**GOVERNMENT ARTS AND SCIENCE COLLEGE**

(Affiliated to University of Madras)

**PERUMBAKKAM – 600131**



EARLY PREDICTION OF CHRONIC KIDNEY DISEASE

**PROJECT REPORT**

Submitted For The Partial Fulfillment Of The Requirement For The

Degree Of Bachelor Of Computer Application

**SUBMITTED by**

**Manikandan. S**

**Santhosh. A**

**Arjun. K**

**Praveena. D**

PROJECT TITLE:EARLY PREDICTION OF CHRONIC KIDNEY DISEASE

**INDROCUTION:**

**OVERVIEW**

Chronic kidney disease(CKD) means that your kidney are damaged and can’t filter blood as they should. This damaged can cause wastes to build up in your body. It can also cause other problems that can harm your health. Diabetes and high blood pressure are the most common causes of CDK.

Chronic Kidney Disease (CKD) is a major medical problem and can be cured if treated in the early

stages. Usually, people are not aware that medical tests we take for different purposes could contain

valuable information concerning kidney diseases. Consequently, attributes of various medical tests

are investigated to distinguish which attributes may contain helpful information about the disease. The

information says that it helps us to measure the severity of the problem, the predicted survival of the

patient after the illness, the pattern of the disease and work for curing the disease.

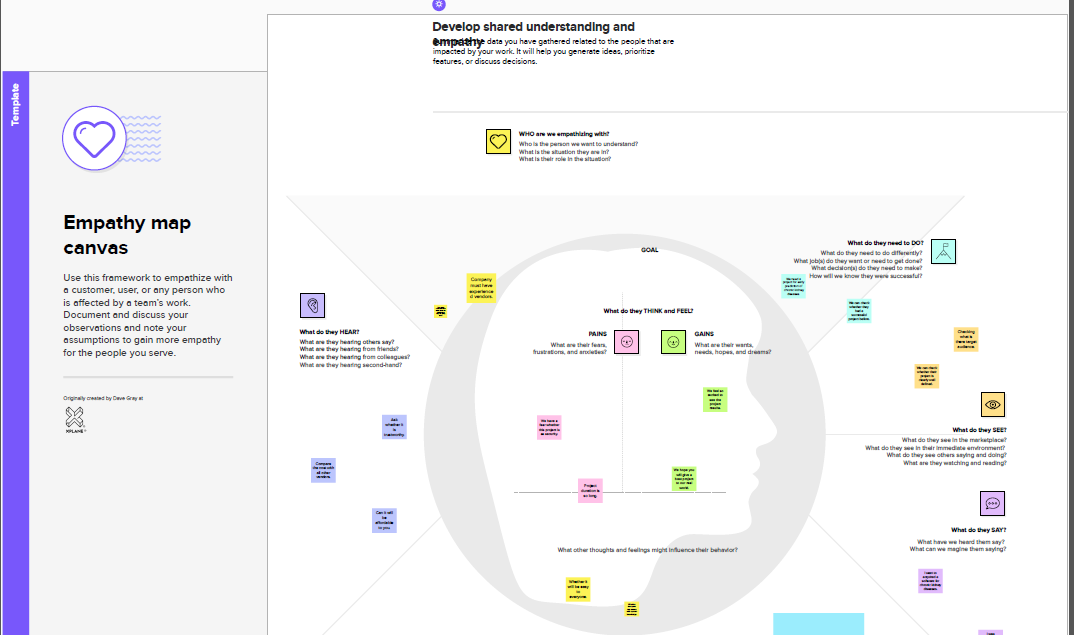
**PURPOSE:**

* Prevent and control risk factors for CKD.
* Raise awareness of CKD and its complications.
* Promote early diagnosis and treatment of CKD.
* The rationale for testing asymptomatic people for CKD is that earlier detection might allow for the implementation of therapeutic interventions and avoidance of inappropriate exposure to nephrotoxic agents, both of which may slow the progression of CKD to end-stage kidney disease.
* There are many people who are suffering from chronic kidney diseases worldwide. Due to the several risk factors like food, environment and living standards many people get diseases suddenly. Diagnosing of chronic kidney diseases is generally invasive, costly, time-consuming and often risky. That is why many patients reach late stages of it without treatment, especially in those countries where the resources are limited. Therefore, the early detection strategy of the disease remains important, particularly in developing countries, where the diseases are generally diagnosed in later stages. Finding a solution for the above-mentioned problems and riding out from disadvantages became a strong motive to conduct this study. Chronic Kidney Disease (CKD) is one of the types of kidney disease which results in gradual loss of kidney function.

**PROBLEM DEFINITION &DESIGN THINKING:**

**Empathy map:**

**Under this activity our team members have gathered and discussed various ideas to solve our project problem. Each member contributed 6 to 10 ideas. After gathering all ideas we have assessed the impact feasibility of each point. Finally we have assigned the priority for each point based on the impact value.**

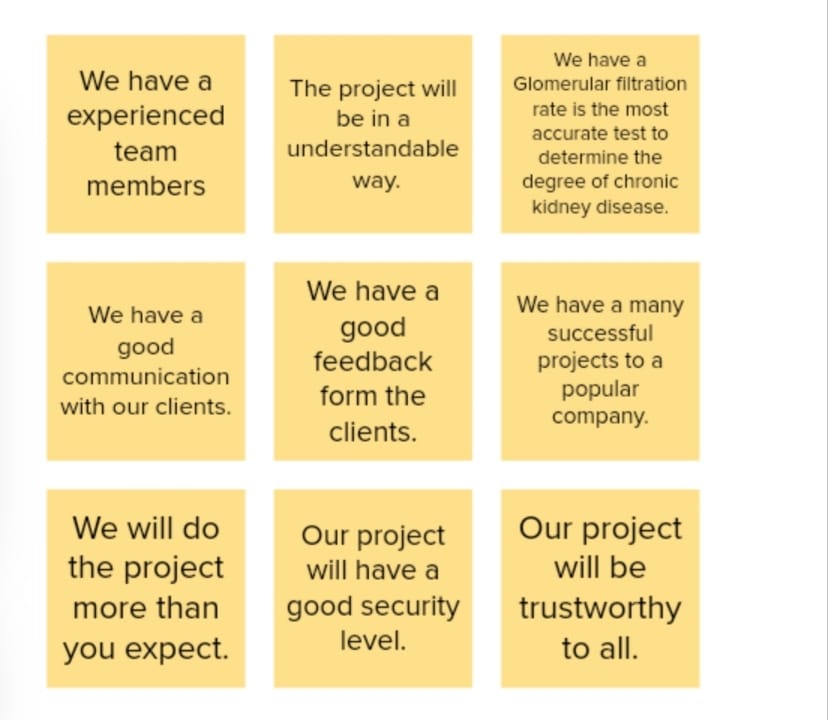


**IDEATION&BRAINSTROMING MAP:**

Under this activity our team members have gathered and discussed various ideas to solve our project problem.

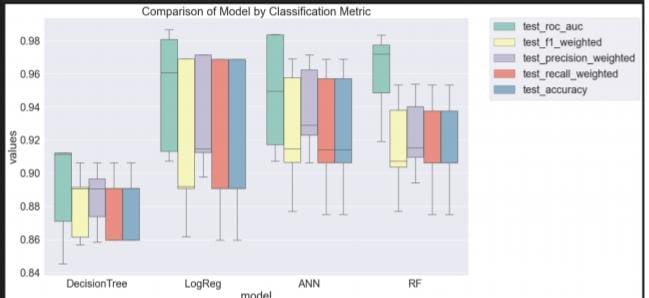
Each member contributed 6 to 10 ideas after gathering all ideas we have assessed the impact and feasibility of each points.

Finally we have assigned the priority for each point based on the impact value.





**RESULT:**

****

**ADVANTAGES & DISADVANTAGES:**

**ADVANTAGES:**

* + **Early treatment: Early detection of CKD allows for early treatment, which can help slow the progression of the disease and prevent complications. This can include lifestyle changes such as adopting a healthier diet and increasing physical activity, as well as medication and other medical interventions.**
* **Preventing complications: Early treatment can also help prevent complications of CKD, such as anemia, bone disease, high blood pressure, and cardiovascular disease.**
* **Cost-effective: Early prediction and treatment of CKD can be cost-effective, as it can prevent or delay the need for more expensive and invasive treatments such as dialysis or kidney transplantation.**
* **Better quality of life: Early intervention can also improve the quality of life for people with CKD, allowing them to maintain their independence and continue their usual activities for longer periods of time.**
* **Improved patient outcomes: Early prediction and treatment of CKD can lead to improved patient outcomes, including a lower risk of hospitalization, longer survival, and better overall health.**

**DISADVANTAGES:**

* **CKD is defined as a progressive loss of kidney function over time, and early detection may identify individuals with mild kidney impairment that may never progress to severe disease or end-stage renal failure.**
* **However, individuals with early-stage CKD may be subjected to unnecessary medical interventions such as medication therapy, dietary restrictions, or even dialysis, which can have adverse effects on quality of life and may not improve long-term outcomes.**
* **Furthermore, early identification of CKD may lead to increased anxiety, psychological distress, and stigma associated with chronic disease, which may negatively impact mental health and well-being.**
* **Therefore, while early detection of CKD can be beneficial in some cases, it is important to balance the potential benefits with the potential harms of overdiagnosis and overtreatment.**

**APPLICATIONS:**

* **Identification of individuals at high risk of developing CKD: Early prediction models can help identify individuals who are at risk of developing CKD, such as those with a family history of kidney disease, diabetes, hypertension, or obesity. These individuals can then be targeted for early interventions to prevent or delay the onset of CKD.**
* **Monitoring disease progression: Early prediction models can help monitor the progression of CKD over time, allowing for early intervention and treatment to slow or stop the progression of the disease.**
* **Personalized treatment plans: Early prediction models can help healthcare providers develop personalized treatment plans for individuals with CKD based on their predicted risk of progression and other individual characteristics.**
* **Patient education: Early prediction models can help educate patients about their risk of developing CKD and the importance of early detection and management of the disease.**
* **Research: Early prediction models can be used in research to identify new biomarkers or risk factors for CKD, as well as to evaluate the effectiveness of new interventions for preventing or treating the disease.**

**CONCLUSION:**

**In conclusion, early prediction of chronic kidney disease (CKD) can provide important clinical and public health benefits, such as identifying individuals at risk of developing CKD, monitoring disease progression, developing personalized treatment plans, educating patients, and advancing research. However, it is important to consider the potential disadvantages of overdiagnosis and overtreatment, as well as the psychological and emotional impact of early diagnosis of a chronic disease. Therefore, healthcare providers should use early prediction models judiciously, weighing the benefits and risks, and engaging patients in shared decision-making to ensure that they receive the most appropriate care based on their individual circumstances. By identifying and treating CKD early, we can improve patient outcomes, reduce healthcare costs, and enhance the overall quality of care for individuals with kidney disease.**

**FUTURE SCOPE:**

**The future scope of early prediction of chronic kidney disease (CKD) is promising, and several areas of development are currently being explored. Some of these include:**

* + **Use of artificial intelligence (AI): AI algorithms are being developed to analyze large volumes of data from electronic health records, laboratory tests, and imaging studies to predict CKD risk more accurately and efficiently.**
  + **Integration of multi-omics data: Integration of data from genomics, transcriptomics, proteomics, and metabolomics may provide a more comprehensive understanding of the molecular mechanisms underlying CKD development and progression.**
  + **Personalized risk prediction: Incorporating individual-level factors such as genetics, lifestyle, and environmental exposures into risk prediction models may improve the accuracy and precision of CKD risk assessment and enable more personalized prevention and treatment strategies.**
  + **Mobile health (mHealth) applications: mHealth applications, such as smartphone apps and wearable devices, can be used to monitor CKD risk factors, provide real-time feedback, and facilitate communication between patients and healthcare providers.**
  + **Telehealth and remote monitoring: Telehealth and remote monitoring technologies can enable more frequent and convenient monitoring of CKD patients, reducing the burden of in-person visits and enabling earlier detection and management of disease complications.**
  + **Overall, these and other advancements in technology and healthcare delivery have the potential to significantly improve the early prediction and management of CKD, ultimately improving patient outcomes and reducing the burden of kidney disease on individuals and society.**

**APPENDIX:**

**SOURCE CODE:**

**COLLECT THE DATA SET:**

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

import warnings

Read the dataset:

data=pd.read\_csv("/content/kidney\_disease.csv")

data.head()

2.**DATA PREPARATION:**

**Rename the 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\_creatinite','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

data.info()

data.isnull().any()

Handling Categorical Columns:

catcols=set(data.dtypes[data.dtypes=='O'].index.values)

print(catcols)

for i in catcols:

    print("Columns:",i)

    print(c(data[i]))

    print('\*'\*120+'\n')

catcols.remove('packed\_cell\_volume')

catcols.remove('red\_blood\_cell\_count')

catcols.remove('white\_blood\_cell\_count')

print(catcols)

**Label Encoding for categorical columns:**

Catcols

from sklearn.preprocessing import LabelEncoder

for i in catcols:

    print("LABEL ENCODING OF:",i)

    LEi = LabelEncoder()

    print(c(data[i]))

    data[i] = LEi.fit\_transform(data[i])

    print(c(data[i]))

    print("\*"\*100)

**Handling Numerical columns:**

contcols=set(data.dtypes[data.dtypes!='O'].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')

contcols.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['coronary\_artery\_disease'])

data['diabetesmellitus'] = data.diabetesmellitus.replace(to\_replace={'\tno':'no','\tyes':'yes'})

c(data['diabetesmellitus'])

**3.EXPLORATORY DATA ANALYSIS:**

Discriptive statistical Analysis:

data.describe()

Visual analysis:

Univariate analysis:

sns.distplot(data.age)

Bivariate analysis:

import matplotlib.pyplot as plt

fig=plt.figure(figsize=(5,5))

plt.scatter(data['age'],data['blood\_pressure'],color='blue')

plt.xlabel('age')

plt.ylabel('blood pressure')

plt.title("age VS blood Scatter Plot ")

Multivariate analysis:

plt.figure(figsize=(20,15), facecolor='white')

plotnumber = 1

for column in columnList:

    if plotnumber<=11 :

       ax = plt.subplot(3,4,plotnumber)

       plt.scatter(data['age'],data[column])

       plt.xlabel(column,fontsize=20)

    plotnumber+=1

plt.show()

Finding correlation between the independent Columns:

f,ax=plt.subplots(figsize=(18,19))

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(x=data['class'])

**Scaling the data:**

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

xscaled=sc.fit\_transform(x)

**Creating Independent and Dependent:**

setcols=['red\_blood\_cells','pus\_cell','blood glucose random','blood\_urea','pedal\_edema','anemia','diabetesmellitus','coronary\_artery\_disease']

x=pd.DataFrame(data,columns=setcols)

y=pd.DataFrame(data,columns=['class'])

print(x.shape)

print(y.shape)

**Splitting 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)

**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',loss='binary\_crossentropy',metrics=['accuracy'])

classification.fit(x\_train,y\_train,batch\_size=10,validation\_split=0.2,epochs=100)

Random Forest Model:

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators = 100,criterion='entropy')

rfc.fit(x\_train,y\_train)

y\_predict\_ = rfc.predict(x\_train)

y\_predict\_train = rfc.predict(x\_train)

**Decision tree model:**

from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(max\_depth=4, splitter='best',criterion='entropy')

dtc.fit(x\_train,y\_train)

y\_predict= dtc.predict(x\_test)

y\_predict

y\_predict\_train = dtc.predict(x\_train)

**Logistic Regresion:**

from sklearn.linear\_model import LogisticRegression

lgr = LogisticRegression()

lgr.fit(x\_train,y\_train)

from sklearn.metrics import accuracy\_score,classification\_report

y\_predict = lgr.predict(x\_test)

**Testing the model:**

y\_pred = lgr.predict([[2,1,93.000000,66.0,0,0,4,1]])

print(y\_pred)

(y\_pred)

y\_pred = dtc.predict([[2,1,93.000000,66.0,0,0,4,1]])

print(y\_pred)

(y\_pred)

y\_pred = rfc.predict([[2,1,93.000000,66.0,0,0,4,1]])

print(y\_pred)

(y\_pred)

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):

sample\_value = np.array(sample\_value)

sample\_value = sample\_value.reshape(1,-1)

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: Hign chance of CKD!')

else:

    print('Prediction: Low chance of CKD')

**5.PERFORMANCE TESTING&EVALUATE THE RESULTS**

from sklearn import model\_selection

dfs = []

models = [('LogReg', LogisticRegression()),

          ('RF', RandomForestClassifier()),

          ('DecisionTree',DecisionTreeClassifier())]

results = []

names = []

scoring = ['accuracy', 'precision\_weighted', 'recall\_weighted', 'f1\_weighted', 'roc\_auc']

target\_names = ['NO CKD', 'CKD']

for name, model in models:

    kfold = model\_selection.KFold(n\_splits=5, shuffle=True,random\_state=90210)

    cv\_results = model\_selection.cross\_validate(model, x\_train,y\_train, cv=kfold,scoring=scoring)

    clf = model.fit(x\_train,y\_train)

    y\_pred = clf.predict(x\_test)

    print(name)

    print(classification\_report(y\_test,y\_pred))

    results.append(cv\_results)

    names.append(name)

    this\_df = pd.DataFrame(cv\_results)

    this\_df['model'] = name

    dfs.append(this\_df)

    final = pd.concat(dfs, ignore\_index=True)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test,y\_predict)

cm

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

sns.heatmap(cm, cmap='Blues', annot=True,xticklabels=['no ckd','ckd'], yticklabels=['no ckd','ckd'])

plt.xlabel('Predicted Values')

plt.ylabel('Actual values')

plt.title('Confusion Matrix for Logistic Regression model')

plt.show()

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test,y\_predict)

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

sns.heatmap(cm, cmap='Blues', annot=True,xticklabels=['no ckd','ckd'], yticklabels=['no ckd','ckd'])

plt.xlabel('Predicted Values')

plt.ylabel('Actual values')

plt.title('Confusion Matrix for Logistic Regression model')

plt.show()

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_predict)

cm

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

sns.heatmap(cm, cmap='Blues',annot=True, xticklabels=['nockd', 'ckd'], yticklabels=['no ckd', 'ckd'])

plt.xlabel('Predicted values')

plt.ylabel('Actual values')

plt.title('Confusion Matrix for RandomForestClassifier')

plt.show()

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_predict)

cm

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

sns.heatmap(cm, cmap='Blues', annot=True, xticklabels=['no ckd', 'ckd'], yticklabels=['no ckd', 'ckd'])

plt.xlabel('Predicted values')

plt.ylabel('Actual values')

plt.title('Confusion Matrix for DecisionTreeClassifier')

plt.show()

bootstraps = []

for model in list(set(final.model.values)):

    model\_df = final.loc[final.model == model]

    bootstrap = model\_df.sample(n=30, replace=True)

    bootstraps.append(bootstrap)

bootstrap\_df = pd.concat(bootstraps, ignore\_index=True)

results\_long = pd.melt(bootstrap\_df,id\_vars=['model'],var\_name='metrics', value\_name='values')

time\_metrics=['fit\_time','score\_time']

results\_long\_nofit = results\_long.loc[~results\_long['metrics'].isin(time\_metrics)]

results\_long\_nofit = results\_long\_nofit.sort\_values(by='values')

results\_long\_fit = results\_long.loc[results\_long['metrics'].isin(time\_metrics)]

results\_long\_fit = results\_long\_fit.sort\_values(by='values')

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(20,12))

sns.set(font\_scale=2.5)

g = 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=0)

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

plt.savefig('./benchmark\_models\_performance.png',dpi=300)

**6.MODEL DEPLOYMENT:**

**Save the best model:**

pickle.dump(lgr,open('CKD.pk1','wb'))