Early Prediction For Chronic Kidney Disease Detection:

A Progressive Approach To Health Management

1.INTRODUCTION

1.1 Overview

Chronic kidney disease (CKD) is a major burden on the healthcare system because of its increasing prevalence, high risk of progression to end-stage renal disease, and poor morbidity and mortality prognosis. It is rapidly becoming a global health crisis. Unhealthy dietary habits and insufficient water consumption are significant contributors to this disease. Without kidneys, a person can only live for 18 days on average, requiring kidney transplantation and dialysis. It is critical to have reliable techniques at predicting CKD in its early stages. Machine learning (ML) techniques are excellent in predicting CKD. The current study offers a methodology for predicting CKD status using clinical data, which incorporates data pre-processing, a technique for managing missing values, data aggregation, and feature extraction. A number of physiological variables, as well as ML techniques such as logistic regression (LR), decision tree (DT) classification, and -nearest neighbour (KNN), were used in this work to train three distinct models for reliable prediction.

2.LITERATURE SURVEY

2.1 Existing problem

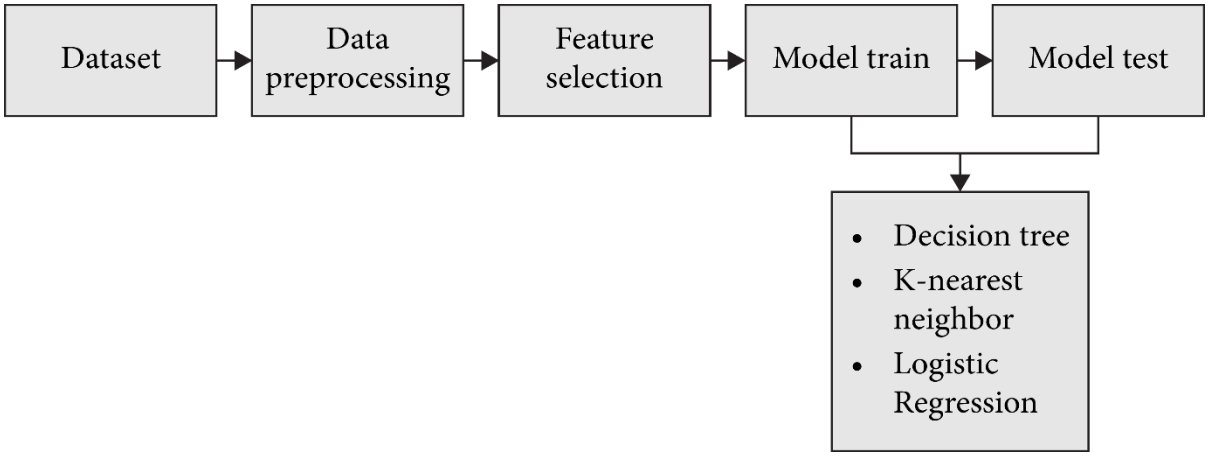
From last 15 years data it has been noticed that increase in number of patient which is suffering from CKD disease and more than 60% patients are not receiving medical attention. CKD ranks number 27 and 18 in 1990 and 2010 respectively as world's prime reason of death. 956,000 people died in 2013 because of CKD. At the last stage, the patient must take dialysis or do kidney transplantation. One of the best ways to reduce this death rate is early treatment. But in developing countries, patients take treatment when they reached in serious state.

2.2 Proposed solution

We are using mean, mode and median based pre-processing techniques for the missing values. Further, we have used K-Nearest Neighbour Classifier, Decision Tree Classifier, Gaussian Naïve Bayes, Logical Regression and Artificial Neural Network to train the model. Then, based on the results of each of these Machine Learning Methods, we can compare and determine which among the following methods can predict the possibility of CKD most accurately.

3.THEORETICAL ANALYSIS

3.1 Block diagram



3.2 Hardware/Software designing

Hardware Requirements:

• Processor: 2 gigahertz (GHz) or faster processor or SoC.

• RAM: 8 gigabyte (GB) for 32-bit or 8 GB for 64-bit.

• Hard disk space: =16GB.

Software Requirements:

• Operating System: Windows XP/7/8/8.1/10, Linux and Mac

• Coding Language: Python

• Tools:

1. Pandas

2. Numpy

3. Tensorflow

4. Keras

5. Sickitlearn

4.EXPERIMENTAL INVESTIGATIONS

One of the first steps we perform during implementation is an analysis of the data. This was done by us in an attempt to find the presence of any relationships between the various attributes present in the dataset.

Acquisition of Training Dataset: The Training data set was acquired from the UCI Repository for kidney disease. The dataset was collected from a number of hospitals from Tamil Nadu and the values are actual test results values that were obtained. We have a total of 24 attributes which make up the dataset but on pre-processing it was found that only 6 of the 24 are important in determining that relationship. There are a total of 400 samples. This way the predictions can be used to correctly determine early detection of Chronic Kidney Disease.

Data Preprocessing: After analyzing and visualizing the data, the next step is preprocessing. Data preprocessing is an important step as it helps in cleaning the data and making it suitable for use in machine learning algorithms. Most of the focus in preprocessing is to remove any outliers or erroneous data, as well as handling any missing values. Missing data can be dealt with in two ways. The first method is to simply remove the entire row which contains the missing or erroneous value. While this an easy to execute method, it is better to use only on large datasets. Using this method on small datasets can reduce the dataset size too much, especially if there are a lot of missing values. This can severely affect the accuracy of the result. Since ours is a relatively small dataset, we will not be using this method. Instead we would be filling our missing values with average or mode of the column based on the type of attribute. If the attribute is nominal than we will use the average and if it is non-nominal we would be using the mode. The dataset that we used had values that were in string format so we had to transform and encode the into integer valued so as to pass as an input to the neural network. First we converted the data into pandas categorical data and create a separated dataframes.

As we all know out of all the 24 attributes not all will directly affect the result so in order to find the level of dependence we used a tree method called Extra tree Classifier to find the level of dependency and came to the conclusion that only 6 attributes namely Al, Sg, HTN, pcv, hemo, dm are the ones which actually affect the final result the most.

Machine Learning Models:

1. Decision Tree: Regression and classification problems. The general objective of using Decision Tree is to create a model that predicts classes or values of target variables by generating decision rules derived from training data sets. Decision tree algorithm follows a tree structure with roots, branches and leaves. The attributes of decision making are the internal nodes and class labels are represented as leaf nodes. Decision Tree algorithm is easy to understand compared with other classification algorithms.

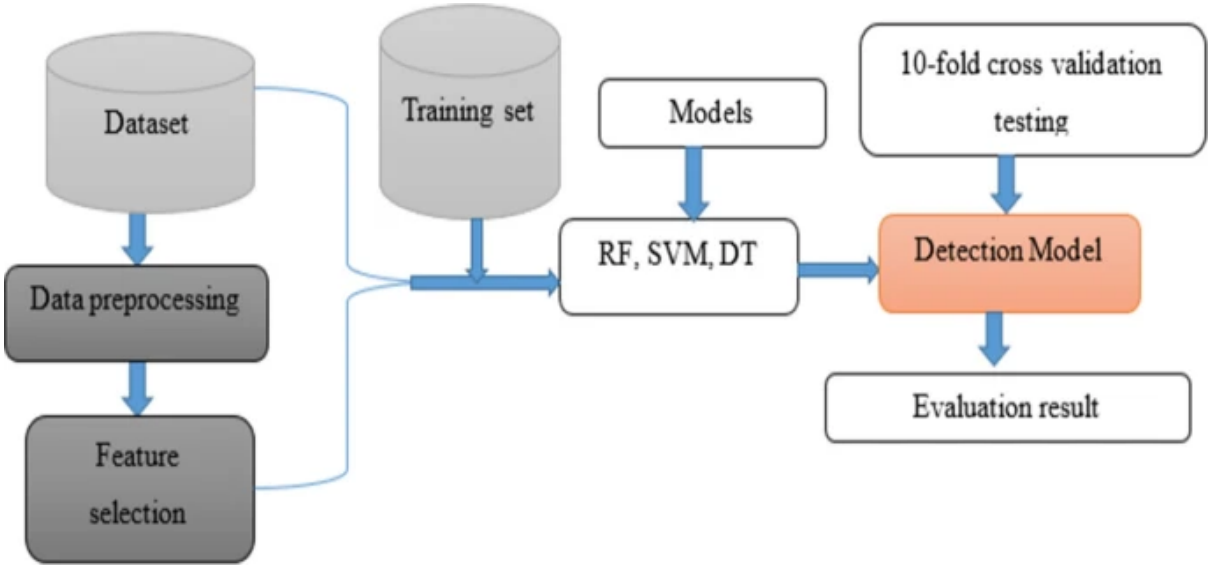
2. KNN (K-nearest Neighbor): In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression.[1] In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression. In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the

object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. In K-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors. K-NN is a type of instance based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until function evaluation

3. Gaussian NB: In machine learning, naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models. But they could be coupled with Kernel density estimation and achieve higher accuracy levels. When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution.

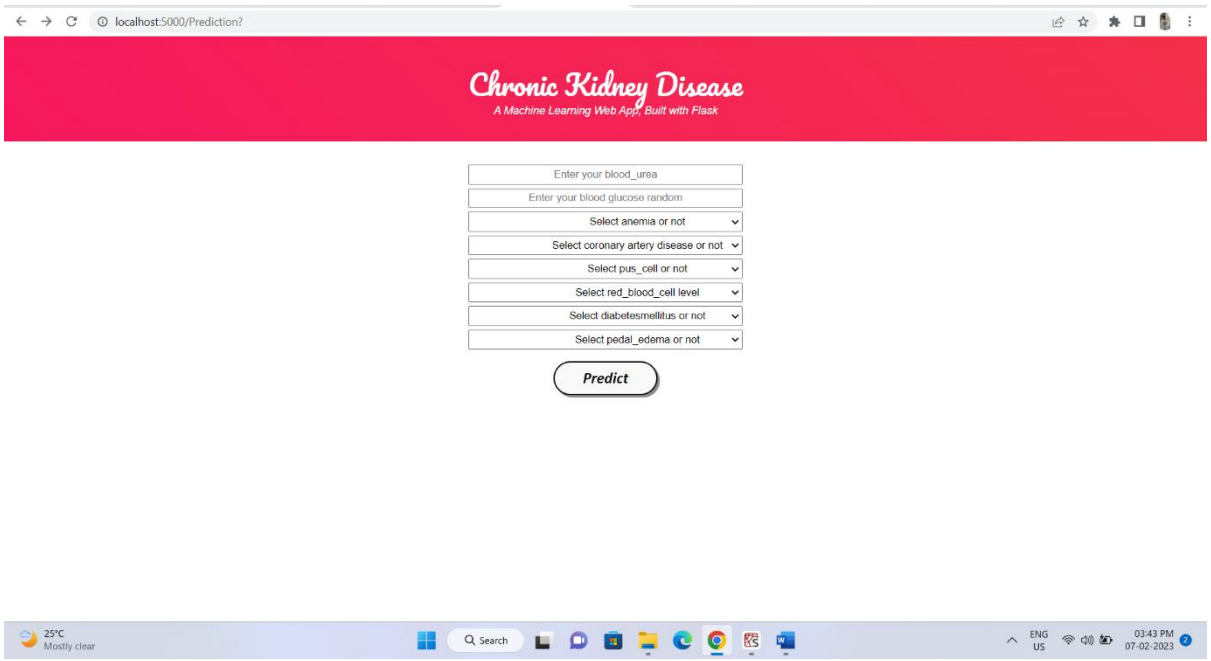
4. Logistic Regression: Logistic Regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). The logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1 and the sum adding to one.

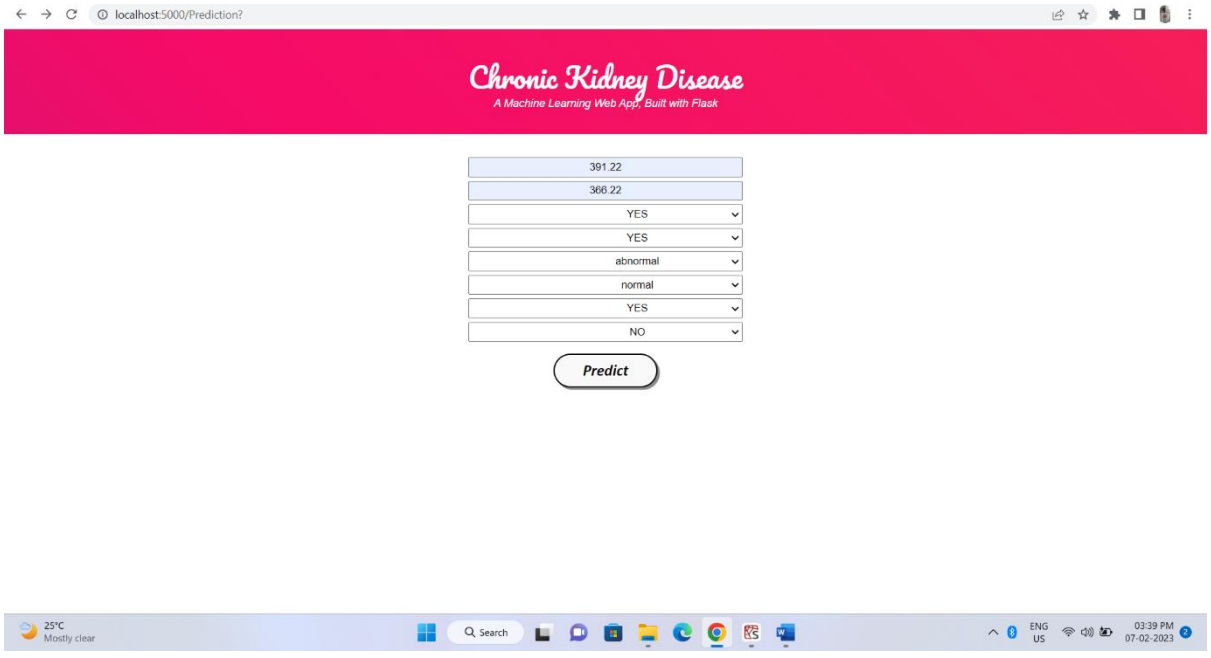
5.FLOWCHART



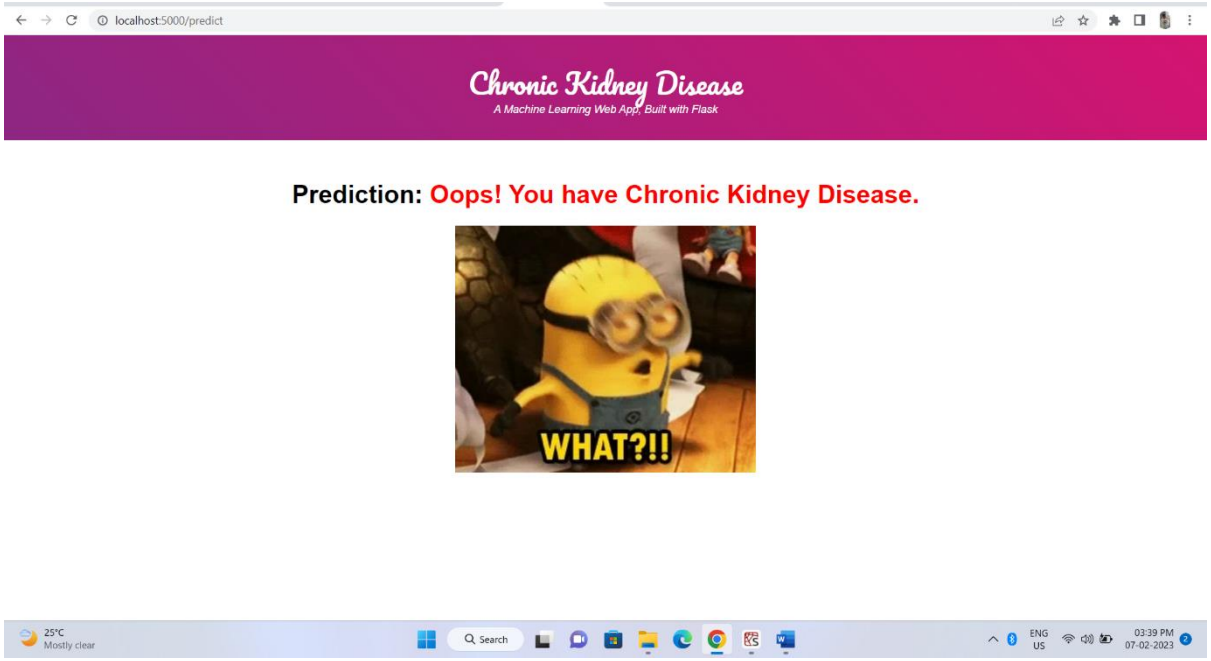
6.RESULT

Input:





Output:



7.ADVANTAGES AND DISADVANTAGES

- **Advantages**
  - 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.
- **Disadvantages**
  - The accuracy of the model depends on the values supplied by the user, if the values are inaccurate the result will also be inaccurate.

8.APPLICATIONS

1. Healthcare & Hospitals
2. Help train medical students

9.CONCLUSION

This system presented the best prediction algorithm to predict CKD at an early stage. The dataset shows input parameters collected from the CKD patients and the models are trained and validated for the given input parameters. K-Nearest-Neighbors Classifier, Decision Tree Classifier, GaussianNB, Logical Regression and Artificial Neural Network learning models are

constructed to carry out the diagnosis of CKD. The performance of the models is evaluated based on a variety of comparison metrics are being used, namely Accuracy, Specificity, Sensitivity and Log Loss. The results of the research showed that Logical Regression model better predicts CKD in comparison to the other models taking all the metrics under consideration. This system would help detect the chances of a person having CKD further on in his life which would be really helpful and cost-effective people. This model could be integrated with normal blood report generation, which could automatically flag out if there is a person at risk. Patients would not have to go to a doctor unless they are flagged by the algorithms. This would make it cheaper and easier for the modern busy person.

10.FUTURE SCOPE

- This would help detect the chances of a person having CKD further on in his life which would be really helpful and cost-effective people.
- This model could be integrated with normal blood report generation, which could automatically flag out if there is a person at risk.
- Patients would not have to go to a doctor unless they are flagged by the algorithms. This would make it cheaper and easier for the modern busy person.

11.BIBLIOGRAPHY

- https://www.kidney.org/atoz/content/about-chronic-kidney-disease
- https://www.niddk.nih.gov/health-information/kidney-disease/chronic-kidney-disease-ckd
- https://smartinternz.com/
- https://smartbridge.teachable.com/

APPENDIX

A. Source Code:

App.py

```
# importing the necessary dependencies
import numpy as np
import pandas as pd
from flask import Flask, request, render_template
import pickle

app = Flask(__name__) # initializing a flask app
model = pickle.load(open('lgr.pkl', 'rb')) #loading the model

@app.route('/')# route to display the home page
def home():
```

```
    return render_template('home.html') #rendering the home page
@app.route('/Prediction',methods=['POST','GET'])
def prediction():
    return render_template('indexnew.html')
@app.route('/Home',methods=['POST','GET'])
def my_home():
    return render_template('home.html')

@app.route('/predict',methods=['POST'])# route to show the predictions in a web UI
def predict():

    #reading the inputs given by the user
    input_features = [float(x) for x in request.form.values()]
    features_value = [np.array(input_features)]

    features_name = ['blood_urea', 'blood glucose random', 'anemia',
      'coronary_artery_disease', 'pus_cell', 'red_blood_cells',
      'diabetesmellitus', 'pedal_edema']

    df = pd.DataFrame(features_value, columns=features_name)

    # predictions using the loaded model file
    output = model.predict(df)

    # showing the prediction results in a UI# showing the prediction results in a UI
    return render_template('result.html', prediction_text=output)

if __name__ == '__main__':
    # running the app
    app.run(debug=False)
```

Filling null values and label encoding:

```
data['blood glucose random'].fillna(data['blood glucose
random'].mean(),inplace=True)
data['blood_pressure'].fillna(data['blood_pressure'].mean(),inplace=Tr ue)
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(),in place=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(),inplac e=True)
data['sodium'].fillna(data['sodium'].mean(),inplace=True)
data['white_blood_cell_count'].fillna(data['white_blood_cell_count'].m
```

```
ean(),inplace=True) data['age'].fillna(data['age'].mode()[0],inplace=True)
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()[0],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['diabetesmellitus'].fillna(data['diabetesmellitus'].mode() [0],inplace=True)
data['pedal_edema'].fillna(data['pedal_edema'].mode()[0],inplace=True)
data['pedal_edema'].fillna(data['pedal_edema'].mode()[0],inplace=True) for i in
catcols: print("LABEL ENCODING OF:",i) LEi = LabelEncoder() # creating an object of
LabelEncoder print(c(data[i])) #getting the classes values before transformation
data[i] = LEi.fit_transform(data[i])# trannsforming our text classes to numerical
values print(c(data[i])) #getting the classes values after transformation
print("*"*100)
```

Model Buildiing:

```
x=data.iloc[:,0:25] y=data.iloc[:,24]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,rando m_state=2)#train test
split print(x_train.shape) print(y_train.shape) print(x_test.shape) print(y_test.shape)
from sklearn.ensemble import RandomForestRegressor rf = RandomForestRegressor() from
sklearn.linear_model import LogisticRegression lgr = LogisticRegression()
lgr.fit(x_train,y_train)
y_pred = lgr.predict(x_test)
```

Testing accuracy:

```
from sklearn.metrics import r2_score
acc= r2_score(y_pred,y_test)
acc
```

Save model:

```
import pickle pickle.dump(lgr,open('lgr.pkl','wb'))
conf_mat = confusion_matrix(y_test,y_pred)
conf_mat
```