Early Prediction for Chronic Kidney Disease Detection: A Progressive Approach to Health Management

INTRODUCTION

1.1 OVERVIEW

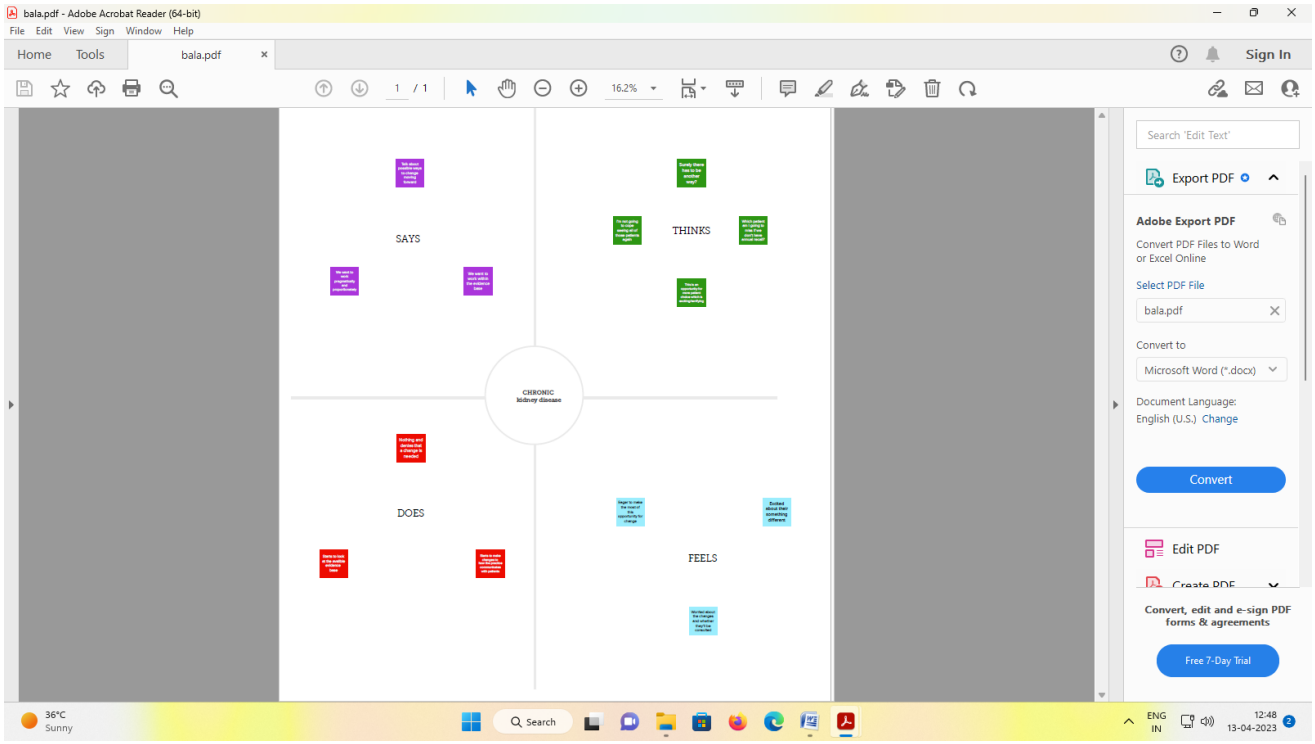
Chronic Kidney Disease (CKD) is a major medical problem and can be cured if treated in the early stages. Usually, people are not aware that medical tests we take for different purposes could contain valuable information concerning kidney diseases. Consequently, attributes of various medical tests are 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.

1.2 PURPOSE

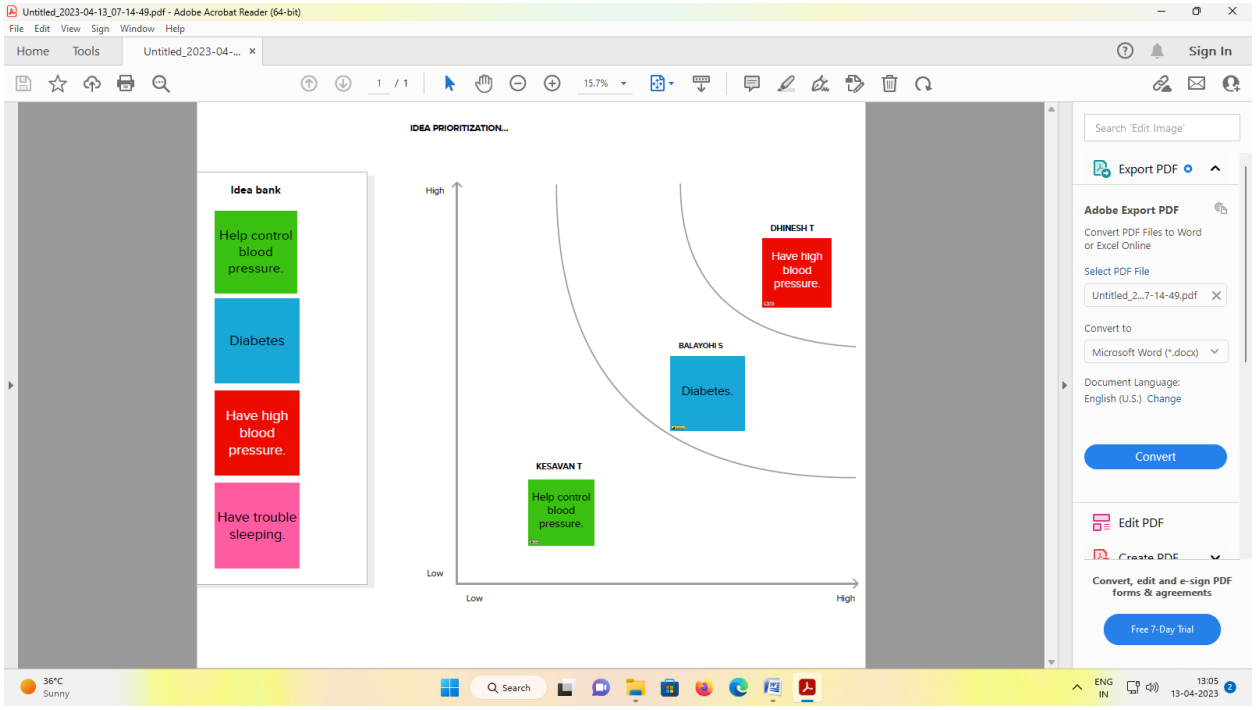
In todays world as we know most of the people are facing so many disease and as this can be curedif we treat people in early stages this project can use a pretrained model to predict the Chronic Kidney Disease which can help in treatments of peoples who are suffer from this disease

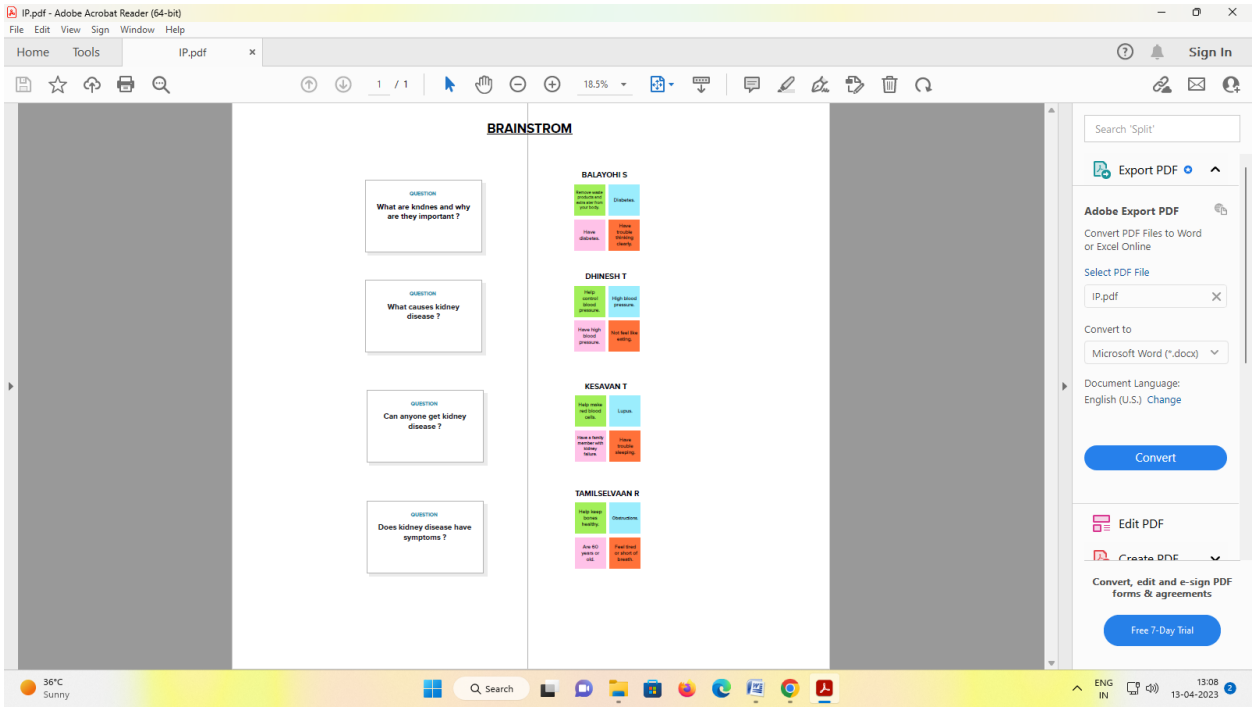
PROBLEM DEFINITION & DESIGN THINKING

2.1 EMPATHY MAP

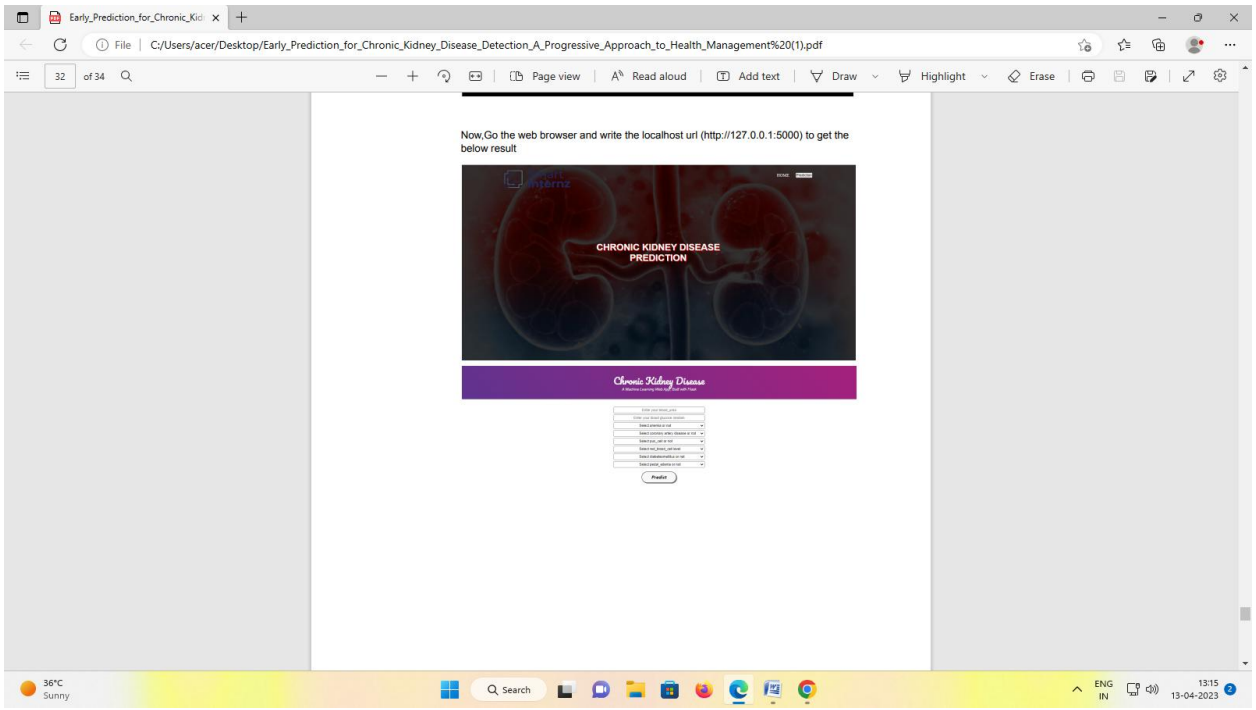


2.2 IDEATION & BRAINSTORMING MAP





RESULT


Early_Prediction_for_Chronic_Kid
File | C:/Users/acer/Desktop/Early_Prediction_for_Chronic_Kidney_Disease_Detection_A_Progressive_Approach_to_Health_Management%20(1).pdf
32 of 34
Page view
Read aloud
Add text
Draw
Highlight
Erase
Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result
CHRONIC KIDNEY DISEASE PREDICTION
Chronic Kidney Disease
Predict
36°C
Sunny
Search
ENG
IN
13:15
13-04-2023

ADVANTAGES AND DISADVANTAGES

- Having a chronic disease can make it feel as if your life has spun out of control. But, knowledge really *is* power. The things we worry about when we don't know what to expect can be far worse than what may really happen. Learning about your kidneys is a way to feel more in control of your life and health again. In this section, we will cover:

- **Make Urine**. Your kidneys make urine to get rid of wastes and extra water. Wastes come from some foods, breaking down medicines, and even just moving your muscles.
- **Balance Minerals.** Your muscles, blood vessels, nerves, and bones need precise amounts of minerals in your blood all the time. Kidneys sense the levels of minerals in your blood. They hold onto what you need and send the rest to your bladder, as urine.
- **Keep Your Bones Strong.** You need the right level of calcium in your blood all the time to live. Your bones are a "storage bank" for calcium and phosphorus. When you need more blood calcium, the kidneys send out a hormone. Active vitamin D is a hormone that lets your gut absorb calcium from foods. If the hormone signal does not work, your body will pull calcium out of your bones, which can make them weak and more likely to break.
- **Help Keep Acid and Base Balance in Your Body.** The pH in your body is close to neutral, but may be a little alkaline, or base (7.38 to 7.42). A pH that is too high or too low can be fatal. Kidneys work with your lungs to keep the right pH level.
- **Glomerular (glom-EAR-you-ler) diseases** attack blood vessels in the nephrons. Focal segmental glomerulosclerosis (FSGS) is one.

APPLICATION

- ✓ Chronic kidney disease (CKD) includes all clinical features and complications during the progression of various kidney conditions towards end-stage renal disease (ESRD).

- ✓ These conditions include immune and inflammatory disease such as: primary and hepatitis C virus (HCV)-related glomerulonephritis; infectious disease such as pyelonephritis with or without reflux and tuberculosis; vascular disease such as chronic ischemic nephropathy; hereditary and congenital disease such as polycystic disease and congenital cystic dysplasia; metabolic disease including diabetes and hyperuricemia; and systemic disease (collagen disease, vasculitis, myeloma).

- ✓ During the progression of CKD, ultrasound imaging and color Doppler imaging (US–CDI) can differentiate the etiology of the renal damage in only 50–70% of cases. Indeed, the end-stage kidney appears shrunken, reduced in volume (Ø < 9 cm), unstructured, amorphous, and with acquired cystic degeneration (small and multiple cysts involving the cortex and medulla) or nephrocalcinosis, but there are rare exceptions, such as polycystic kidney disease, diabetic nephropathy, and secondary inflammatory nephropathies.

CONCLUSION

- **CKD** is **a condition in which the kidneys are damaged and cannot filter blood as well as they should.** Because of this, excess fluid and waste from blood remain in the body and may cause other health problems, such as heart disease and stroke.
- **Severe forms of kidney disease which requires dialysis are curable in some instances**. Even if it is not curable, the patient can still lead a meaningful life while on dialysis. Kidney is the only vital organ which can be replaced long term by a machine with reasonable success.

FUTURE SCOPE

After defining chronic kidney disease (CKD), discussing the classification of CKD, and defining end-stage renal disease (ESRD), this chapter look at the problems with international comparisons of ESRD data before covering the prevalence of CKD in Western countries, Asian countries, and in developing countries, and the incidence of CKD in Western countries and in ethnic minorities in the West. It also covers the prevalence and incidence of treated ESRD, and the global cost of CKD and ESRD

APPENDIX

```
import pandas as pd

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("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','pcv','white_blood_cell_count','red_blood_cell_count',
'hypertension','diabetesmellitus','coronary_artery_disease','appetite',
'pedal_edema','anemia','class']

data.columns

data.info()

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['pcv'].fillna(data['pcv'].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=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['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]))

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_edma','appetite','batcteria','class','coronary_artery_dise
ase','diabetesmellit','hyp ertension','pus_cell_clumps','red_blood_cells']

from sklearn.preprocessing import lableEncoder

for i in catcols:

print("LABLE ENCODING OF:",i)

LEi=LableEncoder()

print(c(data[i]))

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

print(c(data[i]))

print("*"*100)

contcols=set(data.types[sata.type!='0'].index.values)

print(contcols) 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.remove('white_blood_cell_count')

print(catcols)

catcols=['anemia','pedal_edma','appetite','batcteria','class','coronary_artery_dise
ase','diabetesmellit','hyp ertension','pus_cell_clumps','red_blood_cells']

from sklearn.preprocessing import lableEncoder

for i in catcols:

print("LABLE ENCODING OF:",i)

LEi=LableEncoder()

print(c(data[i]))

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

print(c(data[i]))

print("*"*100) contcols=set(data.types[sata.type!='0'].index.values)

print(contcols) 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')

contcols.add('specific_gravity')

cantcols.add('albumin')

cantcols.add('sugar')

print(catcols)

data['coronary_artery_disease']
```

```
c(data[’coronary_artery_diease’])

data[’diabetesmellitus’]=sata.diabetesmellitus.replace(to_replace={’\tno’:’no’,’\tyes’:’yes’,’yes’:})

c(data[’diabetesmellitus’])

import matplotlib.pyplot as pltisport the matplotlib libaray

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

plt.scatter(datal age’], datal blood pressure),color="blue")

plt.xlabel(age) #set the Label for x-axis

plt.ylabel(’blood pressure) eset 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 :

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

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

plt.xlabel(column, fontsize=20)

plotnumber+=1

plt.show()

fax-pit.subplots(figsize=(18,10))

sns.heatmap(data.corr(), annot=True, fat=".2f", ax-ax,
linewidth=8.5,linecolor="orange")

plt.xticks(rotation-45)

plt.yticks(rotation=15)

plt.show()
```

```
sns.countplot(data['class'])

from sklearn.preprocessing import StandardScaler

sc-StandardScaler()

x_bal-sc.fit_transform(x)

selcols=['red_cells','pus_cell','blood gluose
random','blood_urea','pes=dal_edema','anemia','diabetesmelli
tus','coronary_artery_disease']

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

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

print(x.shape)

print(y.shape)

fromsklearn.model_selection import train_test_split

import tensorflow

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

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

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

classification.add(Dense (64, 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)

from sklearn.linear model import LogisticRegression

Igr = Logistic Regression()

Igr.fit(x train,y train)
```

```
from sklearn metrics import accuracy score, classification

y pred - 1gr.predict([[1.1‘,121,000000,36,0,0,0,1]])

printfy pred)

(y_pred)

print(y pred)

(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=[

(Loge, LogisticRegression()).. Candoni orestclassifier()),

("Decisionfree" „Decision treeclassifier()),

]

resulta=[]

names=[]

scoring = [’accuracy, precision weighted", "recallweighted", "f1 weighted" "roc auch’]
```

target names ’NO CKD, CKD | for name, model in models:

for name, model in model:

kfold - model_selection.Fold(n_splits-5, shufflestra, random_state=90210)

cv_results - model selection.cross_validate(model, x train, y train, cvskfold, scoring-scoring)

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(cv results)

this def 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 confusion_matrix

cm confusion matrix(y_test, y predict)

cm

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

plt.heatmap(cm, cap='Blues', annot=True, xticklabels=['no ckd', 'ckd'],
yticklabels-["no ckd', 'cku'])

plt.xlabel('Predicted values")

plt.ylabel('Actual values')

plt.tittle('confusion Matrix for RandomForestClassifier')

plt.show()

from sklearn.metrics import confusion_matrix

cm confusion matrix(y_test, y pred)

cm plt.figure(figsize-(8,6))

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

plt.xlabel('Predicted values")

plt.ylabel('Actual values')

plt.title('Confusion Matrix for ANN model')

plt.show()

from sklearn.metrics import confusion_matrix

cm confusion matrix(y_test, y pred)

cm plt.figure(figsize-(8,6))

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

plt.xlabel('Predicted values")
```

```
plt.ylabel('Actual values')

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'] # fit time metrics

results_long.loc[~results_long['metrics'].isin(time_metrics)]

results_long_nofit.sort_values(by-'values')

results_long_fit results_long.loc[results_long['metrics'].isin(time_metrics)]

results_long_fit.sort_values(by-'values')

pickle.dumb(lgr,open('CKD.PKL','wbfrom flask'))

impport flash,render_template,request

importnumpy as np

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',method=['POST'])

def predict

input_feature=[float(x)for x in request.from.values()]

feature_value=[np.array(input_features)]

feture_name=['blood urea','b;ood glucose
random','anemia','coronary_artery_diasease','pus_cell','red_blo
od_cell','diabetesmellitus','peda;_edema']

df=pd.DataFrame(feture_value,columns=features_name
output=model.presition(pf))

return render_template('result.html',predictipon_text=output)

if_name_=='_main_':

app.run(debug=true)
```

* * * * * * *