (Sodium, Creatinine, Urea and Electrolyte) in a model structure. The criterion selection entails viewing the four quantities, testing, training and displaying predicted results. The technique used in program execution will be deployed i.e middling with the whole data sample and recording 100 randomly selected samples. The contents of the model structure were deduced by viewing its summary. Keras conveniently provides the <Summary ()> API extension for this task as shown below;

```
print("Model Summary:")
model.summary()

############ OUTPUT ###############################
Model: "sequential_1"
_________________________________________________________________
Layer (type)              Output Shape            Param #
=================================================================
dense_1 (Dense)              (None, 8)              32
_________________________________________________________________
dense_2 (Dense)              (None, 1)              9
=================================================================
Total params: 41
Trainable params: 41
Non-trainable params: 0
```

From the model structure above, modelling consists of 2 layers. The total number of parameters inferred during feature extraction was 41. Next is to test the results and see if there is need for more complexity. To test the model for accuracy, one makes predictions on the test data split earlier. This is done using the Keras <evaluate ()> API extension.

```
print("Accuracy Score: ")
score = model.evaluate(X_test, y_test, verbose=1)
print(score)

############ OUTPUT ###############################
_________________________________________________________________
Accuracy Score:
90/90 [==============================] - 0s 234us/step
[0.3570466885964076, 0.855555534362793]
```

The above depicts that the first try model of minimal complexity and predicted correctly with an 85.6% accuracy and loss standing at approximately 0.36 which is not high considering the data range. Having trained and tested the model, the next step available for criterion prediction is displaying of prediction results and viewing the variations amongst the variables.

```
# visualize some of our results against true values
y_pred = model.predict_classes(X_test)
print("Sodium-Urea-Creatine: Predictions vs Actual Values: ")
i =0
while i <len(y_pred):
print(
"Na - Urea - Creatine",
      X_test_unscaled[i],
" =>",
"Pred.:",
"NORMAL"if y_pred[i]==1else"KIDNEY FAILURE",
' vs ',
"Actual:",
"NORMAL"if y_test[i]==1else"KIDNEY FAILURE")
   i +=1
############ OUTPUT ###############################
Sodium-Urea-Creatine: Predictions vs Actual Values:
Na - Urea - Creatine [122.  34.   0.7] =>  Pred.: KIDNEY FAILURE  vs  Actual: KIDNEY FAILURE
Na - Urea - Creatine [136.  19.   2.3] =>  Pred.: KIDNEY FAILURE  vs  Actual: NORMAL
Na - Urea - Creatine [135.  31.   1.7] =>  Pred.: KIDNEY FAILURE  vs  Actual: KIDNEY FAILURE
Na - Urea - Creatine [144.  18.   0.7] =>  Pred.: NORMAL  vs  Actual: NORMAL
Na - Urea - Creatine [129.  30.   2.3] =>  Pred.: KIDNEY FAILURE  vs  Actual: KIDNEY FAILURE
Na - Urea - Creatine [138.  22.   1.1] =>  Pred.: NORMAL  vs  Actual: NORMAL
Na - Urea - Creatine [140.  21.   1.6] =>  Pred.: NORMAL  vs  Actual: KIDNEY FAILURE
Na - Urea - Creatine [141.  15.   1.1] =>  Pred.: NORMAL  vs  Actual: NORMAL
Na - Urea - Creatine [143.  20.   1.1] =>  Pred.: NORMAL  vs  Actual: NORMAL
Na - Urea - Creatine [120.  37.   1.5] =>  Pred.: KIDNEY FAILURE  vs  Actual: KIDNEY FAILURE
...
Na - Urea - Creatine [142.  25.   0.9] =>  Pred.: NORMAL  vs  Actual: NORMAL
Na - Urea - Creatine [121.  28.   1.3] =>  Pred.: KIDNEY FAILURE  vs  Actual: KIDNEY FAILURE
Na - Urea - Creatine [125.  28.   2.2] =>  Pred.: KIDNEY FAILURE  vs  Actual: KIDNEY FAILURE
```

Conclusion

Using the Virtual kidney diagnostic platform at 97% accuracy as indicated in the final model accuracy test to determine level of kidney degradation, the quantities depicted that at;

a. Creatinine 0 -14; Sodium 138 – Max limit (200) & Urea insignificant, the kidneys are normal;
b. Sodium [(138 – max limit (200)], Urea 0-20 & Creatinine insignificant, the kidneys are normal;
c. Urea 0-20, Creatinine (0- 3.4) & Sodium at max limit (200), the kidneys are normal.

Other arbitrary points in the platform outside the ones above yielded abnormal kidneys scenarios.

References

Rettner, R. (2018). 10 Amazing things we learned about Humans. LiveScience

American Kidney Fund Inc. (2019). *Complications of Kidney*. EIN: 23-7124261. 11921 Rockville Pike, Suite 300, Rockville, MD 20852 | 800-638-8299

TensorFlow (2018). TensorFlow Version Compatibility. API Documentation". Retrieved June 27, 2018.

Segal, Kalife & Koren (2020). ML Algorithm for Early Detection of End-Statge Renal Disease. BMC Nephrol, 21:518. doi:10.1186/212882-020-02093-0.

Sabanayagam, C. (2020). A Deep Learning Algorithm to Detect CKD. National Library of Medicine, Maryland

Makinno, Yoshimoto and Suzuki (2019). Artificial Intelligence Predicts the Progression of Diabetic Kidney Disease using Big Data ML. Scientific Reports: 11862.

Senan et al (2021). Diagnosis of CKD Using effective Classification Algorithms and Recursive Feature Elimination Techniques. Journal of Healthcare Engineering Article ID 1004767.

Koyner, JL, Carey, KA, Edelson DP and Churpek (2018). The Development of ML Inpatient Acute Kidney Injury Prediction Model.. PubMed ID: 29596073 DOI: 10.1097/CCM.0000000000003123.

Papadakis, M.; McPhee, S.; Rabow, M. (2018). *Current Medical Diagnosis & Treatment* (eds.). New York, NY: McGraw-Hill Education