[CHRONIC KIDNEY DISEASE PREDICTION](https://www.cdc.gov/kidneydisease/basics.html" \l ":~:text=CKD%20is%20a%20condition%20in,as%20heart%20disease%20and%20stroke.)

1. **INTRODUCTION**

* Chronic kidney disease is a progressive condition that affects >10% of the general population worldwide, amounting to >800 million individuals.
* Chronic kidney disease (CKD) is known as a ‘silent killer’, as most people live with the disease for years before they have any symptoms.
* By the time the disease is detected, for many people it’s already too late. In an advanced stage, CKD can only be treated with regular dialysis or a transplant, making it very expensive to manage.
* However, it can usually be prevented by diagnosis the disease through the major factor that causing CKD and then controlling key risk factors.

**2. PROBLEM STATEMENT**

* Chronic kidney disease (CKD) has emerged as one of the most prominent causes of death and suffering in the 21st century. Due in part to the rise in risk factors, such as blood pressure and diabetes mellitus, the number of patients affected by CKD has also been increasing, affecting an estimated 843.6 million individuals worldwide in 2017.
* The prevalence of CKD worldwide is 10.4% among men and 11.8% among women. AKI, experienced by 13.3 million people each year, may resolve or lead to CKD or kidney failure in the future.

1. **OBJECTIVE**

* Predicting CKD chance there or not by major factor like age, blood pressure, blood glucose, blood urea, sodium level and potassium level in blood that causing CKD.
* Data analysis is the way of predicting future, i.e. whether there is a chance of getting CKD by the factors.
* Data set is converted into numerical values from string data type as Machine learning work with numerical datatype.
* Here the prediction are made Random Forest with Machine Learning using Python.
* At first, multiple regression is used but there accuracy is just 71.2 % so, Random Forest regression is used as there accuracy is 99.17 %
* A random forest produces good predictions and it can handle large datasets efficiently.
* The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm.

1. **IMPLEMENTATION**
   1. RANDOM FOREST REGRESSOR

* Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique.
* It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
* As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
* **The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**
  1. ALGORITHM FOR RANDOM FOREST REGRESSOR

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

**Step-1:** Select random K data points from the training set.

**Step-2:** Build the decision trees associated with the selected data points (Subsets).

**Step-3:** Choose the number N for decision trees that you want to build.

**Step-4:** Repeat Step 1 & 2.

**Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

* 1. CALCULATION

Decision trees learn how to best split the dataset into smaller and smaller subsets to predict the target value. The condition, or test, is represented as the “leaf” (node) and the possible outcomes as “branches” (edges). This splitting process continues until no further gain can be made or a preset rule is met, e.g. the maximum depth of the tree is reached.

## 

## 4.4 Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

## 4.5 Advantages of Random Forest

* Random Forest is capable of performing both Classification and Regression tasks.
* It is capable of handling large datasets with high dimensionality.
* It enhances the accuracy of the model and prevents the overfitting issue.

## 4.6 Disadvantages of Random Forest

* Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

1. **CODING FOR CHRONIC KIDNEY DISEASE PREDICTION**

# Importing dataset

from google.colab import files

upload=files.upload()

# Importing libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import sklearn

# Reading dataset

df=pd.read\_csv('/content/1Project\_kidney.csv')

# Column attributes in dataset

df.columns

# Converting String to number

df=pd.DataFrame(df)

df['rbc']=np.where((df.rbc=='normal'),0,np.where((df.rbc=='abnormal'),1,0))

df=pd.DataFrame(df)

df['pc']=np.where((df.pc=='normal'),0,np.where((df.pc=='abnormal'),1,0))

df=pd.DataFrame(df)

df['pcc']=np.where((df.pcc=='notpresent'),0,np.where((df.pcc=='present'),1,0))

df=pd.DataFrame(df)

df['ba']=np.where((df.ba=='notpresent'),0,np.where((df.ba=='present'),1,0))

df=pd.DataFrame(df)

df['htn']=np.where((df.htn=='yes'),1,np.where((df.htn=='no'),0,0))

df=pd.DataFrame(df)

df['dm']=np.where((df.dm=='yes'),1,np.where((df.dm=='no'),0,0))

df=pd.DataFrame(df)

df['cad']=np.where((df.cad=='yes'),1,np.where((df.cad=='no'),0,0))

df=pd.DataFrame(df)

df['appet']=np.where((df.appet=='good'),0,np.where((df.appet=='poor'),1,0))

df=pd.DataFrame(df)

df['pe']=np.where((df.pe=='yes'),1,np.where((df.pe=='no'),0,0))

df=pd.DataFrame(df)

df['ane']=np.where((df.ane=='yes'),1,np.where((df.ane=='no'),0,0))

df=pd.DataFrame(df)

df['classification']=np.where((df.classification=='ckd'),1,np.where((df.classification=='notckd'),0,0))

# Displaying numerical dataset

df

# Training the dataset for cocerned attributes only

X = df[['Age','BloodPressure','BloodGlucose ','Bloodurea','sodium\_level', 'potassium\_level']]

y = df['classification']

# Training the data using multiple regression

import pandas as pd

from sklearn import linear\_model

regr = linear\_model.LinearRegression()

regr.fit(X, y)

r\_sq = regr.score(X, y)

print(r\_sq)

# Accuracy is just 71.23% so random forest is used for more accuracy

from sklearn.ensemble import RandomForestRegressor

model=RandomForestRegressor()

model.fit(X,y)

r\_sq = model.score(X, y)

print(“Accuracy for Random Forest:”)

print(r\_sq)

# Accuracy is 99.17% so random forest is used

# Creating Graphical User Interface

# installing gradio for GUI

pip install gradio

# To display the min and max value in GUI based on dataset

df = pd.DataFrame(df)

print(df.min(),df.max())

# Function definition for using gradio

def ckd(Age,BloodPressure,BloodGlucose ,Bloodurea,sodium\_level,potassium\_level):

x =np.array([Age,BloodPressure,BloodGlucose,Bloodurea,sodium\_level,potassium\_level])

ypred1=model.predict(x.reshape(1, -1))

if(ypred1<0.5):

return (" No Chronic Kidney Diseases")

else:

return (" Chronic Kidney Diseases")

# importing gradio

import gradio as grd

# Attributes used in gradio

outputs = gr.outputs.Textbox()

app1 = gr.Interface(fn=ckd, inputs=['number','number','number','number','number','number'], outputs=outputs,title="Prediction of Chronic Kidney Disease",description=" Age(0-100)in Years ,BloodPressure(10-310) in mm/Hg ,BloodGlucose(100-125) in mgs/dl ,Sodium(110--150)inmEq/L,Potassium(2.5-50)mEq/L ",examples=[[48,70,117,56,111,2.5],[58,80,131,18,141,3.5]])

app1.launch()

# Decision tree for

from sklearn import tree

clf = tree.DecisionTreeClassifier()

clf = clf.fit(X, y)

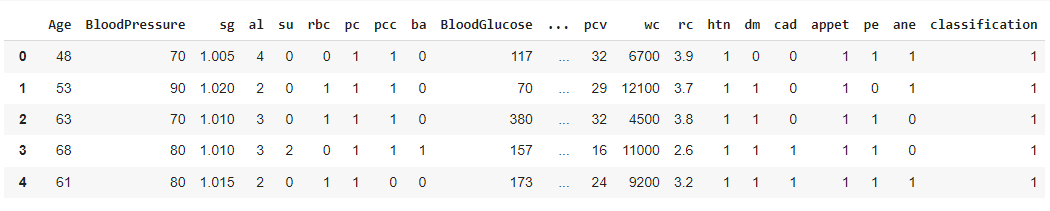
tree.plot\_tree(clf)

import graphviz

dot\_data = tree.export\_graphviz(clf, out\_file=None)

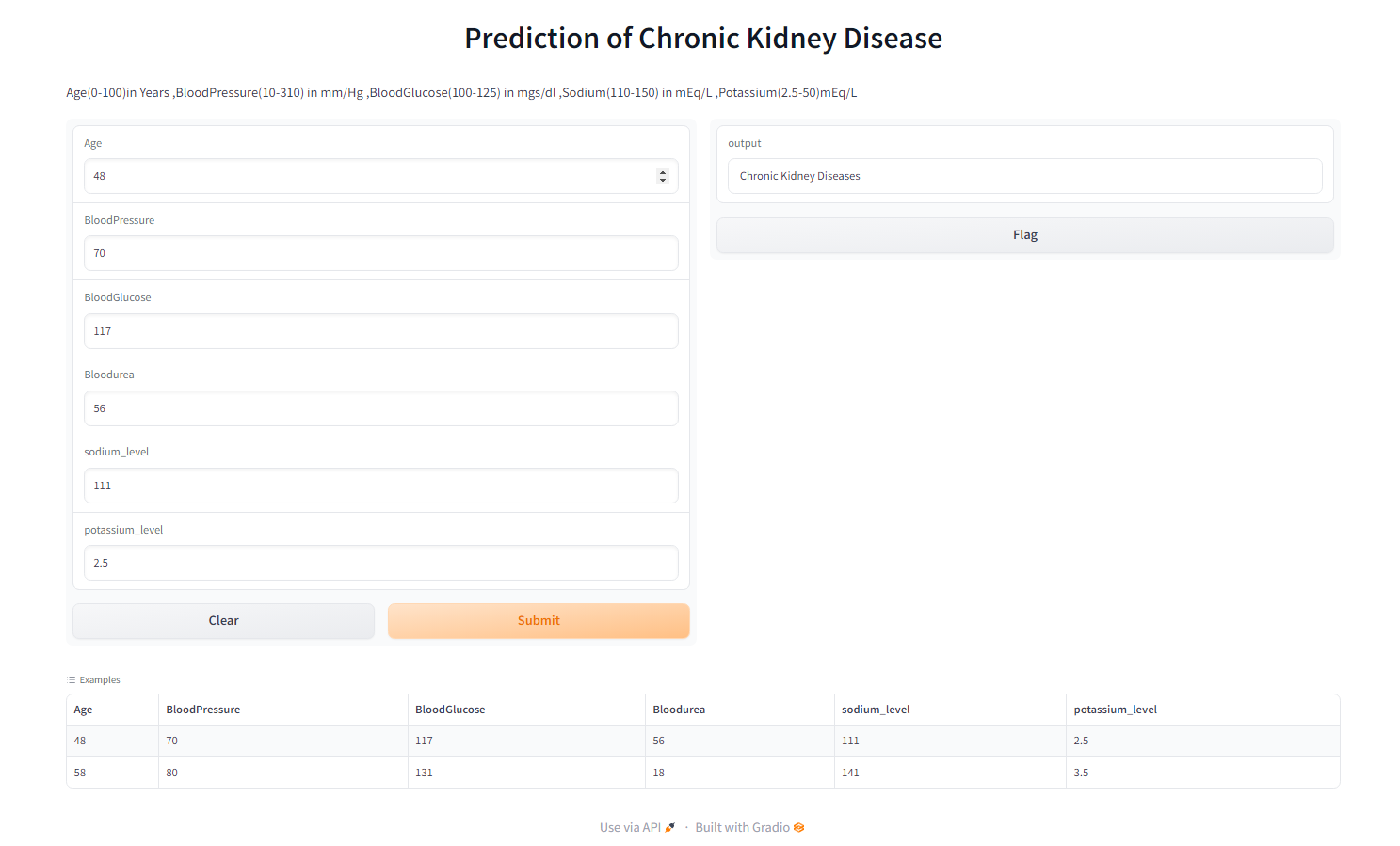
graph = graphviz.Source(dot\_data) graph.render("Chronic Kidney Disease")

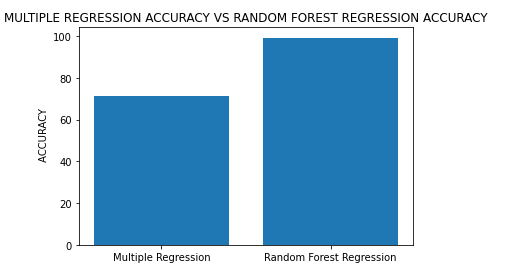
1. **OUTPUT FOR CHRONIC KIDNEY DISEASE PREDICTION**



Accuracy for Multiple Regression: 71.2%

Accuracy for Random Forest: 99.17 %





The final output is presented using the Google colab’s default GUI in which the Chronic Kidney Disease is predicted based on the given input and the accuracy multiple and Random forest regression is visualized.

1. **CONCLUSION**

I hereby conclude that the prediction of the Chronic Kidney Disease is present for the persons or not by the factors of age, blood pressure, blood glucose, blood urea level, sodium level and potassium level.

1. **RESULT**

We have successfully predicted the Chronic Kidney Disease for the person based on the input.

**NAME** : Ashwini S

**DEPT** : B.E Computer Science and Engineering

**YEAR**  : IInd Year

**REG.NO**: 910021104005

**STUDENT SIGNATURE FACULTY’S SIGNATURE**