EARLY PREDICTION CHRONIC KIDNEY DISASESED DETECTEION

1. INTRODUCTION

- CHRONIC KIDNEY DISEASE IS A SERIOUS LIFELONG CONDITION THAT INDUCED BY EITHER KIDNEY PATHOLOGY OR REDUCED KIDNEY FUNCTIONS. EARLY PREDICTION AND PROPER TREATMENTS CAN POSSIBLY STOP, OR SLOW THE PROGRESSION OF THIS CHRONIC DISEASE TO END-STAGE, WHERE DIALYSIS OR KIDNEY TRANSPLANTATION IS THE ONLY WAY TO SAVE PATIENT'S LIFE.

- IN THIS STUDY, WE EXAMINE THE ABILITY OF SEVERAL MACHINE-LEARNING METHODS FOR EARLY PREDICTION OF CHRONIC KIDNEY DISEASE. THIS MATTER HAS BEEN STUDIED WIDELY; HOWEVER, WE ARE SUPPORTING OUR METHODOLOGY BY THE USE OF PREDICTIVE ANALYTICS, IN WHICH WE EXAMINE THE RELATIONSHIP IN BETWEEN DATA PARAMETERS AS WELL AS WITH THE TARGET CLASS ATTRIBUTE. PREDICTIVE ANALYTICS ENABLES US TO INTRODUCE THE OPTIMAL SUBSET OF PARAMETERS TO FEED MACHINE LEARNING TO BUILD A SET OF PREDICTIVE MODELS

1.1 Overview:

Chronic kidney disease, or CKD, is a condition in which the kidneys are so damaged that they can't filter blood as well as they should. The kidneys' main job is to get rid of waste and extra water from the blood.8 This is how urine is made. CKD means that waste has built up in the body. This

condition is called chronic because the damage happens slowly over a long period of time. It is a disease that affects people all over the world.7 Because of CKD, you might experience various difficulties with your health. Diabetes, high blood pressure, and heart disease are only 3 of the many conditions that can lead to CKD. In addition to these serious health problems, age and gender also play a role in who gets a CKD.26 If one or both of your kidneys aren't working right, you may have a number of symptoms, such as back pain, stomach pain, diarrhea, fever, nosebleeds, rash, and vomiting. The 2 most common illnesses that might cause long-term damage to the kidneys are diabetes and high blood pressure.28 Therefore, the prevention of CKD can be thought of as the control of these 2 diseases. Because chronic kidney disease (CKD) does not often present any symptoms until it has progressed to a more advanced state, many people who have it do not realize they have it until it is too late.

PROJECT DESCRIPTION

1.2 PURPOSE

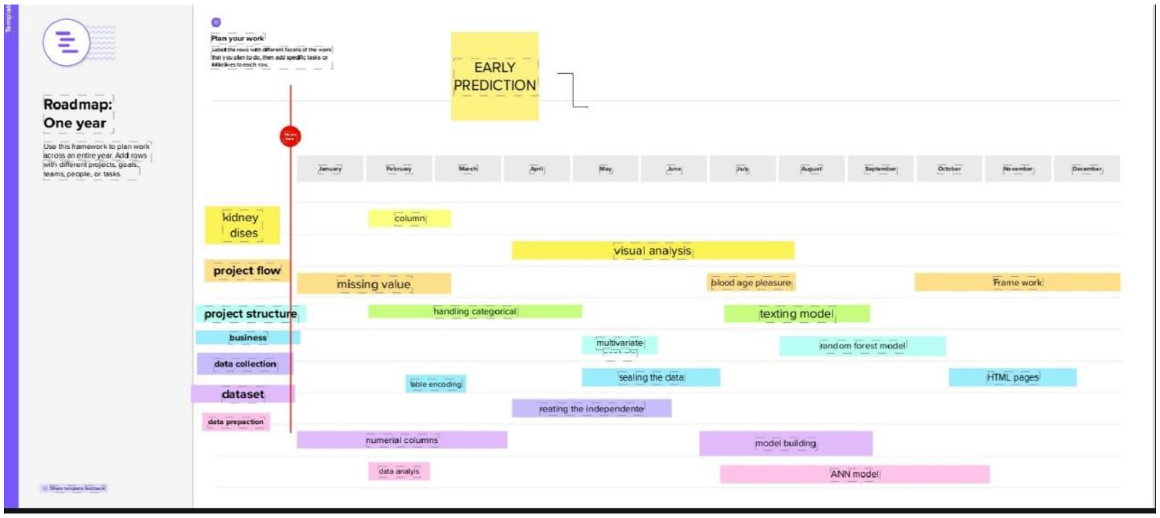
.GOAL THREE OF THE UN'S SUSTAINABLE DEVELOPMENT GOAL IS GOOD HEALTH AND WELL-BEING WHERE IT CLEARLY EMPHASIZED THAT NON-COMMUNICABLE DISEASES IS EMERGING CHALLENGE. ONE OF THE OBJECTIVES IS TO REDUCE PREMATURE MORTALITY FROM NON-COMMUNICABLE DISEASE BY THIRD IN 2030. CHRONIC KIDNEY DISEASE (CKD) IS AMONG THE SIGNIFICANT CONTRIBUTOR TO MORBIDITY AND MORTALITY FROM NON-COMMUNICABLE DISEASES THAT CAN AFFECTED 10–15% OF THE GLOBAL POPULATION. EARLY AND ACCURATE DETECTION OF THE STAGES OF CKD IS BELIEVED TO BE VITAL TO MINIMIZE IMPACTS OF PATIENT'S HEALTH COMPLICATIONS SUCH AS HYPERTENSION, ANEMIA (LOW BLOOD COUNT), MINERAL BONE DISORDER, POOR NUTRITIONAL HEALTH, ACID BASE ABNORMALITIES, AND NEUROLOGICAL COMPLICATIONS WITH TIMELY

2https://www.instagram.com/p/CrDqGVJsR--/?igshid=YmMyMTA2M2Y=

INTERVENTION THROUGH APPROPRIATE MEDICATIONS. VARIOUS RESEARCHES HAVE BEEN CARRIED OUT USING MACHINE LEARNING TECHNIQUES ON THE DETECTION OF CKD AT THE PREMATURE STAGE. THEIR FOCUS WAS NOT MAINLY ON THE SPECIFIC STAGES PREDICTION. IN THIS STUDY, BOTH BINARY AND MULTI CLASSIFICATION FOR STAGE PREDICTION HAVE BEEN CARRIED OUT. THE PREDICTION MODELS USED INCLUDE RANDOM FOREST (RF), SUPPORT VECTOR MACHINE (SVM) AND DECISION TREE (DT). ANALYSIS OF VARIANCE AND RECURSIVE FEATURE ELIMINATION USING CROSS VALIDATION HAVE BEEN APPLIED FOR FEATURE SELECTION. EVALUATION OF THE MODELS WAS DONE USING TENFOLD CROSS-VALIDATION. THE RESULTS FROM THE EXPERIMENTS INDICATED THAT RF BASED ON RECURSIVE FEATURE ELIMINATION WITH CROSS VALIDATION HAS BETTER PERFORMANCE THAN SVM AND DT

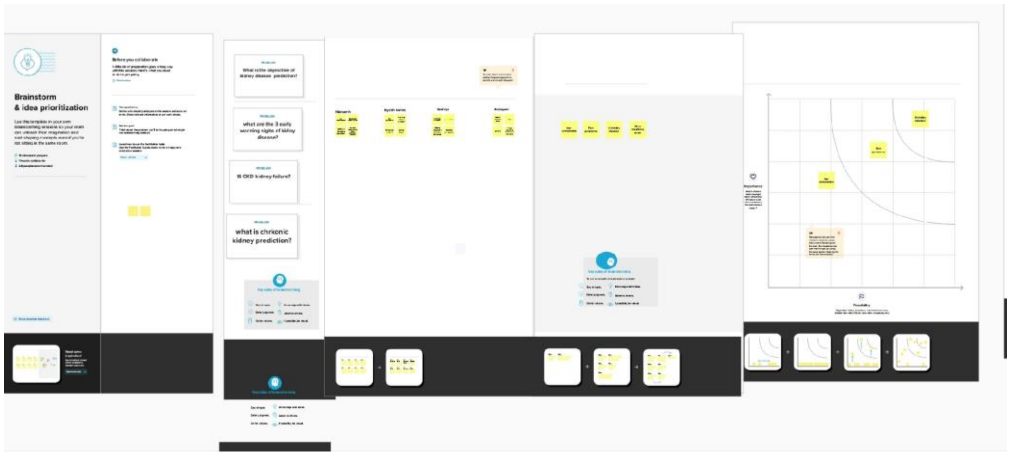
2. PROBLEM DEFINITION & DESING THINKING

2.1 EMPATHY MAP



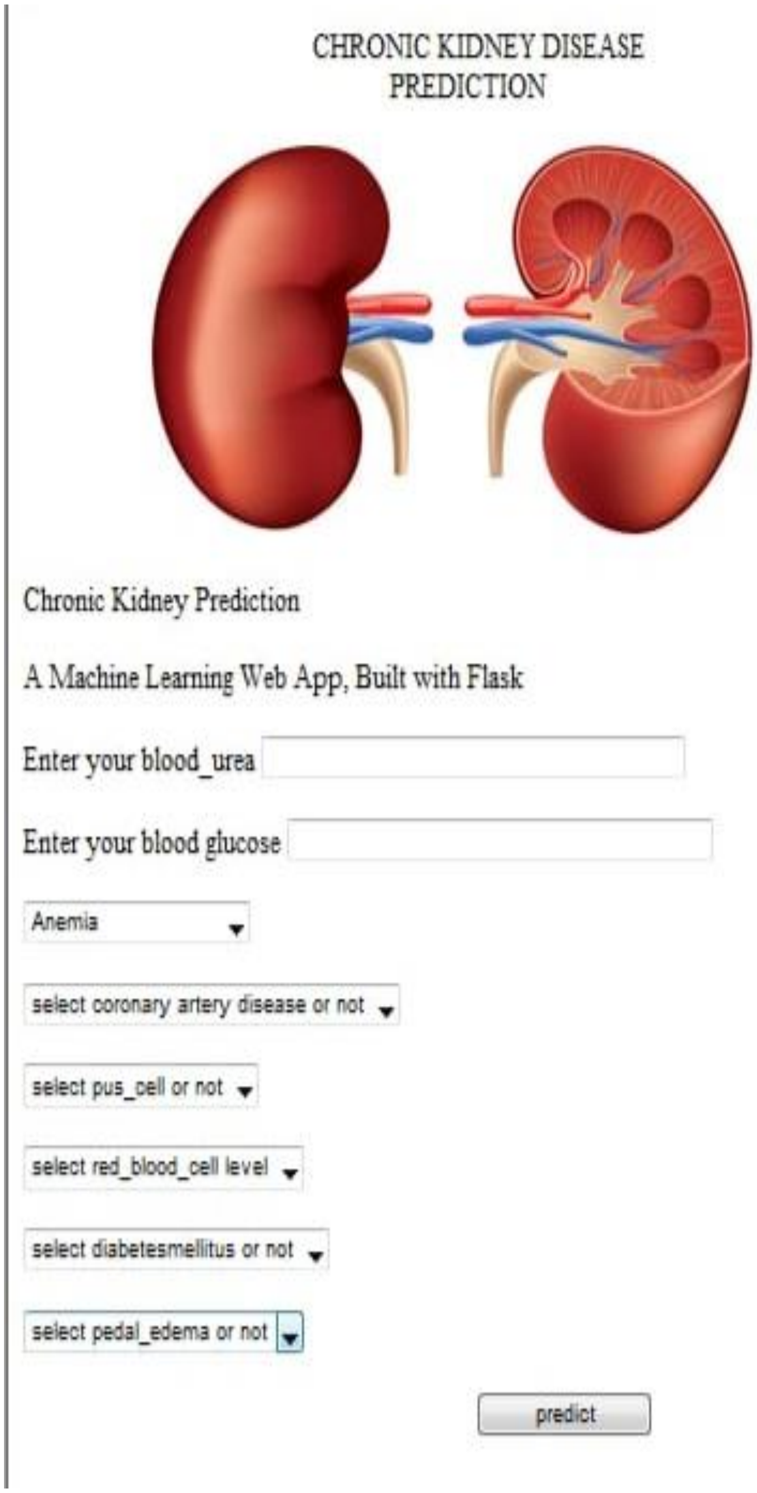
3https://www.instagram.com/p/CrDqGVJsR--/?igshid=YmMyMTA2M2Y=

3.2 IDEATION & BRAINSTORMING MAP:


Brainstorm & idea prioritization
What is the objective of kidney disease prediction?
what are the 3 early warning signs of kidney disease?
Is CKD kidney failure?
what is chronic kidney prediction?

3. WEBSITE RESULT:

4https://www.instagram.com/p/CrDqGVJsR--/?igshid=YmMyMTA2M2Y=



CHRONIC KIDNEY DISEASE PREDICTION

Chronic Kidney Prediction

A Machine Learning Web App, Built with Flask

Enter your blood_urea

Enter your blood glucose

Anemia

select coronary artery disease or not

select pus_cell or not

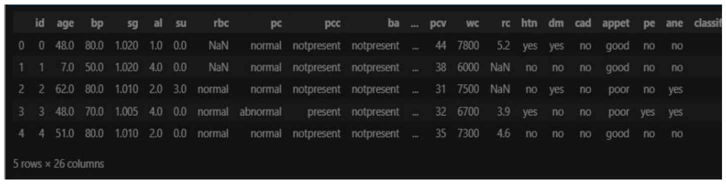
select red_blood_cell level

select diabetesmellitus or not

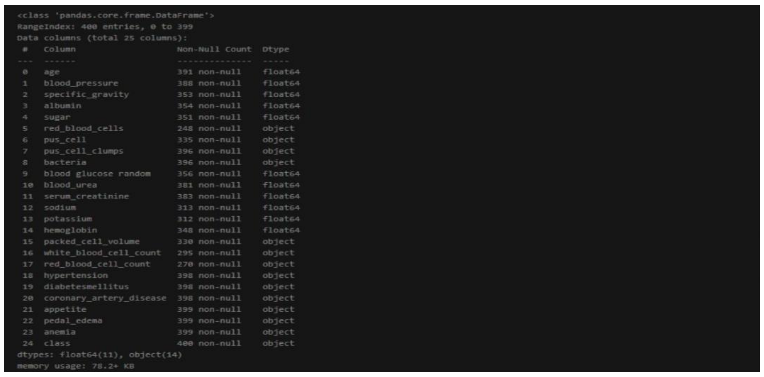
select pedal_edema or not

5https://www.instagram.com/p/CrDqGVJsR--/?igshid=YmMyMTA2M2Y=

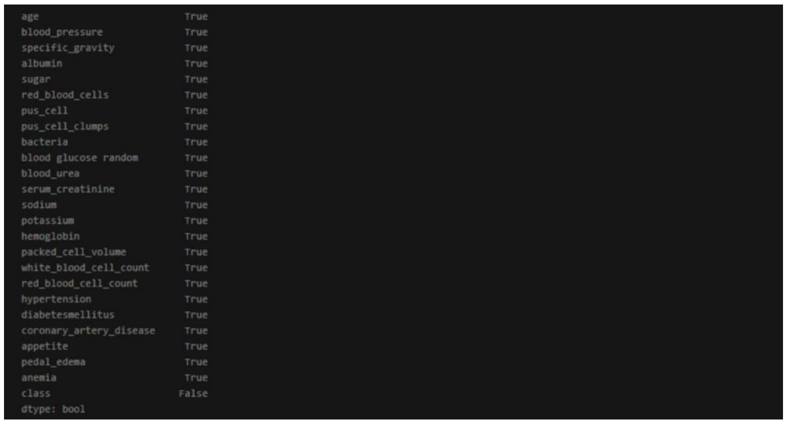
4. COLAB RESULT:



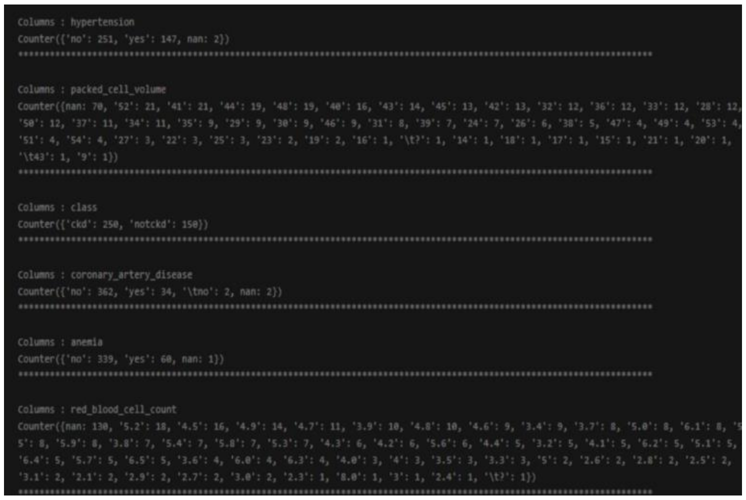
```
   id  age   bp     sg   al   su     rbc        pc         pcc          ba  ...  pcv   wc    rc  htn  dm  cad  appet  pe  ane  classif
0   0  48.0  80.0  1.020  1.0  0.0    NaN    normal  notpresent  notpresent  ...   44  7800   5.2  yes  yes  no  good  no   no
1   1   7.0  50.0  1.020  4.0  0.0    NaN    normal  notpresent  notpresent  ...   38  6000  NaN   no   no   no  good  no   no
2   2  62.0  80.0  1.010  2.0  3.0  normal   normal  notpresent  notpresent  ...   31  7500  NaN   no  yes   no  poor  no  yes
3   3  48.0  70.0  1.005  4.0  0.0  normal abnormal     present  notpresent  ...   32  6700   3.9  yes  no   no  poor yes  yes
4   4  51.0  80.0  1.010  2.0  0.0  normal   normal  notpresent  notpresent  ...   35  7300   4.6  no   no   no  good  no   no

5 rows × 26 columns
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   age                      391 non-null    float64
 1   blood_pressure           388 non-null    float64
 2   specific_gravity         353 non-null    float64
 3   albumin                  354 non-null    float64
 4   sugar                    351 non-null    float64
 5   red_blood_cells          248 non-null    object
 6   pus_cell                 335 non-null    object
 7   pus_cell_clumps          396 non-null    object
 8   bacteria                 396 non-null    object
 9   blood glucose random     356 non-null    float64
 10  blood_urea               381 non-null    float64
 11  serum_creatinine         383 non-null    float64
 12  sodium                   313 non-null    float64
 13  potassium                312 non-null    float64
 14  hemoglobin               348 non-null    float64
 15  packed_cell_volume       330 non-null    object
 16  white_blood_cell_count   295 non-null    object
 17  red_blood_cell_count     270 non-null    object
 18  hypertension             398 non-null    object
 19  diabetesmellitus         398 non-null    object
 20  coronary_artery_disease  398 non-null    object
 21  appetite                 399 non-null    object
 22  pedal_edema              399 non-null    object
 23  anemia                   399 non-null    object
 24  class                    400 non-null    object
dtypes: float64(11), object(14)
memory usage: 78.2+ KB
```

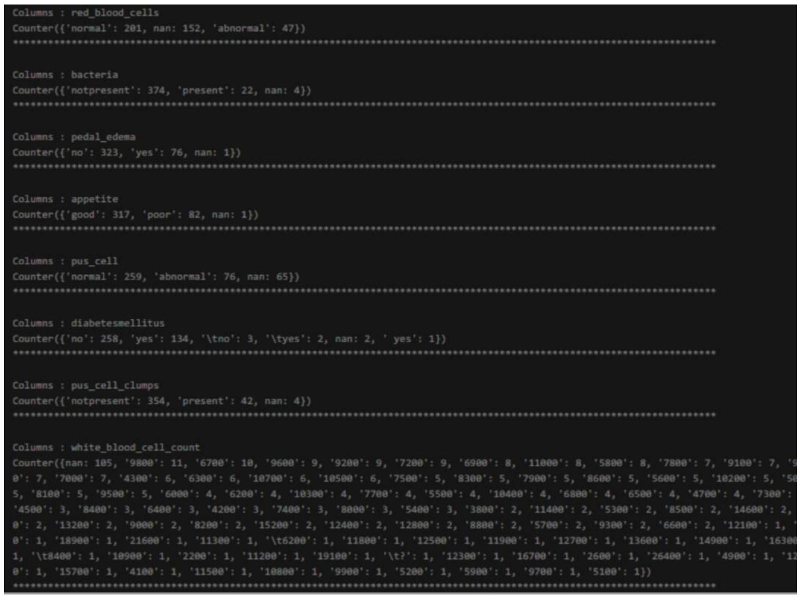


```
age                         True
blood_pressure              True
specific_gravity            True
albumin                     True
sugar                       True
red_blood_cells             True
pus_cell                    True
pus_cell_clumps             True
bacteria                    True
blood glucose random        True
blood_urea                  True
serum_creatinine            True
sodium                      True
potassium                   True
hemoglobin                  True
packed_cell_volume          True
white_blood_cell_count      True
red_blood_cell_count        True
hypertension                True
diabetesmellitus            True
coronary_artery_disease     True
appetite                    True
pedal_edema                 True
anemia                      True
class                      False
dtype: bool
```

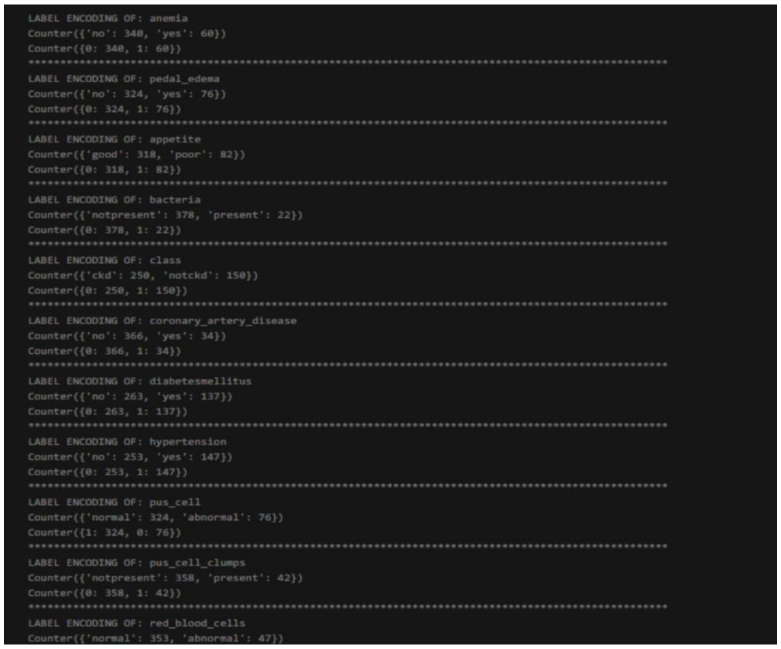


```
Columns : hypertension
Counter({'no': 251, 'yes': 147, nan: 2})
****************************************************************************************************

Columns : packed_cell_volume
Counter({nan: 70, '52': 21, '41': 21, '44': 19, '48': 19, '40': 16, '43': 14, '45': 13, '42': 13, '32': 12, '36': 12, '33': 12, '28': 12,
'50': 12, '37': 11, '34': 11, '35': 9, '29': 9, '30': 9, '46': 9, '31': 8, '39': 7, '24': 7, '26': 6, '38': 5, '47': 4, '49': 4, '53': 4,
'51': 4, '54': 4, '27': 3, '22': 3, '25': 3, '23': 2, '19': 2, '16': 1, '\t?': 1, '14': 1, '18': 1, '17': 1, '15': 1, '21': 1, '20': 1,
'\t43': 1, '9': 1})
****************************************************************************************************

Columns : class
Counter({'ckd': 250, 'notckd': 150})
****************************************************************************************************

Columns : coronary_artery_disease
Counter({'no': 362, 'yes': 34, '\tno': 2, nan: 2})
****************************************************************************************************

Columns : anemia
Counter({'no': 339, 'yes': 60, nan: 1})
****************************************************************************************************

Columns : red_blood_cell_count
Counter({nan: 130, '5.2': 18, '4.5': 16, '4.9': 14, '4.7': 11, '3.9': 10, '4.8': 10, '4.6': 9, '3.4': 9, '3.7': 8, '5.0': 8, '6.1': 8, '5
5': 8, '5.9': 8, '3.8': 7, '5.4': 7, '5.8': 7, '5.3': 7, '4.3': 6, '4.2': 6, '5.6': 6, '4.4': 5, '3.2': 5, '4.1': 5, '6.2': 5, '5.1': 5,
'6.4': 5, '5.7': 5, '6.5': 5, '3.6': 4, '6.0': 4, '6.3': 4, '4.0': 3, '4': 3, '3.5': 3, '3.3': 3, '5': 2, '2.6': 2, '2.8': 2, '2.5': 2,
'3.1': 2, '2.1': 2, '2.9': 2, '2.7': 2, '3.0': 2, '2.3': 1, '8.0': 1, '3': 1, '2.4': 1, '\t?': 1})
****************************************************************************************************
```

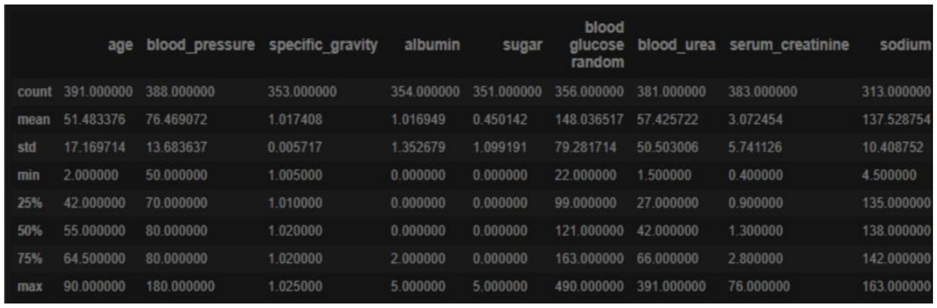
6https://www.instagram.com/p/CrDqGVJsR--/?igshid=YmMyMTA2M2Y=



```
Columns : red_blood_cells
Counter({'normal': 201, nan: 152, 'abnormal': 47})
****************************************************************************************************

Columns : bacteria
Counter({'notpresent': 374, 'present': 22, nan: 4})
****************************************************************************************************

Columns : pedal_edema
Counter({'no': 323, 'yes': 76, nan: 1})
****************************************************************************************************

Columns : appetite
Counter({'good': 317, 'poor': 82, nan: 1})
****************************************************************************************************

Columns : pus_cell
Counter({'normal': 259, 'abnormal': 76, nan: 65})
****************************************************************************************************

Columns : diabetesmellitus
Counter({'no': 258, 'yes': 134, '\tno': 3, '\tyes': 2, nan: 2, ' yes': 1})
****************************************************************************************************

Columns : pus_cell_clumps
Counter({'notpresent': 354, 'present': 42, nan: 4})
****************************************************************************************************

Columns : white_blood_cell_count
Counter({nan: 105, '9800': 11, '6700': 10, '9600': 9, '9200': 9, '7200': 9, '6900': 8, '11000': 8, '5800': 8, '7800': 7, '9100': 7, '9
0': 7, '7000': 7, '4300': 6, '6300': 6, '10700': 6, '10500': 6, '7500': 5, '8300': 5, '7900': 5, '8600': 5, '5600': 5, '10200': 5, '50
5, '8100': 5, '9500': 5, '6000': 4, '6200': 4, '10300': 4, '7700': 4, '5500': 4, '10400': 4, '6800': 4, '6500': 4, '4700': 4, '7300':
'4500': 3, '8400': 3, '6400': 3, '4200': 3, '7400': 3, '8000': 3, '5400': 3, '3800': 2, '11400': 2, '5300': 2, '8500': 2, '14600': 2,
0': 2, '13200': 2, '9000': 2, '8200': 2, '15200': 2, '12400': 2, '12800': 2, '8800': 2, '5700': 2, '9300': 2, '6600': 2, '12100': 1, '
0': 1, '18900': 1, '21600': 1, '11300': 1, '\t6200': 1, '11800': 1, '12500': 1, '11900': 1, '12700': 1, '13600': 1, '14900': 1, '1630
1, '\t8400': 1, '10900': 1, '2200': 1, '11200': 1, '19100': 1, '\t?': 1, '12300': 1, '16700': 1, '2600': 1, '26400': 1, '4900': 1, '12
0': 1, '15700': 1, '4100': 1, '11500': 1, '10800': 1, '9900': 1, '5200': 1, '5900': 1, '9700': 1, '5100': 1})
****************************************************************************************************
```



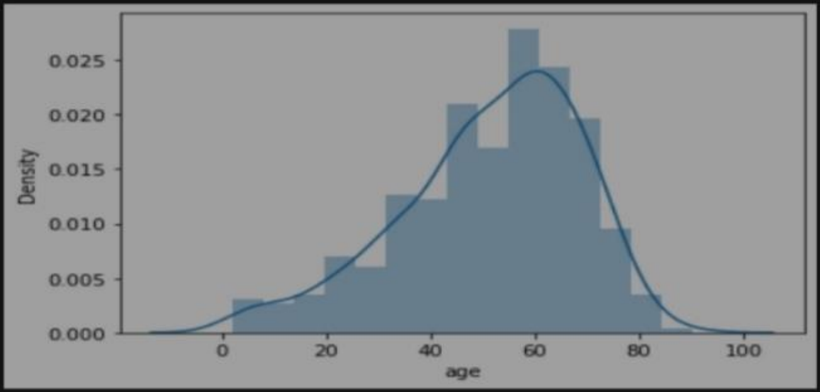
```
LABEL ENCODING OF: anemia
Counter({'no': 340, 'yes': 60})
Counter({0: 340, 1: 60})
******************************************************************************************
LABEL ENCODING OF: pedal_edema
Counter({'no': 324, 'yes': 76})
Counter({0: 324, 1: 76})
******************************************************************************************
LABEL ENCODING OF: appetite
Counter({'good': 318, 'poor': 82})
Counter({0: 318, 1: 82})
******************************************************************************************
LABEL ENCODING OF: bacteria
Counter({'notpresent': 378, 'present': 22})
Counter({0: 378, 1: 22})
******************************************************************************************
LABEL ENCODING OF: class
Counter({'ckd': 250, 'notckd': 150})
Counter({0: 250, 1: 150})
******************************************************************************************
LABEL ENCODING OF: coronary_artery_disease
Counter({'no': 366, 'yes': 34})
Counter({0: 366, 1: 34})
******************************************************************************************
LABEL ENCODING OF: diabetesmellitus
Counter({'no': 263, 'yes': 137})
Counter({0: 263, 1: 137})
******************************************************************************************
LABEL ENCODING OF: hypertension
Counter({'no': 253, 'yes': 147})
Counter({0: 253, 1: 147})
******************************************************************************************
LABEL ENCODING OF: pus_cell
Counter({'normal': 324, 'abnormal': 76})
Counter({1: 324, 0: 76})
******************************************************************************************
LABEL ENCODING OF: pus_cell_clumps
Counter({'notpresent': 358, 'present': 42})
Counter({0: 358, 1: 42})
******************************************************************************************
LABEL ENCODING OF: red_blood_cells
Counter({'normal': 353, 'abnormal': 47})
```



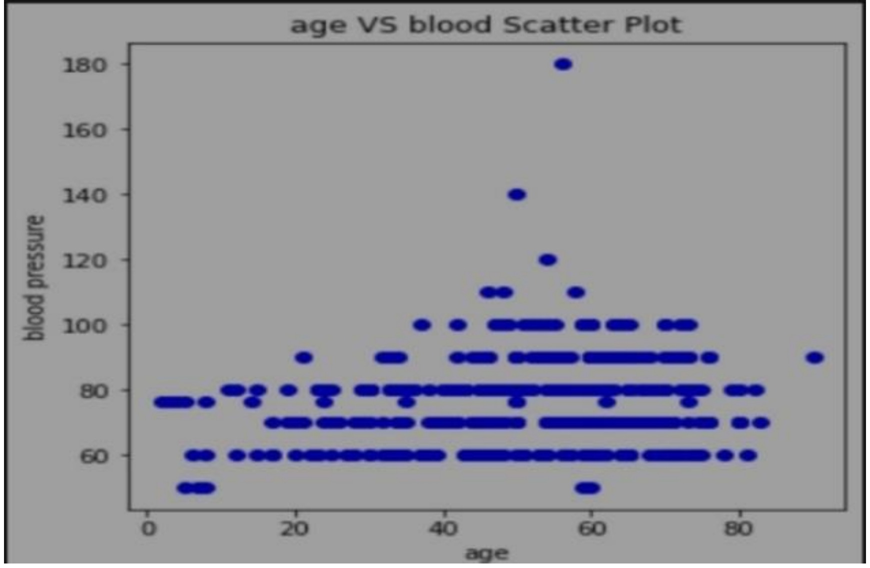
	age	blood_pressure	specific_gravity	albumin	sugar	blood glucose random	blood_urea	serum_creatinine	sodium
count	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	313.000000
mean	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754
std	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	10.408752
min	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	4.500000
25%	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	135.000000
50%	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	138.000000
75%	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	142.000000
max	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	163.000000

7https://www.instagram.com/p/CrDqGVJsR--/?igshid=YmMyMTA2M2Y=

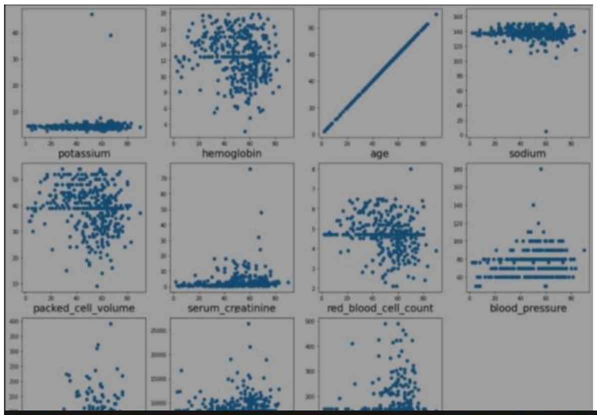
```
<AxesSubplot:xlabel='age', ylabel='Density'>
```

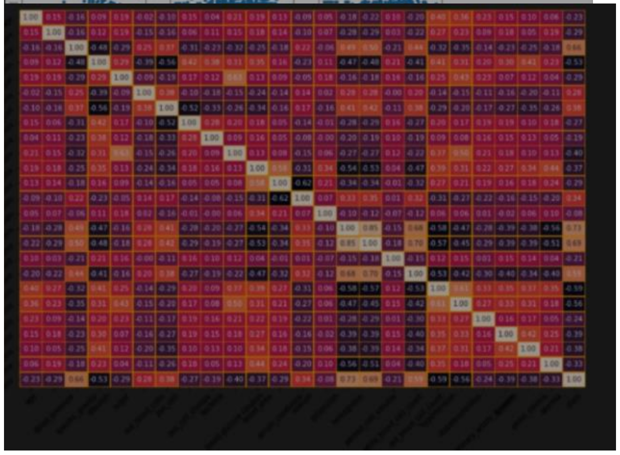


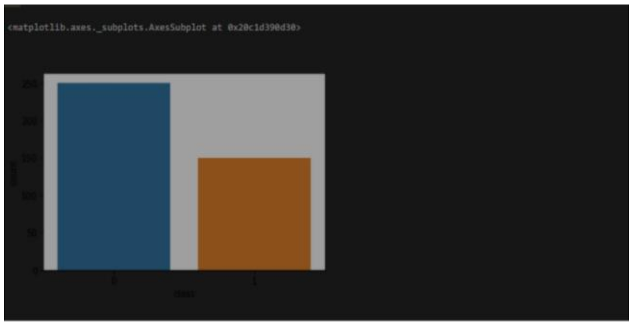
```
Text(0.5, 1.0, 'age VS blood Scatter Plot')
```



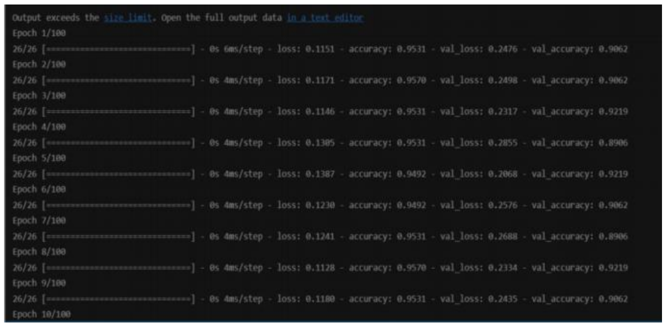
8https://www.instagram.com/p/CrDqGVJsR--/?igshid=YmMyMTA2M2Y=





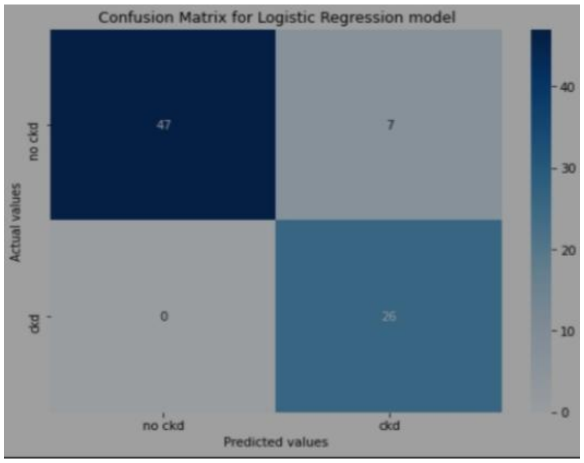


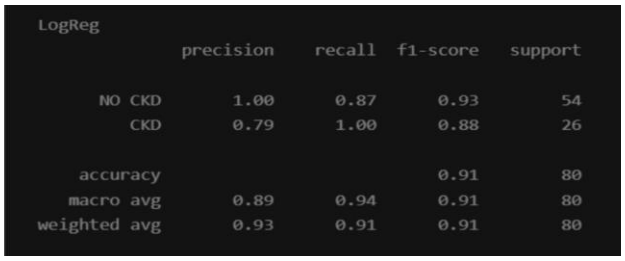
```
<matplotlib.axes._subplots.AxesSubplot at 0x20c1d390d30>
```

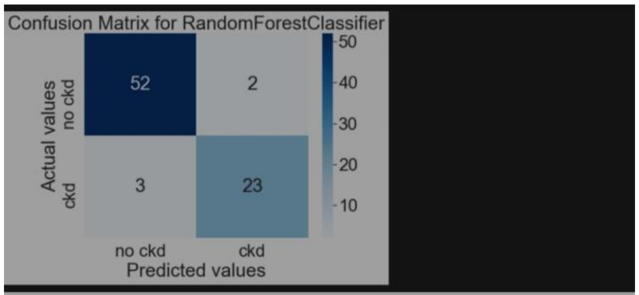


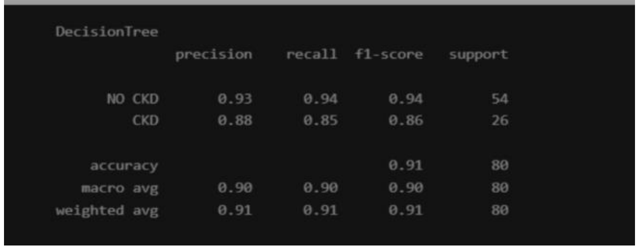
```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/100
26/26 [==============================] - 0s 6ms/step - loss: 0.1151 - accuracy: 0.9531 - val_loss: 0.2476 - val_accuracy: 0.9062
Epoch 2/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1171 - accuracy: 0.9570 - val_loss: 0.2498 - val_accuracy: 0.9062
Epoch 3/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1146 - accuracy: 0.9531 - val_loss: 0.2317 - val_accuracy: 0.9219
Epoch 4/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1305 - accuracy: 0.9531 - val_loss: 0.2855 - val_accuracy: 0.8906
Epoch 5/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1387 - accuracy: 0.9492 - val_loss: 0.2068 - val_accuracy: 0.9219
Epoch 6/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1230 - accuracy: 0.9492 - val_loss: 0.2576 - val_accuracy: 0.9062
Epoch 7/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1241 - accuracy: 0.9531 - val_loss: 0.2688 - val_accuracy: 0.8906
Epoch 8/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1128 - accuracy: 0.9570 - val_loss: 0.2334 - val_accuracy: 0.9219
Epoch 9/100
26/26 [==============================] - 0s 4ms/step - loss: 0.1180 - accuracy: 0.9531 - val_loss: 0.2435 - val_accuracy: 0.9062
Epoch 10/100
```

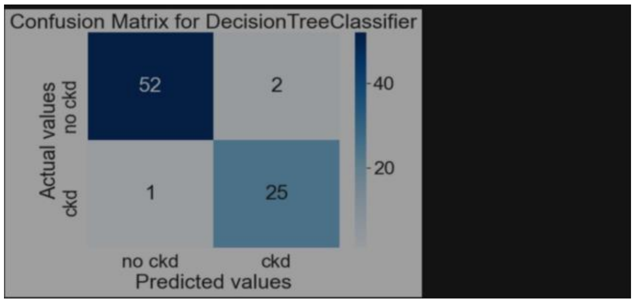
9https://www.instagram.com/p/CrDqGVJsR--/?igshid=YmMyMTA2M2Y=

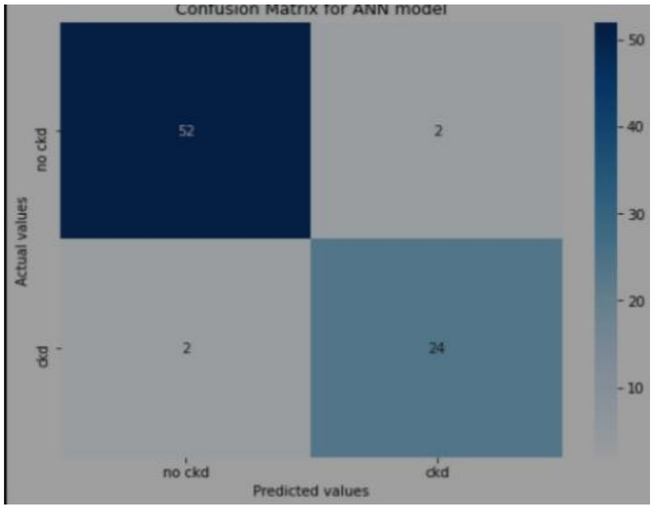


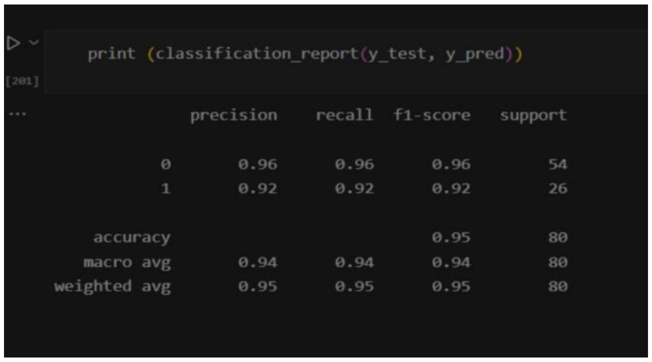

```
LogReg
              precision    recall  f1-score   support

      NO CKD       1.00      0.87      0.93        54
         CKD       0.79      1.00      0.88        26

    accuracy                           0.91        80
   macro avg       0.89      0.94      0.91        80
weighted avg       0.93      0.91      0.91        80
```



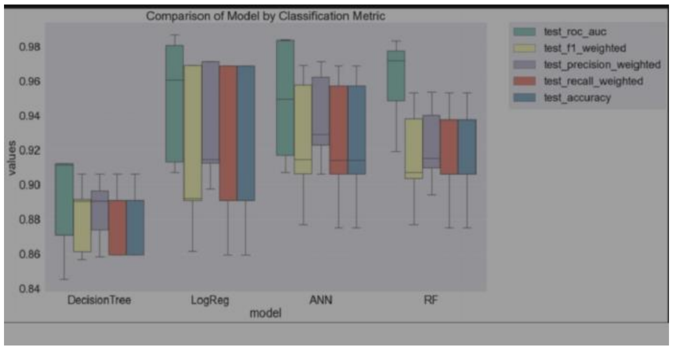

```
DecisionTree
              precision    recall  f1-score   support

      NO CKD       0.93      0.94      0.94        54
         CKD       0.88      0.85      0.86        26

    accuracy                           0.91        80
   macro avg       0.90      0.90      0.90        80
weighted avg       0.91      0.91      0.91        80
```







```
print (classification_report(y_test, y_pred))
[201]
...
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        54
           1       0.92      0.92      0.92        26

    accuracy                           0.95        80
   macro avg       0.94      0.94      0.94        80
weighted avg       0.95      0.95      0.95        80
```



11https://www.instagram.com/p/CrDqGVJsR--/?igshid=YmMyMTA2M2Y=

5. ADVANTAGE AND DISADVANTAGE :

ADVANTAGE OF PREDICTING PERSONAL LOAN:

1. APPROACH FOR PREDICTING STAGES OF CHRONIC KIDNEY DISEASE ABSTRACT CHRONIC KIDNEY DISEASE REFERS TO THE KIDNEYS HAVE BEEN DAMAGED BY CONDITIONS, SUCH AS DIABETES, GLOMERULONEPHRITIS OR HIGH BLOOD PRESSURE..

2. IT ALSO CREATES MORE POSSIBLE TO MATURE HEART AND BLOOD VESSEL DISEASE. THESE PROBLEMS MAY HAPPEN GENTLY, OVER A LONG PERIOD OF TIME, OFTEN WITHOUT ANY SYMPTOMS. .IT MAY ULTIMATELY LEAD TO KIDNEY FAILURE REQUIRING DIALYSIS OR A KIDNEY TRANSPLANT TO PRESERVE SURVIVAL TIME. SO THE EARLY DETECTION AND TREATMENT CAN PREVENT OR DEFERRAL OF THESE COMPLICATIONS.

3. ONE OF THE MAIN TASKS IS GIVING PROPER TREATMENT AND ACCURATE DIAGNOSIS OF THE DISEASE. THE MAJOR PROBLEM IS FINDING AN ACCURATE ALGORITHM WHICH DOESN'T REQUIRE LONG TIME TO RUN FOR…SHOW MORE CONTENT…

- RELATED WORK MIGUEL A ET AL [8]. PROPOSED A DISTRIBUTED APPROACH FOR THE MANAGEMENT OF ALARMS RELATED TO MONITORING CKD PATIENTS WITHIN THE ENEFRO

DISADVANTAGES OF PREDICTING PERSONAL LOAN:

1. HAVING CKD INCREASES THE CHANCES OF HAVING HEART DISEASE AND STROKE. MANAGING HIGH BLOOD PRESSURE, BLOOD SUGAR, AND CHOLESTEROL LEVELS—ALL FACTORS THAT INCREASE THE RISK FOR HEART DISEASE AND STROKE—IS VERY IMPORTANT FOR PEOPLE WITH CKD.

.

6.APPLICATIONS

Special Issue

Mental Illness Detection and Analysis on Social Media

View this Special Issue

Research Article | Open Access

Show citation

Implementation of Machine Learning Models for the Prevention of Kidney Diseases (CKD) or Their Derivatives

Khalid Twarish Alhamazani

,1Jalawi Alshudukhi

,1Saud Aljaloud

,1and Solomon Abebaw

2

Show more

Academic Editor: Deepika Koundal

Published30 Dec 2021

Abstract

Chronic kidney disease (CKD) is a global health issue with a high rate of morbidity and mortality and a high rate of disease progression. Because there are no visible symptoms in the early stages of CKD, patients frequently go unnoticed. The early detection of CKD allows patients to receive timely treatment, slowing the disease's progression. Due to its rapid recognition performance and accuracy, machine learning models can effectively assist physicians in achieving this goal. We propose a

14https://www.instagram.com/p/CrDqGVJsR--/?igshid=YmMyMTA2M2Y=

machine learning methodology for the CKD diagnosis in this paper. This information was completely anonymized. As a reference, the CRISP-DM® model (Cross industry standard process for data mining) was used. The data were processed in its entirety in the cloud on the Azure platform, where the sample data was unbalanced. Then the processes for exploration and analysis were carried out. According to what we have learned, the data were balanced using the SMOTE technique. Four matching algorithms were used after the data balancing was completed successfully. Artificial intelligence (AI) (logistic regression, decision forest, neural network, and jungle of decisions). The decision forest outperformed the other machine learning models with a score of 92%, indicating that the approach used in this study provides a good baseline for solutions in the production.

1. *Introduction*

Chronic kidney disease (CKD) is one of the leading causes of death in recent years, according to a report by the Global Burden of Disease [1]. One in every seven persons has CKD, one of the undiscovered illnesses that have the greatest influence on patients' quality of life and increase the chance of death significantly. The general system of social security in health (SGSSS) has taken chronic kidney disease (CKD) into account [2], as a high-cost pathology for generating a powerful economic impact on the finances of the system, causing a dramatic effect on the quality of life of the patient and their family, including employment repercussions. To reduce the high mortality of CKD, research should be deepened and directed to the initial stages of the disease, analyzing its risk group, with the help of laboratory tests, seeking that patients do not reach the final stages such as dialysis, transplantation, or death [3]. Through automatic learning, the aim is to find a valuable contribution so that an early classification of the disease can be carried out in its initial stages through the results of clinical laboratories, taking advantage of the great potential of automatic learning in the analysis and classification of the data. It is necessary that the technical help tools that are based on data can support the decision-making process in the initial diagnoses quickly, with high precision, and at low cost. With them, the time required for diagnosis is reduced, allowing the patient to receive treatment for the disease before it progresses to a stage of no return.

Machine learning can be broadly divided into supervised learning, unsupervised learning, and reinforcement learning [4]. Supervised learning is the most common form of machine learning used in medical research [5]. Each instance of supervised learning contains an input *object (usually a vector) and the desired output value (also known as a supervised signal) [3]. Usually, the algorithms applied for supervised learning are decision trees, naïve bayes classification, least squares regression, logistic regression, and vector support machine (SVM) methods (Classifier Sets). Recent studies show that deep neural networks have achieved comparable high performance at the expert level in natural and biomedical image classification tasks [6]. This, coupled with the ability to generate assumptions [7], the adaptability to heterogeneous data set analysis, and open-source deep learning programs that are widely disseminated, makes deep learning play an essential role in promoting medical development [8].*

This research work aims to design and implement a machine learning model that, based on data from clinical laboratories, allows predicting the possible diagnosis of CKD in its initial stages, helping reduce the mortality rate and costs for the health system.

2. *Methodology*

In the development of this project, the CRISP-DM® model [9–15] is used, which is the broadest reference guide used in the development of analytical and mining projects to data collected from clinical laboratories. For this, each of the proposed stages will be implemented.

2.1. Phase I. Understanding the Business

This phase is divided into four tasks that will help better understand the business.

Business understanding tasks.

1. Determination of Business Objectives

Chronic kidney disease (CKD) is a type of kidney disease in which there is gradual loss of kidney function over a period of months to years, Initially, there are generally no symptoms; later, symptoms may include leg swelling, feeling tired, vomiting, loss of appetite, and confusion. Complications include an increased risk of heart disease, high blood pressure, bone disease, and anaemia. CKD is associated with a decrease in kidney function related to age and is accelerated in hypertension, diabetes, obesity, and primary kidney disorders. CKD is a global health problem with a high morbidity and mortality rate, and it induces other *diseases. As there are no obvious symptoms during the early stages of CKD, patients often do not notice the disease, this being the main feature, eventually leading to a complete loss of kidney function. Early detection of CKD allows patients to receive timely treatment to improve the progression of this disease. As it has been proposed in the objectives of the work, the aim is to develop an automatic learning model for the prediction in the diagnosis of CKD and to contribute to the reduction of significant complications in the disease such as dialysis processes, kidney transplantation, or reaching death. The main criterion of success for this project, with the help of machine learning, is to identify the behaviors or behavior patterns in the initial stages of CKD to improve the quality of life of patients.*

2.. Assessment of the Situation

The idea for the approach of this project arises from the current situation regarding the increase in the confirmatory diagnosis of CKD, and lack of treatment or the user's ignorance of its pathologies leads to irreversible kidney failure in the final stages of CKD, such as dialysis for life, financially affecting the health system, as it is a costly treatment that generates the most significant amount of absorption of the resources available for health in Iraq. This could be reduced by using tools such as machine learning to classify ERC from the initial stages. Although the application of machine learning in healthcare and other areas is favorable, the field of kidney disease has not yet exploited its full potential [16–25].

2.3. Determination of the Data Mining Objectives

As referenced in the general objective, the technical terms of this project are to design, implement, and deploy a machine learning model that, based on data from clinical laboratories, allows to classify the possibility of a diagnosis of CKD. Through the analysis of laboratory studies that are low-cost for health entities, these data reduce the mortality rate and costs of the health system.

The medical history and the laboratory tests indicate identifying symptoms or signs that can be used as constitutive variables of the problem in CKD patients on a large scale since a large amount of data can be handled without inconvenience. With the initial data, a description and exploration of these are made, verifying that they can be used or have the minimum information to perform the classification, through the analysis of these data and obtain the patients with an incidence of CKD. With the data obtained, a training set is molded. Several tests are carried out that *define or determine the most appropriate technique(s) for the classifier and that the results are practical and efficient. With the defined classifier, the predictive models are trained and validated to establish the model with the highest precision for the data, selecting the one that offers the best results. Predictive models often run calculations during ongoing transactions, for example, to assess the risk or opportunity for a particular patient in a way that provides insight into the treatment .*

.

7. CONCLUSION

REAL WORLD DATA IS OFTEN INCONSISTENT WHICH CAN AFFECT THE PERFORMANCES OF MODELS. PREPROCESSING THE DATA BEFORE IT IS FED INTO CLASSIFIERS IS VITAL PART OF DEVELOPING MACHINE-LEARNING MODEL. SIMILARLY, THE DATASET FOR THIS STUDY CONTAINS MISSING VALUES THAT NEEDS TO BE HANDLED APPROPRIATELY. IT HAS TO ALSO BE IN A SUITABLE FORMAT FOR MODELING

18https://www.instagram.com/p/CrDqGVJsR--/?igshid=YmMyMTA2M2Y=

8.FUTURE SCOPE

IDENTIFY SUBSET OF RELEVANT PREDICTIVE FEATURES IS IMPORTANT FOR QUALITY RESULT [22]. FEATURE SELECTION IS THE PROCESS OF SELECTING MOST IMPORTANT PREDICTIVE FEATURES TO USE THEM AS INPUT FOR MODELS. IT IS IMPORTANT PREPROCESSING STEP TO DEAL WITH THE PROBLEM OF HIGH DIMENSIONALITY. HENCE, THE MAIN AIM OF FEATURE SELECTION IS TO SELECT THE SUBSET OF FEATURES THAT ARE RELEVANT AND INDEPENDENT OF EACH OTHER FOR TRAINING THE MODEL [23]. SIMILARLY, FEATURE SELECTION IS CRUCIAL TO DEVELOP CHRONIC KIDNEY DISEASE PREDICTIVE MODEL. THIS REDUCES THE DIMENSIONALITY AND COMPLEXITY OF THE DATA AND MAKES THE MODEL BE FASTER, MORE EFFECTIVE AND ACCURATE. HENCE, FEATURE SELECTION ALGORITHM HAVE BEEN USED TO SELECT RELEVANT FEATURES AFTER THE CONSTRUCTION OF THE DATASET.

.

9. APPENDIX

ACCURACY IMPLIES THE ABILITY OF THE CLASSIFICATION ALGORITHM TO PREDICT THE CLASSES OF THE DATASET CORRECTLY. IT IS THE MEASURE OF HOW CLOSE OR NEAR THE PREDICTED VALUE IS TO THE ACTUAL OR THEORETICAL VALUE [35]. GENERALLY, ACCURACY IS THE MEASURE OF THE RATIO OF CORRECT PREDICTIONS OVER THE TOTAL NUMBER OF INSTANCES. THE EQUATION OF ACCURACY IS SHOWN IN EQ. 3.

ACCURACY=TP+TNTP+FP+TN+FN

SOURCE CODE:

Import the libraries

```
import pandas as pd
import numpy as np

from collections import Counter as c

import matplotlib.pyplot as plt

import seaborn as sns

import missingno as msno

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LogisticRegression

import pickle
```

Read the Dataset

```
data=pd.red_csv("chronickidneydisese.csv")
dat.head()
```

Rename the columns

```
data.columns
data.columns=['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',
 'hemoglobin', 'packed_cell_volume", 'white_blood_cell_count', 'red_blood_cell_count,
 'hypertension', 'diabetesmellitus', 'coronary artery_disease', 'appetite',
```

'pedal_edema', 'anemia', 'class']

data.columns

Handling missing values

data.info()

data.isnull().ary()

data['blood glucose random'].fillna(data['blood glucose random'].mean(), inplace=True)

data['blood pressure'].fillna(data['blood pressure'].mean(), inplace=True) data['blood urea']. fillna(data['blood_urea"].mean(), inplace=True)

data['hemoglobin']. fillna(data['hemoglobin'].mean(), inplace=True)

data['packed_cell_volume'].fillna(data['packed_cell_volume'].mean(), inplace=True)

data['potassium].fillna(data['potassium'].mean(), inplace=True) data['red_blood_cell_count].fillna(data['red_blood_cell_count'].mean(), inplace=True)

data['serum creatinine'].fillna(data['serum_creatinine'].mean(), inplace=True)

data['sodium'].fillna(data['sodium'].mean(), inplace=True)

data['white_blood_cell_count].fillna(data['white_blood_cell_count'].mean(), inplace=True)

data['age'].fillna(data['age*].mode()[0], inplace=Trum)

data['hypertension'].fillna(data['hypertension].mode()[0],inplace=True)

data['pus_cell_clumps'], fillna(data['pus_cell_clumps'].mode()[0],inplace=True)

data['appetite'].fillna(data['appetite'].mode()[@]}, inplace=True) data['albumin].fillna(data['albumin]. mode()[0], inplace=True)

data['pus_cell'].fillna(data['pus_cell'].mode()[0], inplace=True) data['red blood cells'].fillna(data['red_blood cells'].mode()[0],inplace=True)

```
data['coronary artery disease'].fillna(data['coronary artery disease'], mode()[0], inplace=True)

data['bacteria].fillna(data['bacteria'].mode()[0],inplace=True)

data['anemia"].fillna(data['anemia].mode()[0], inplace=True)

data['sugar'].fillna(data['sugar'].mode()[0], inplace=True)

data['diabetes mellitus'], fillna(data['diabetes mellitus ]. mode()[0],inplace=True)

data['pedal_edema"].fillna(data['pedal_edema].mode()[0],inplace=True)

data[ 'specific gravity'].fillna(data[ 'specific gravity'].mode()[0],inplace=True)
```

Handling Categorical columns

```
catcols=set(data.dtypes[data.dtypes=='0'].index.values)
print(catcols)
for i in catcols:
   print("Columns :",i)
   print(c(data[i]))
   print('*'*120+'\n')
catcols.remove('red_blood_cell_count')
catcols.remove('packed_cell_volume')
catcols.revove('white_blood_cell_count')
print(catcols)
```

Labeling Encoding of Categorical Column

```
catcols=['anemia', 'pedal edema", "appetite', 'bacteria', 'class', 'coronary artery_disease", "hypertension", "pus_cell', 'pus_cell_clumps', 'red_blood_cells']

from sklearn.preprocessing import LabelEncoder
```

```
for i in catcols:

print("LABEL ENCODING OF:",1)

LEi = LabelEncoder() creating an object of LobelEncoder

print(c(data[i])) n

data[i] = LEi.fit_transform(data[i])

print(c(data[i]))

print("*"*100)
```

Handling Numerical columns

```
controls=set(data.dtypes[data.dtypes!='0'].index values]
print(contcols)

for i in (contcols)
  print("Continous Columns :",i)
  print(c(data[i])
print('*'*120+'\n')

contcols.remove(specific_gravity')
contcols.remove('albumin')
cokntcols.remove('sugar')
print(contcols)

contcols.add("red_blood_cell_count')

contcols.add('packed_cell_volume')

contcols.add('white_blood_cell_count')

print(contcols)
```

catcols.add('specific gravity')

catcols.add('albumin')

catcols.add('sugar')

print(catcols)

data['coronary artery disease'] = data.coronary_artery_disease.replace('\tno','no'

c(data["cormmary artery disease'])

data[diabetesallitus'] = data.diabetes mellitus.replace(to replace={'\tno':"no','\tyes':'yes','yes':}
c(data['diabetesmellitus'])

Milestone 3: Exploratory Data Analysis

Descriptive statistical Analysis

data.describe()

Univariate analysis
Age distribution

sns.distplot(data.age)

Bivariate analysis
Age vs Blood Pressure

import matplotlib.pyplot as plt
fig-plt.figure(figsize=(5,5))
plt.scatter(data['age'],data['blood pressure'],color='blue')
plt.scatter(data['age'],
plt.ylabel('blood pressure')

```
plt.title("age vs blood Scatter Plot")
```

Multivariate analysis

Age vs all continous columns

```
plt.figure(figsize-(20,15), facecolor="white")
plotnumber = 1

for column in contcols:
  if plotnumber<-11:
ax- plt.subplot(3,4,plotnumber)
plt.scatter(data['age'],data[column])
plt.xlabel(column,fontsize=20)
plotnumber+=1
plt.show()



f,ax-plt.subplots(figsize=(18,10))

sns.heatmap(data.corr(), annot=True, fmt=".2f", ",ax=ax,linewidths=0.5,linecolor="orange")

plt.xticks(rotation=45)

plt.yticks(rotation=45)

plt.show()

sns.countplot(data['class'])

sklearn.preprocessing import StandardScaler
sc=StandardScalser()
x_bal=sc.fit_transform(x)
```

Creating Independent and Dependent

```
selcals-["red_blood_cells', 'pus_cell', 'blood glucose random", "blood_area", "pedal edema", "anemia', 'diabetes mellitus', 'coronary artery disease")

x-pd.DataFrame(data,columns-selcols)

y-pd.DataFrame(data,columns-["class"}}

print(x.shape)

print(y.shape)
```

Spalitting the data into train and test

```
from sklearn.model_selection import train_test_split

x_train_x_test,y_train,y test-train test split(x,y.test_size=0.2,random_state=2)
```

Milestone 4: Model Building

ANN Model

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

classification = Sequential()

classification.add (Dense (30, activation='relu'))

classification.add(Dense (128, activation='relu')) classification.add(Dense (64, activation='relu'))

classification.add(Dense(32, activation='relu'))
```

classification.add(Dense (1, activation='sigmoid'))

classification.compile(optimizer "adam,losse binary_crossentropy", metrics=['accuracy'])

classification.fit(x_train,y train,batch size-10,validation split-0.2,epochis=100)

Random Forest model

from sklearn.ensemble import RandomForestclassifier rfc - RandomForestclassifier(n_estimators-10, criterion-entropy")

rfc.fit(x_train,y_train)

<ipython-input-255-b87bb2ba9825>:1: DataConversionWarning: A column-vecyor y (n_samples,), for example using ravel().

rfc.fit(x_train,y_train)

RandomForestClassifier(criterion="entropy', n_estimators=10)

y_predict = rfc.predict(x_test)

y_predict_train= rfc.predict(x_train)

Decision tree model

From sklearn.free import Decisiontreeclassifier

dtc - DecisionTreeclassifier(maxdepth-4,splitter-"best",criterion-entropy")

dtc.fit(x_train,y_train)

DecisionTreeclassifier(criterion='entropy', max_depth=4)

y_predict dtc.predict(x_test)

y_predict

Logistic Regression

from sklearn.linear model import LogisticRegression

Igr Logistickegression() 1gr.fit(x train,y train)

C:\Users\Saumya\Anaconda\lib\site-packages\sklearn\utils\validation.py:72: DataConversionkar Please
charge the shape of y to (n samples,), for example using ravel(). return (**kwargs)

LogisticRegression()

Predicting our output with the model which we build

from sklearn.metrics import accuracy score,classification_report

y_predict = Igr.predict(x_ test)

Testing the model

y_pred- 1gr.predict(((3.1.121.000000,36,0,0,0,1,011)

print(y_pred)
(y_prod)

y_pred = dtc.predict({|1,1,121.000000, 36.0,0,0,1,0;])

print(y_pred)
(y_peed)

y_pred= rfi.predict([[1,1,121.000000赂36,0,0,0,1,0)])

```
print(y_pred).
(y_peed)

classification.save("ckd.h5")


y_pred=classification.predict(x_test)

y_pred

y_pred= (y_pred>0.5)
y_pred

def predict_exit(sample_value):

# Convert list to numpy array sample_value = np.array(sample_value)

# Reshape because sample value contains only 1 record sample_value = sample_value.reshape(1, -1)

# Feature Scaling sample_value = sc.transform(sample_value)

return classifier.predict(sample value)

test-classification.predict([[1,1,121.000000, 36.0,0,0,1,0]]) if test==1: print("Prediction: High chance of CKD!")
else: print("Prediction: Low chance of CKD.')
```

```
Prediction: Low chance of CKD
```

Milestone 5: Performance Testing & Evaluate the results

Testing model with multiple evaluation metrics

```
from sklearn import model selection
```

models

Clogg, Logistickegression()), CRF RandomForestclassifier());

(Decisionfree,DecisionTreeclassifier()),

results = []

names-13

scoring = "accuracy, precision weighted",

"recall weighted", "EX_weighted", "roc_and]

target names['HO CI", "CD]

for

name, model in models:

kfold model selection.Fold(n splits-5, shuffle-True, random state-90210) scoring-scorin

cv_results - model selection.cross_validate(model, x _train, y train, cu-kfold, clf-model.fit(x train, y train)

y_predclf.predict(x_test)

print (name) print(classification_report(y_test, y pred, target_names-Target_names})

results.append(cv results)

names.append(name)) this df = pd.DataFrame(_results)

this dr model] = name

dis.append(this_df)

final-pd.concat(dfs, Ignore Index-True)

return final

Making the Confeston Matrix

from sklearn.metrics import confusion_matrix

cm=confusion matrixly test, y_predict)

cm

pit.figure(figsize-(8,6)

sns.figure(Cm, cmap= Bus,annot='blue',annot=true, sticklabels-('c', ''), yticklabels- 'no du", "d"])

plt.xlabel(Predicted values")

plt.ylabel('Actual values")

pit.title("Confusion Matrix f眉r Ingistic Regression mode]\"}

plt.show()

Making the Confusion Matrix sklearn.metrics Import confusion matrix confusion matrix(y_test, y predict)

array([[52, 21],

[1, 25]], dtype-int64)

Plotting confusion satis

plt.figure(figsize=(8,6))

plt.xlabel(Predicted values plt.ylabel('Actual values')

plt.title("Confusion Matrix for DecisionTreeClassifier")

plt.show()

Hilding the Confustile Matrix

```
from sklearn.metrics import confusion matrix cm confusion matrix(y_test, y pred)

array([[52,

21. [2, 24]],

dtype-int:54)
```

Plekting confusion matrix plt.figure(figsize-(8,6))

```
sns.beatmap(ch, chap-"Bldes, unnot Tru, xticklabels=['nockd", "cid"), yticklabels=['mocki鈥, Teku#!)))

plt.xlabel("Predicted values")

plt.ylabel('Actual values)

pit.title Confusion Matrix for ANN model)
plt.show()
```

Evaluae the results

```
bootstraps=[]

for model in list(set(final,model.values)); model_df- final.loc[final.model -model]
bootstraps.append(bootstrap)

pootstrap model_df.sample(n-30, replace=True)

strap_df- pd.concat(bootstraps, ignore_index-Trum)
```

Its long pd.melt (bootstrap_df, ld_vars['model'], var_name-metrics, value_name="values") metrics =
['fit time, score time'] fit time metries

REGRIANCE HETRICS Its_long_nofit results_long.loc["results_long trics].isin(time_metrics)] get of
without fit data

its_long_nofit results_long_nofit.sort_values(by "values")

Its long fit = results long.loc[results_long['metrics'].isin(time_metrics)] = df with fit data. Its_long_fit = results long fit.sort_values(by="values")

mport matplotlib.pyplot as plt

Import seaborn as sns plt.figure(figsize=(20, 12))

sns.set(Font scale-2.5) E = sns.boxplot(x="model", y="values", hue="metrics", data-results long_nofit, palette="Set3")

plt.legend (bbox_to_anchor-(1.05, 1), loc-2, borderaxespad-.)

plt.title("Comparison of Model by Classification Metric")

plt.savefig("//benchmark_models_performance.png",dpi=300)|

Milestone6: Model Deployment

Save the best model

pickle.dump(1gr,open('CKD.pkl','wb'))

35https://www.instagram.com/p/CrDqGVJsR--/?igshid=YmMyMTA2M2Y=