# GOVERNMENT ARTS COLLEGE

## CHIDAMBARAM - 608 102

***Re-Accredited by NAAC with 'B' Grade***

***(Affiliated to Thiruvalluvar University, Vellore)***

**Team ID : NM2023TMID23525**

**Team Leader : DHARANYA D**

**Team Member : DEEPA V**

**Team Member : DHANUSHIYA M**

**Team Member : GNANAPRAKASH M**

# Project Report

## Introduction:

### 1.1 Overview:

1. Chronic kidney disease (CKD) is a condition in which the kidneys gradually lose function over time. The kidneys are responsible for filtering waste and excess fluids from the blood, and when they don't work properly, these waste products can build up in the body, causing various health problems.

2.There are five stages of CKD, with stage 1 being the mildest and stage 5 being the most severe. The stages are determined by the level of kidney function, as measured by the glomerular filtration rate (GFR).

3.Common causes of CKD include high blood pressure, diabetes, and certain kidney diseases. Other risk factors include age, obesity, smoking, and a family history of kidney disease.

### 1.2 Purpose:

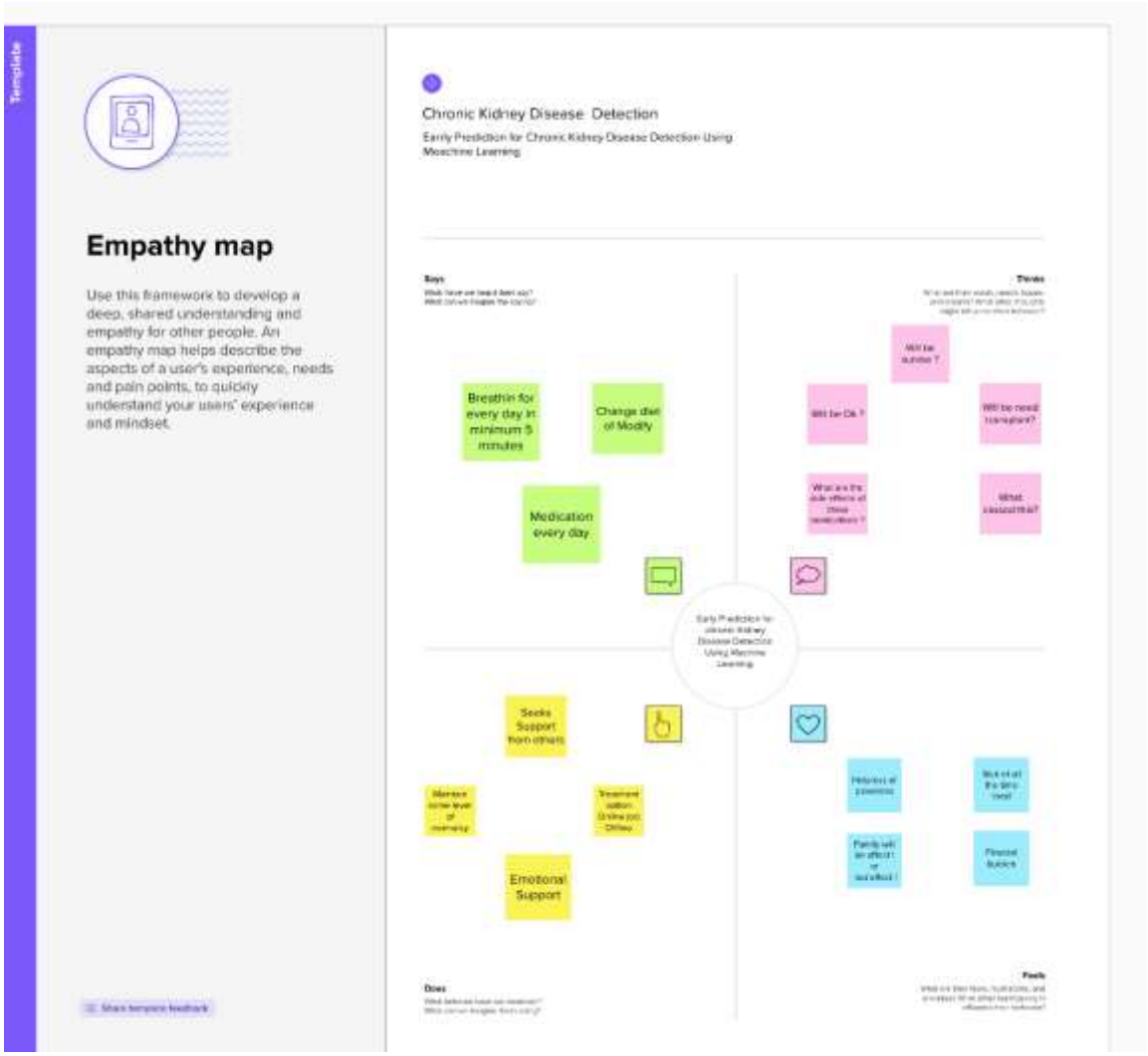
1.Chronic kidney disease (CKD) is not a purposeful condition, but rather a disease that occurs when the kidneys lose function over time. The purpose of the kidneys is to filter waste and excess fluids from the blood, maintain the body's fluid and electrolyte balance, and produce hormones that help regulate blood pressure and red blood cell production. When CKD occurs, the kidneys are unable to perform these functions properly, which can lead to a variety of health problems.

The purpose of managing CKD is to slow the progression of the disease, prevent complications, and improve quality of life. This may involve treating the underlying cause of the disease, such as high blood pressure or diabetes, and making lifestyle changes to promote kidney

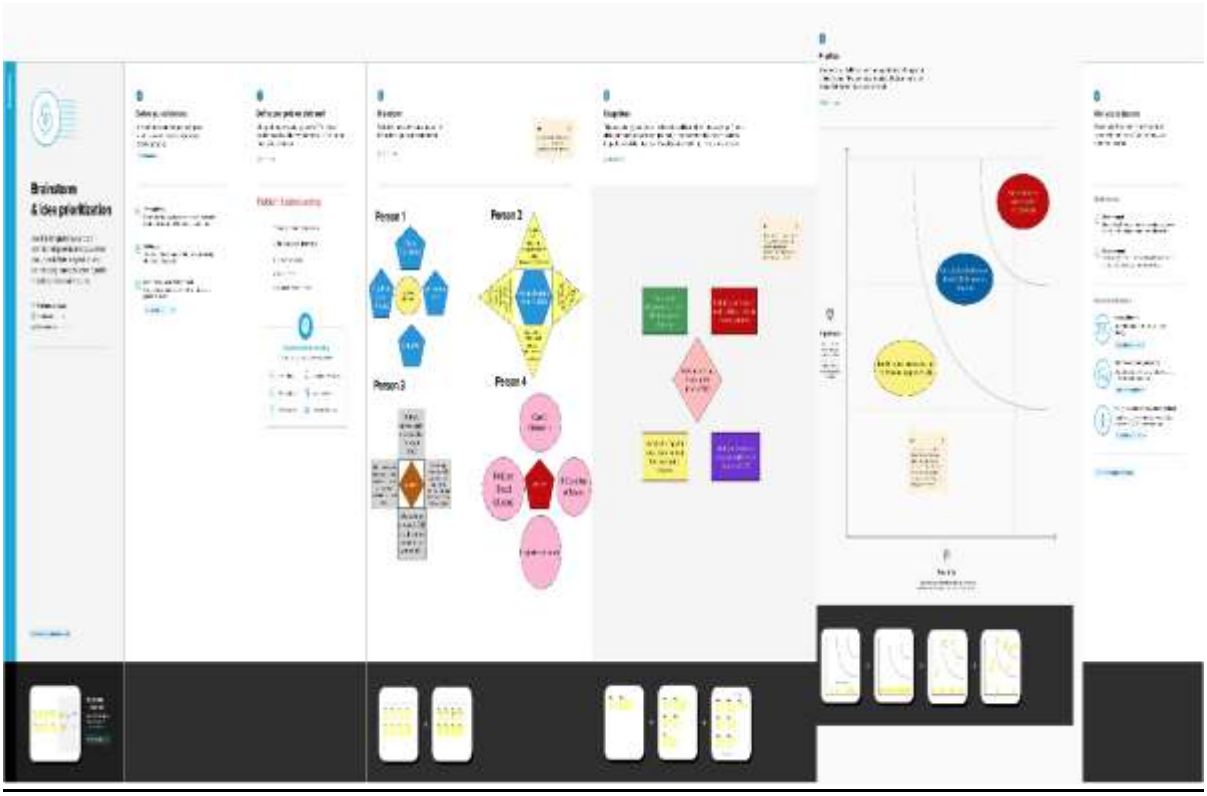
health. Medications may be 2.prescribed to control blood pressure and blood sugar levels, and to manage complications such as anemia and bone disease. In some cases, dialysis or kidney transplant may be necessary to replace lost kidney function. Regular monitoring and management of CKD can help prevent or delay the need for these interventions and improve long-term outcomes

# 2 Problem Definition & Design Thinking

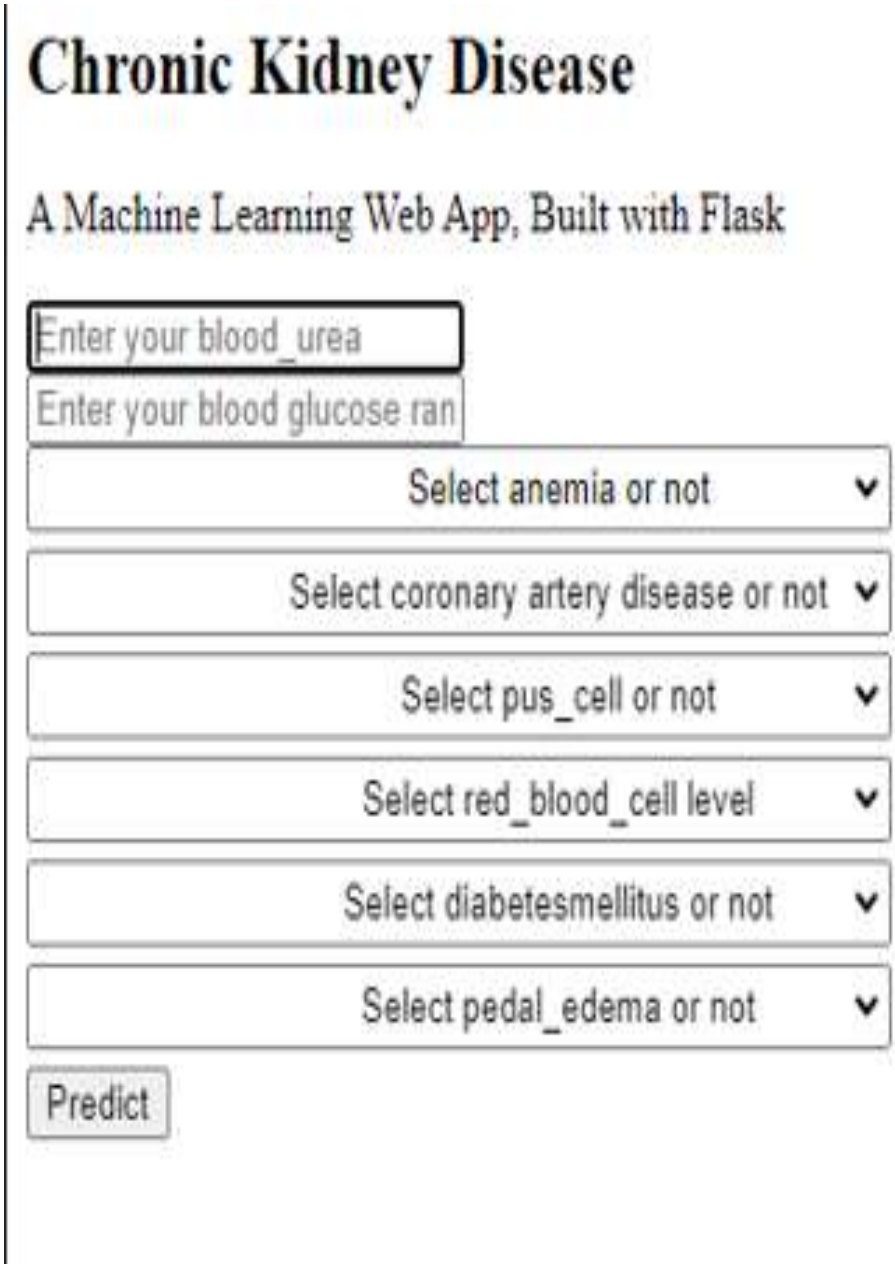
## 2.1 Empathy Map



# Ideation & Brainstorming Map



## 3 RESULT:


Chronic Kidney Disease
A Machine Learning Web App, Built with Flask
Enter your blood_urea
Enter your blood glucose ran
Select anemia or not
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
Predict

# Chronic Kidney Disease

A Machine Learning Web App, Built with Flask

36

121

NO

NO

normal

Select red_blood_cell level

YES

NO

Predict

# Chronic Kidney Disease

A Machine Learning Web App, Built with Flask

# Prediction: Oops! You have Chronic Kidney Disease.

Diabetes Image

# 4 ADVANTAGES & DISADVANTAGES

## Advantages :

1.There are no advantages to having chronic kidney disease (CKD). CKD is a serious medical condition that can lead to a range of complications and health problems. If left untreated or poorly managed, CKD can progress to end-stage kidney disease, which requires dialysis or kidney transplant for survival.

2.However, there are potential advantages to managing CKD effectively. By taking steps to slow the progression of the disease and prevent complications, individuals with CKD can improve their quality of life and reduce their risk of serious health problems. For example, controlling blood pressure and blood sugar levels can help prevent heart disease and stroke, which are common complications of CKD. Managing anemia can improve energy levels and reduce fatigue, while treating bone disease can reduce the risk of fractures and bone pain.

3.In addition, early detection and management of CKD can help delay or prevent the need for dialysis or kidney transplant. This can improve outcomes and reduce the burden of treatment for individuals with CKD.

4.Overall, while there are no advantages to having CKD itself, effective management can improve quality of life and reduce the risk of serious complications.

## Disadvantages:

1. Decreased kidney function: CKD is characterized by a gradual loss of kidney function over time. This can lead to a buildup of waste products and fluids in the body, which can cause a range of health problems.
2. Cardiovascular disease: CKD is a major risk factor for cardiovascular disease, which can include heart attack, stroke, and heart failure. Individuals with CKD are more likely to develop these conditions than the general population.
3. Anemia: CKD can lead to a decrease in red blood cells, which can cause anemia. Anemia can lead to fatigue, weakness, and shortness of breath.

4. Bone disease: CKD can cause bone disease, which can lead to bone pain, fractures, and deformities. This is because the kidneys play a role in regulating calcium and other minerals in the body.

5. Dialysis: In severe cases of CKD, dialysis may be necessary to replace lost kidney function. Dialysis is a time-consuming and often uncomfortable treatment that can have a range of side effects.

6. Reduced quality of life: The symptoms and complications of CKD can significantly impact quality of life, leading to fatigue, depression, anxiety, and other psychological and emotional challenges.

## 5. APPLICATION:

1. Early Detection: Early detection is crucial in managing CKD. Solutions can be developed to improve early detection and diagnosis of CKD through routine screening, including blood tests, urine tests, and blood pressure monitoring.

2. Treatment: Treatment of CKD typically involves managing underlying health conditions, such as high blood pressure and diabetes, which can contribute to kidney damage. Solutions can be developed to improve the effectiveness of treatments and medications for CKD.

3. Dialysis: Dialysis is a treatment option for patients with kidney failure. Solutions can be developed to improve the efficiency and effectiveness of dialysis, as well as to reduce the risk of complications associated with dialysis.

4. Transplantation: Kidney transplantation is another treatment option for patients with kidney failure. Solutions can be developed to increase the availability of donor kidneys and to improve the success rate of kidney transplant surgeries.

5. Lifestyle Changes: Lifestyle changes can help slow the progression of CKD and improve overall health. Solutions can be developed to promote healthy lifestyle changes, such as regular exercise and a healthy diet, and to provide education and support for patients with CKD.

## 6.CONCLUSION

Chronic kidney disease (CKD) is a serious medical condition that affects millions of people worldwide. It is a progressive disease that can lead to kidney failure if left untreated. CKD is often caused by underlying health conditions such as diabetes and hypertension, but it can also result from genetic factors or exposure to certain toxins.

Early detection and treatment of CKD are crucial for preventing the disease from progressing to kidney failure. Treatment options include medication, lifestyle changes such as diet and exercise, and in severe cases, dialysis or kidney transplantation.

Research on CKD has led to a better understanding of the disease and improved treatment options. However, there is still much to learn about the underlying causes and mechanisms of CKD, as well as how to prevent and manage it more effectively.

Overall, CKD is a complex and challenging condition that requires ongoing attention and care from healthcare providers, patients, and their families. With proper management and support, many people with CKD are able to maintain a good quality of life and avoid the worst complications of the disease.

## 7 .FUTURE SCOP

1. Early Detection: One of the biggest challenges in managing CKD is early detection. Efforts should be made to increase awareness of CKD risk factors and promote regular screening tests for people at high risk, such as those with diabetes or high blood pressure.

2. Personalized Medicine: Personalized medicine holds great promise for CKD management. Advances in genetic testing and molecular profiling may enable healthcare providers to tailor treatments based on a patient's individual genetic and molecular characteristics.

3. Novel Therapies: There is a need for new therapies that can slow or halt the progression of CKD. Researchers are exploring new drug targets and developing innovative treatments, such as regenerative medicine approaches, to improve kidney function.

4. Patient Engagement: Patient engagement is critical to CKD management. Patients need to be educated about the disease and encouraged to take an active role in their care. Tools such as telemedicine and mobile health apps can help patients track their symptoms and communicate with their healthcare providers.

5. Health Equity: Health disparities exist in the incidence and outcomes of CKD. Efforts should be made to address social determinants of health, such as poverty and lack of access to healthcare, that contribute to these disparities.

# 8 APPENDIX

## Source Code:

```
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

data=pd.read_csv("/content/kidney_disease.csv")

data.head()

data.columns

data.columns=['id','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

data.info()

data.isnull().any()

data.isnull().sum()

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

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

data['sodium'].fillna(data['sodium'].mean(), 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['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[ 'specific_gravity'].fillna(data[ 'specific_gravity'].mode()[0], inplace=True)

catcols=set(data.dtypes[data.dtypes!='0'].index.values)

print(catcols)

for i in catcols:

    print("Columns :",i)
print(c(data[i])) #using counter for checking the number of classess in the colu
mn
```

```
    print('*'*120+'\n')

catcols.remove('red_blood_cell_count')

catcols.remove('packed_cell_volume')

catcols.remove('white_blood_cell_count')

print(catcols)

catcols=['anemia','pedal_edama','appetite','bacteria',
         'class','coronary_artery_disease','diabetesmellit','hypertension',
         'pus_cell','pus_cell_clumps','red_blood_cells']

from sklearn.preprocessing import LabelEncoder

for i in catcols:

  print("LABEL ENCODING OF:",i)

  LEi=LabelEncoder()

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

print(c(data[i]))

print('*'*100)

contcols=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')

contcols.remove('sugar')

print(contcols)

contcols.add('red_blood_cell_count')

contcols.add('packed_cell_volume')

contcols.add('white_blood_cell_count')

print(contcols)

print(catcols)

data['coronary_artery_disease']=data.coronary_artery_disease.replace('\tno', 'no'
)

c(data['coronary_artery_disease'])

data['diabetesmellitus']=data.diabetesmellitus.replace('\tno', 'no')

c(data['diabetesmellitus'])

data.describe()

sns.distplot(data.age)


import matplotlib.pyplot as plt #import the matplotlib liharay

fig=plt.figure(figsize=(5,5)) #plot size

plt.scatter(data['age'],data['blood pressure'],color='blue')
plt.xlabel('age') #set the label for x-axis
```

```
plt.ylabel('blood pressure') #set the label for y axis

plt.title("age vs blood scatter Plot") #set a title for the axes

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

plotnumber = 1

for column in contcols:

    if plotnumber<=11:

        if data[column].dtype == 'float64' or data[column].dtype == 'int64':

            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="o
range")

plt.xticks(rotation=45)

plt.yticks(rotation=45)

plt.show()

import matplotlib.pyplot as plt

# Create a sample DataFrame with a 'class' column

data = pd.DataFrame({'class': ['A', 'A', 'B', 'C', 'C', 'C']})
```

```
# Use Matplotlib to create a countplot
fig, ax = plt.subplots()
ax.hist(data['class'])
ax.set_xlabel('Class')
ax.set_ylabel('Count')
plt.show()
import numpy as np
from sklearn.preprocessing import StandardScaler
# Create a sample 2D array of input data
x = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])
# Create a StandardScaler object and call its fit_transform() method
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
# Print the standardized input data
print(x_scaled)
selcols=['red_blood_cells', 'pus_cell', 'blood glucose random', 'blood_urea',
        '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)

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)

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'])

import numpy as np

import tensorflow as tf

from tensorflow import keras

# Create a sample dataset
```

```
x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

y_train = np.array([0, 1, 1, 0])

# Create a simple Keras model

model = keras.Sequential([

    keras.layers.Dense(2, input_shape=(2,), activation='sigmoid'),

    keras.layers.Dense(1, activation='sigmoid')

])

# Compile the model with binary crossentropy loss and Adam optimizer

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

# Train the model for 100 epochs with batch size of 2 and 20% validation split

model.fit(x_train, y_train, batch_size=2, validation_split=0.2, epochs=100)

from sklearn.ensemble import RandomForestClassifier

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score

import numpy as np

# Create a sample dataset

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

y_train = np.array([0, 1, 1, 0])
x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
```

```
y_test = np.array([0, 1, 1, 0])

# Create a RandomForestClassifier object

rfc = RandomForestClassifier(n_estimators=100)


# Train the model using the training data

rfc.fit(x_train, y_train)

# Test the model using the testing data

y_pred = rfc.predict(x_test)

# Calculate the accuracy of the model on the testing data

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)

y_predict= rfc.predict(x_test)

y_predict_train = rfc.predict(x_train)

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

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)

from sklearn.linear_model import LogisticRegression

import numpy as np

# Create a sample dataset

X = np.array([[0, 0, 121.000000, 36.0, 0, 0, 1, 0], [1, 1, 121.000000, 36.0, 0, 0,
1, 0]])

y = np.array([0, 1])

# Create a LogisticRegression object

lgr = LogisticRegression()

# Train the model using the dataset

lgr.fit(X, y)

# Predict the outcome for a new set of input features

X_new = np.array([[1, 1, 121.000000, 36.0, 0, 0, 1, 0]])

y_pred = lgr.predict(X_new)

print(y_pred)

from sklearn.tree import DecisionTreeClassifier

import numpy as np

# Create a sample dataset

X = np.array([[0, 0, 121.000000, 36.0, 0, 0, 1, 0], [1, 1, 121.000000, 36.0, 0, 0,
1, 0]])
```

```
y = np.array([0, 1])

# Create a DecisionTreeClassifier object

dtc = DecisionTreeClassifier()

# Train the model using the dataset

dtc.fit(X, y)

# Predict the outcome for a new set of input features

X_new = np.array([[1, 1, 121.000000, 36.0, 0, 0, 1, 0]])

y_pred = dtc.predict(X_new)

print(y_pred)

from sklearn.ensemble import RandomForestClassifier

import numpy as np

# Create a sample dataset

X = np.array([[0, 0, 121.000000, 36.0, 0, 0, 1, 0], [1, 1, 121.000000, 36.0, 0, 0,
1, 0]])

y = np.array([0, 1])

# Create a RandomForestClassifier object

rfc = RandomForestClassifier()

# Train the model using the dataset

rfc.fit(X, y)

# Predict the outcome for a new set of input features

X_new = np.array([[1, 1, 121.000000, 36.0, 0, 0, 1, 0]])
```

```
y_pred = rfc.predict(X_new)

print(y_pred)

import tensorflow as tf


# Load the saved model

model = tf.keras.models.load_model("ckd.h5")

# Call predict on the model to create the weights

_ = model.predict(x_test)

# Save the model

model.save("ckd.h5")

import numpy as np

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

# Create a sample dataset

X_train = np.array([[0, 0, 121.000000, 36.0, 0, 0, 1, 0], [1, 1, 121.000000, 36.0,
 0, 0, 1, 0]])

y_train = np.array([0, 1])

X_test = np.array([[1, 1, 121.000000, 36.0, 0, 0, 1, 0]])


# Create a machine learning classifier object
```

```
# Example 1: Logistic Regression
classification = LogisticRegression()
# Example 2: Decision Tree
# classification = DecisionTreeClassifier()
# Example 3: Random Forest
# classification = RandomForestClassifier()
# Train the model using the training dataset
classification.fit(X_train, y_train)
# Predict the outcome for the test dataset
y_pred = classification.predict(X_test)
print(y_pred)
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: High chance of CKD!')
else:
    print('Prediction: Low chance of CKD!')
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 = 90
210)
  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, target_names=target_names))
    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)
return final
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 = 90
210)

  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, target_names=target_names))

  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)

return final

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 classification_report

print(classification_report(y_test, y_pred))

bootstraps = []

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

  model_df = final.loc[final.modelmodel]

  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'] # fit time metrics

#A PERFORMANCE METRICS

results_long_nofit=results_long.loc[~results_long['metrics'].isin(timo_metrics)]

# get of without fit data

results_long_nofit=results_long_nofit.sort_values(by='values')

#TIME METRICS

results_long_fit = results_long.loc[results_long['netrics'].isin(tine_metrics)]
```

```
# df with fit data

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)

pickle.dump(lgr, open('CKD.pkl', 'wb'))
```

**App.py**

```
import numpy as np

import pandas as pd

import sklearn

from flask import Flask, request, render_template

import pickle


app=Flask(__name__)

model=pickle.load(open('CKD.pkl','rb'))
```

```
@app.route('/')
def home():
    return render_template('home.html')
@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'])
def predict():
    input_features=[float(x) for x in request.form.values()]
    features_value=[np.array(input_features)]



features_name=['blood_urea','blood_glucose_random','coronary_artery_disease','
anemia','pus_cell','red_blood_cells','diabetesmellitus','pedal_edema']
    df=pd.DataFrame(features_value, columns=features_name)
    output=model.predict(df)
    render_template('result.html',prediction_text=output)


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

## Home.html

- Home

Prediction

# CHRONIC KIDNEY DISEASE PREDICTION

## Indexnew.html :

**Chronic Kidney Disease**

A Machine Learning Web App, Built with Flask

Predict

## Result.html :

**Chronic Kidney Disease**

A Machine Learning Web App, Built with Flask

{% if prediction_text==1 %}

# Prediction: Oops! You have Chronic Kidney Disease.

{% elif prediction_text==0 %}

# Prediction: Great! You DON'T have Chronic Kidney Disease

{% endif %}