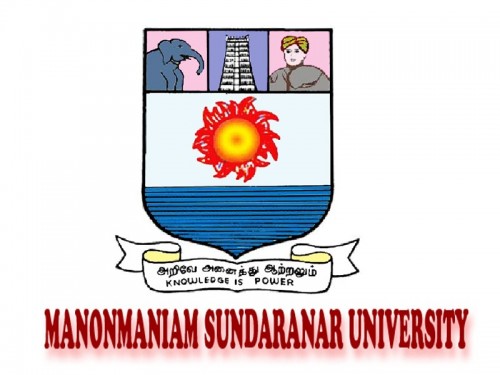
**Kamarajar Government Arts College,**

**Surandai,**

**Tenkasi (Dt)-627859.**



**April - 2023**

**Early Prediction for Chronic Kidney Disease Detection: A Man Progressive Approach to Health management**

**Team ID :** NM2023TMID20522

**Team Leader :** A.MOHAMED BASITH

**Reg No :** 20201601506125

**Team Member :** A.MOHAMED BEERUSHA

**Reg No :** 20201061506122

**Team Member :** A.MOHAMED MIRSATH

**Reg No :** 20201061506118

**Team Member :** R.MARISELVAM

**Reg No :** 20201361506118

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.no**  01 | **Title**  Introduction | **Pg.no**  01 |
| 02 | Problem Definition & Design Thinking | 11 |
| 03 | Result | 12 |
| 04 | Advantages & Disadvantages | 20 |
| 05 | Application | 21 |
| 06 | Conclusion | 23 |
| 07 | Future Scope | 24 |
| 08 | Appendix | 25 |

**INTRODUCTION:**

Chronic kidney disease (CKD) refers to a long-term, progressive decline in kidney function. The kidneys play an essential role in filtering waste and excess fluid from the blood, as well as regulating blood pressure, electrolyte balance, and red blood cell production. CKD is a common condition that affects millions of people worldwide, and it can lead to serious complications, such as cardiovascular disease, anemia, and bone disorders. CKD can develop due to a variety of factors, including diabetes, high blood pressure, kidney infections, and genetic disorders.

Chronic kidney disease (CKD) is a long-term condition that involves the progressive loss of kidney function over time. The kidneys are vital organs that are responsible for filtering waste and excess fluids from the blood, maintaining electrolyte balance, regulating blood pressure, and producing hormones that stimulate the production of red blood cells.

CKD is typically diagnosed when there is evidence of kidney damage or a decline in kidney function, as indicated by abnormal blood tests, urine tests, or imaging studies. The severity of CKD is classified based on the estimated glomerular filtration rate (eGFR), which is a measure of how well the kidneys are filtering the blood.

There are several stages of CKD, ranging from mild (stage 1) to severe (stage 5). In the early stages of CKD, there may be few or no symptoms, and the condition may be detected incidentally during routine blood or urine tests. As CKD progresses, however, symptoms may develop, such as fatigue, fluid retention, decreased appetite, and changes in urination patterns.

The most common causes of CKD are diabetes and high blood pressure, which can damage the kidneys over time. Other risk factors for CKD include obesity, smoking, family history of kidney disease, and certain medical conditions such as autoimmune disorders or kidney infections.

Early prediction for CKD detection is a proactive approach to health management that involves identifying individuals who are at risk of developing CKD before significant kidney damage occurs. This approach allows for early intervention and treatment, which can help to slow or prevent the progression of the disease and reduce the risk of complications.

By identifying individuals who are at high risk for CKD, healthcare providers can implement strategies to prevent or delay the onset of the disease. This may include lifestyle modifications, such as weight loss or increased physical activity, as well as medical interventions, such as blood pressure or glucose-lowering medications.

**Overview of Chronic Kidney Disease Predictions :**

**Data collection:**

Chronic Kidney Disease (CKD) prediction involves collecting data related to various demographic, clinical, and laboratory factors that are known to be associated with CKD. Here are some key factors to consider when collecting data for CKD prediction:

Demographic data: This includes age, gender, race, and ethnicity, which are important factors in CKD prediction.

Medical history: This includes any pre-existing conditions such as diabetes, hypertension, or heart disease, as well as any past surgeries or hospitalizations.

Lifestyle factors: This includes smoking, alcohol consumption, and physical activity levels.

Clinical examination data: This includes blood pressure, body mass index (BMI), and other physical examination findings.

Laboratory tests: This includes blood tests for creatinine, estimated glomerular filtration rate (eGFR), and urine tests for proteinuria.

Imaging tests: This includes ultrasound or CT scans to evaluate the kidneys' size and structure.

**Data preprocessing:**

Data preprocessing is a critical step in preparing data for machine learning modules used in CKD prediction.Here are some essential preprocessing steps:

Data Cleaning: This involves handling missing values, outliers, and inconsistent or incorrect data.

Feature Scaling: This involves scaling the numerical features to a common range, such as between 0 and 1 or -1 and 1, to prevent one feature from dominating over the others during model training.

Feature Selection: This involves selecting relevant features that contribute to CKD prediction and removing irrelevant or redundant ones.

Data Transformation: Common transformations include converting categorical features to numerical ones using encoding methods such as one-hot encoding or label encoding.

Data Splitting: This involves splitting the data into training and validation sets. The training set is used to train the machine learning model, while the validation set is used to evaluate its performance.

**Feature engineering:**

Chronic kidney disease (CKD) is a serious and potentially life-threatening condition that requires early detection and intervention for effective management. Feature engineering is an important step in building a machine learning model for CKD prediction.

Here are some features that could be useful for CKD prediction: Age,Sex, Blood pressure,Diabetes,Body mass index (BMI),Smoking status, Serum creatinine , Blood urea nitrogen (BUN),Serum albumin, Hemoglobin , Urine protein

**Model training:**

The preprocess data is used to train machine learning algorithms, such as decision trees, random forests, logistic regression, or deep learning models, using various techniques such as supervised learning or time-series analysis. The data is typically split into training, validation, and testing sets to evaluate model performance and prevent overfitting.

**Model evaluation:**

The trained models are evaluated using various performance metrics, such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic (ROC) curve, to assess their effectiveness in predicting chronic kidney disease. Different models and techniques are compared to select the best-performing model.

**Model deployment:**

Prepare a trained model with relevant features. Choose a deployment platform, considering cost, scalability, and security. Package the model and its dependencies. Deploy the packaged model to the chosen platform. Test the model to ensure it's working correctly. and maintain the model to ensure continued accuracy over time.

**Model monitoring and maintenance:**

The deployed model needs to be continuously monitored and updated to ensure its accuracy and relevance over time. This may involve retraining the model with new data, fine-tuning hyper parameters, and handling concept drift or data drift to maintain its predictive performance.

**Purpose of Chronic Kidney Disease Predictions :**

**Improved operational efficiency:**

Streamline patient workflows to reduce wait times and automate processes. Utilize technology solutions like EHRs to track patient progress and reduce manual data entry. Optimize staffing by ensuring that the right staff members are in the right roles. Implement patient education to help them manage their symptoms and reduce complications. Monitor outcomes by tracking patient progress and measuring key performance indicators.

**Enhanced customer experience:**

Foster a culture of empathy and compassion among healthcare providers to improve patient-provider communication and trust.Provide personalized care plans that take into account patients' individual needs, preferences, and values. Engage patients and their families in shared decision-making and goal-setting to increase their sense of control and involvement in their care. Ensure access to support services like counseling, financial assistance, and transportation to alleviate patients' non-medical concerns. Use patient feedback to continuously improve care processes and address areas for improvement.

**Risk management and decision-making:**

Identify and manage risk factors: Risk factors for CKD include diabetes, high blood pressure, and a family history of kidney disease. Managing these risk factors through lifestyle changes and medication can help slow the progression of CKD.

Monitor kidney function: Regular kidney function tests can help detect CKD early, allowing for prompt intervention and treatment.

Consider treatment options: Depending on the severity of CKD, treatment options may include lifestyle changes, medication, dialysis, or kidney transplant. Careful consideration of the benefits and risks of each option is important for making an informed decision.

Communicate with healthcare providers: Open and honest communication with healthcare providers can help ensure that all options are explored and that the best course of action is taken.

Manage complications: Complications of CKD may include anemia, bone disease, and cardiovascular disease. Managing these complications through appropriate treatment and lifestyle changes can help improve outcomes for people with CKD.

In summary, managing risk factors, monitoring kidney function, considering treatment options, communicating with healthcare providers, and managing complications are all important for effective risk management and decision making for CKD.

**Research and analysis:**

Prevalence: CKD is a common condition worldwide, with an estimated prevalence of around 10% in adults. The prevalence increases with age and is higher in certain populations, such as those with diabetes or hypertension.

Risk factors: Several risk factors have been identified for CKD, including diabetes, hypertension, obesity, smoking, and a family history of kidney disease.

Diagnosis: Diagnosis of CKD is typically based on laboratory tests, including blood and urine tests. These tests can help determine the level of kidney function and the presence of protein or other abnormalities in the urine.

Stages: CKD is classified into stages based on the level of kidney function, with stage 1 being the mildest form and stage 5 indicating end-stage kidney disease (ESKD). The stage of CKD is important for guiding treatment and monitoring disease progression.

Complications: CKD can lead to several complications, including cardiovascular disease, anemia, bone disease, and fluid and electrolyte imbalances. Management of these complications is an important aspect of CKD treatment.

Treatment: Treatment of CKD aims to slow the progression of the disease, manage complications, and prepare for the possibility of kidney failure. This may involve lifestyle changes, such as diet and exercise, as well as medications to control blood pressure, blood sugar, and other factors.

Dialysis and transplantation: For patients with ESKD, dialysis or kidney transplantation may be necessary to replace the lost kidney function. These treatments can improve quality of life and prolong survival, but also have their own risks and complications.

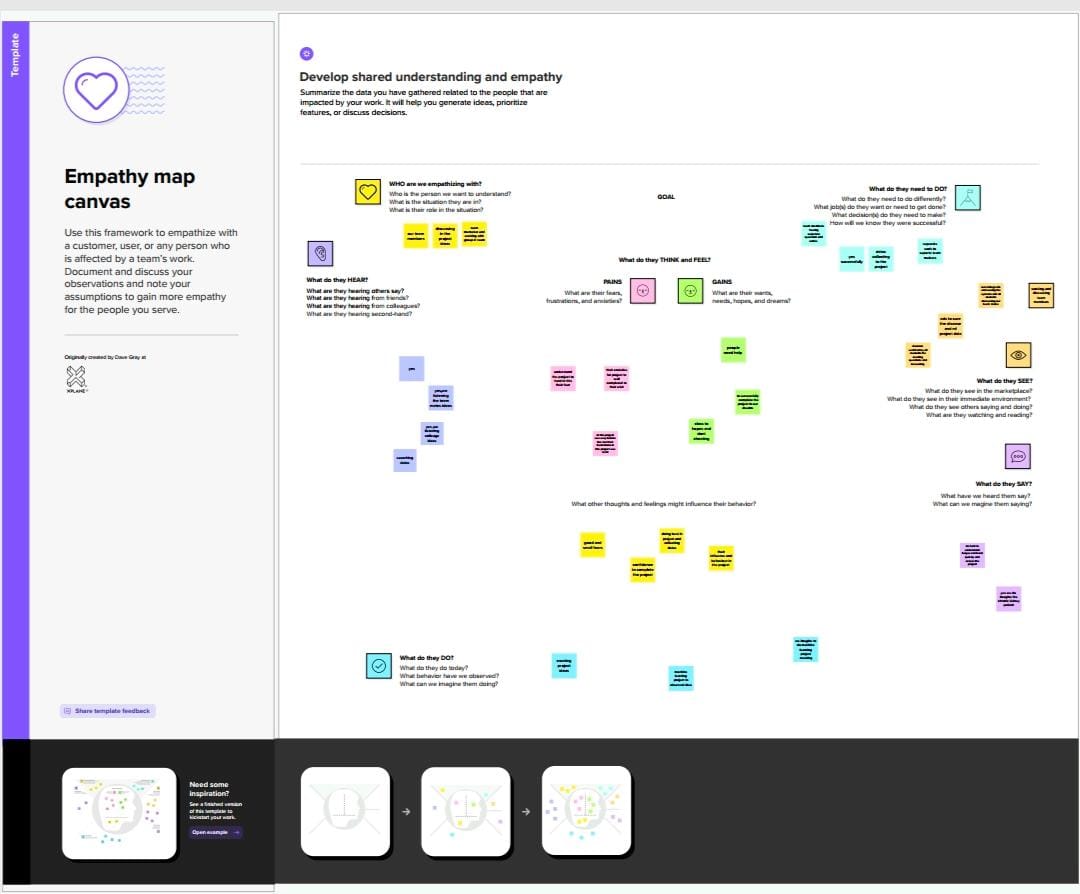
Prognosis: The prognosis for CKD varies depending on the stage of the disease, the presence of complications, and other factors. Early detection and treatment can help slow the progression of the disease and improve outcomes.

**Innovation and technological advancement:**

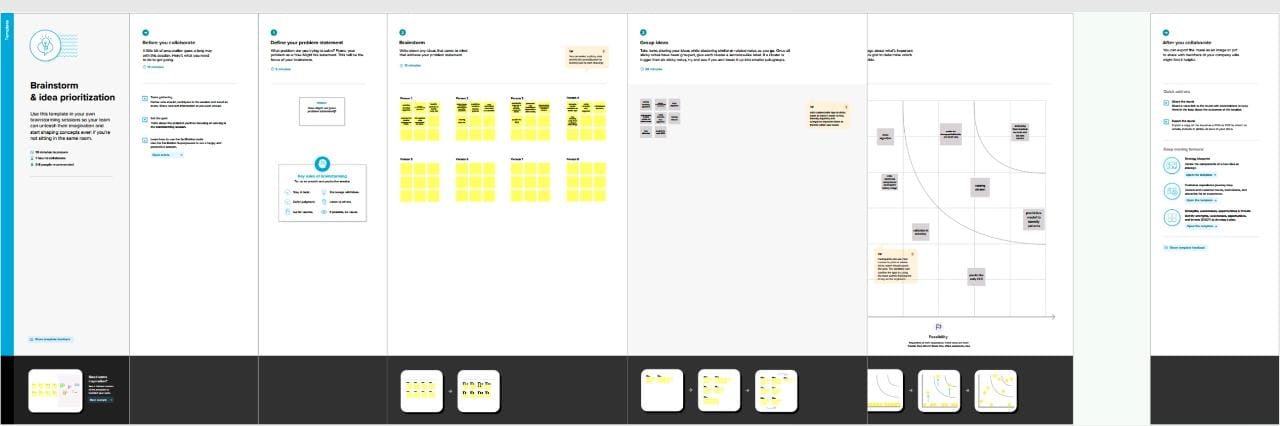
Incorporating Chronic kidney disease prediction using machine learning in a project can be driven by a desire to explore and adopt cutting-edge technologies, leverage advanced analytics techniques, and promote innovation in the aviation industry. It can also serve as a proof-of-concept for demonstrating the potential of machine learning in solving real-world problems and driving technological advancements in the field of aviation.

**Problem definition & designing thinking:**

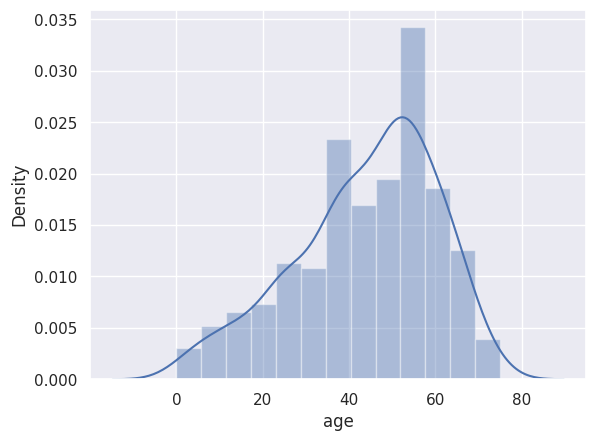
***Empathy map:***

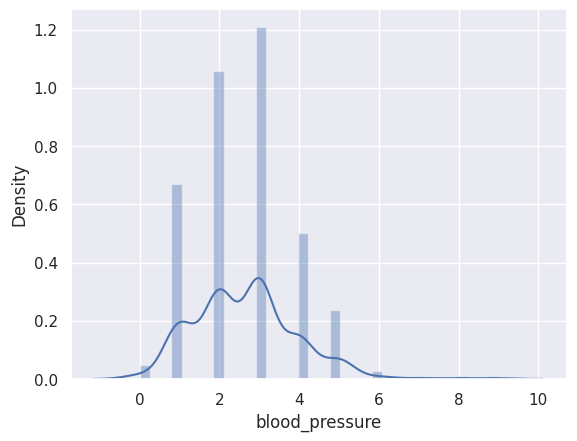
******

***Brainstroming Map:***

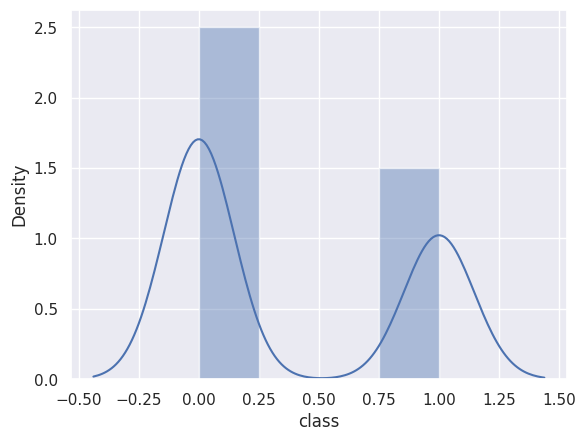
******

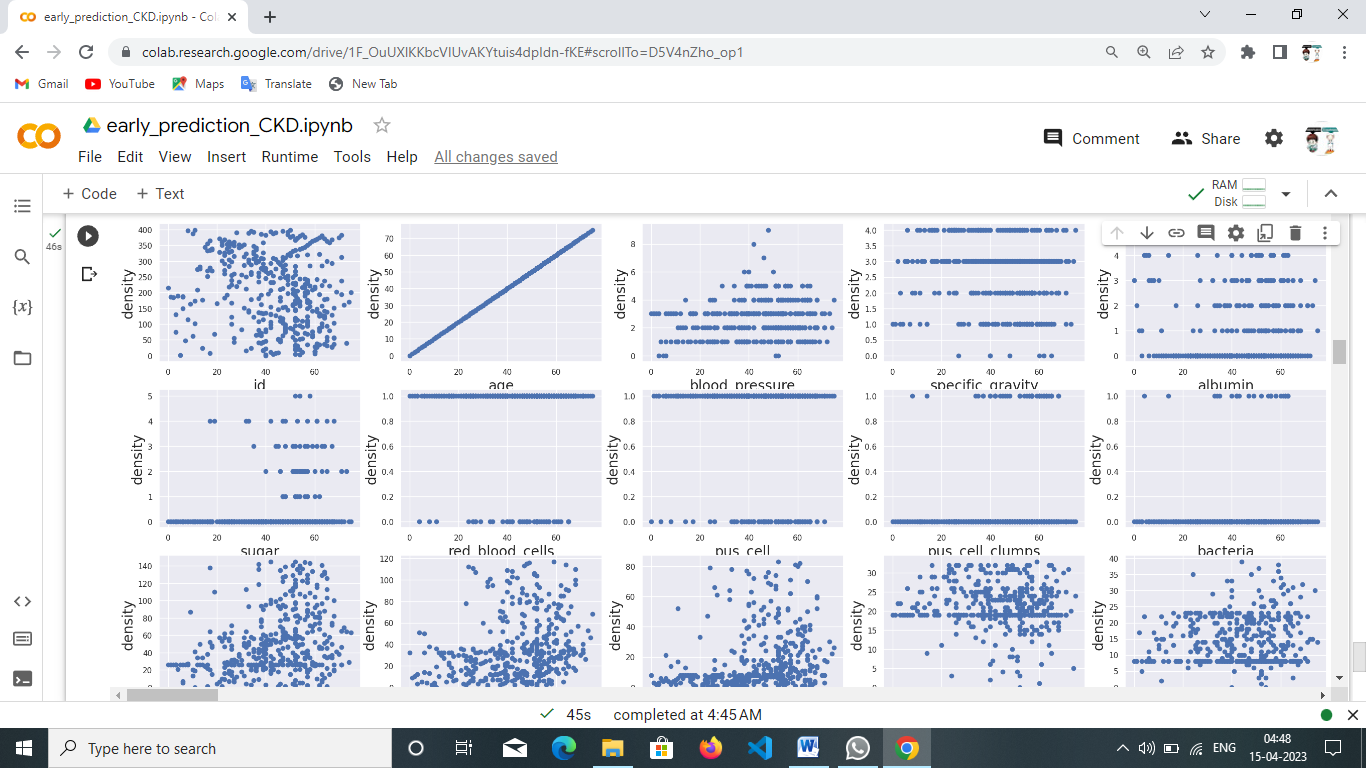
**Result:**

****

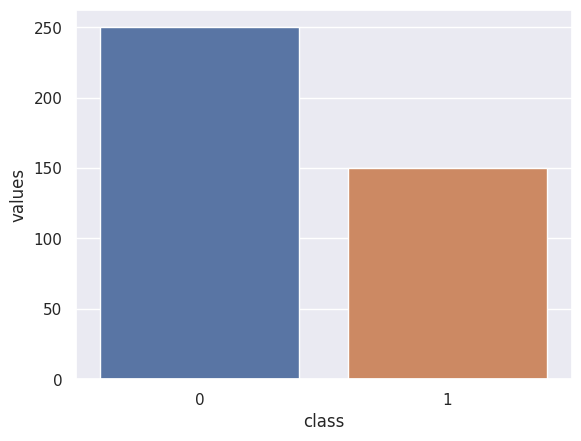
****

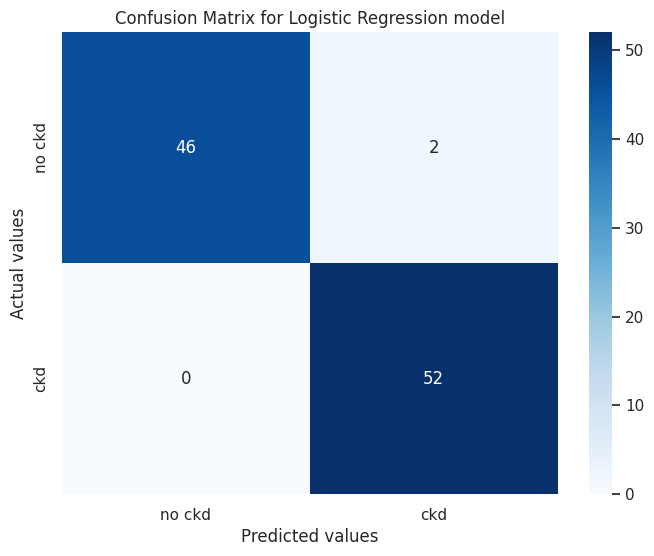
**……….**

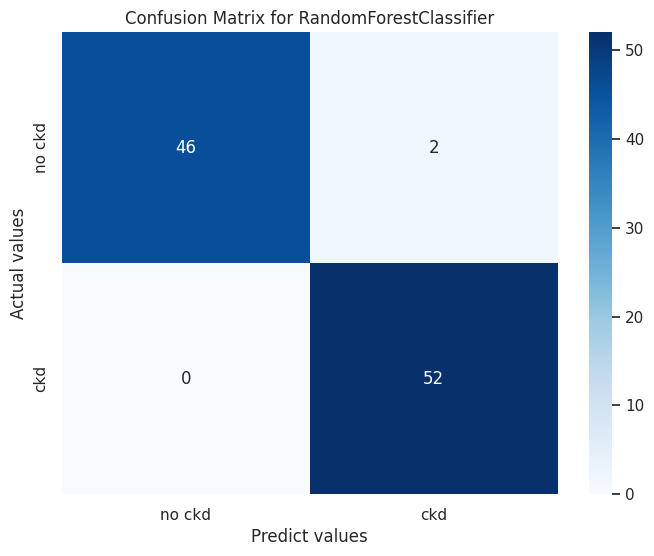
****

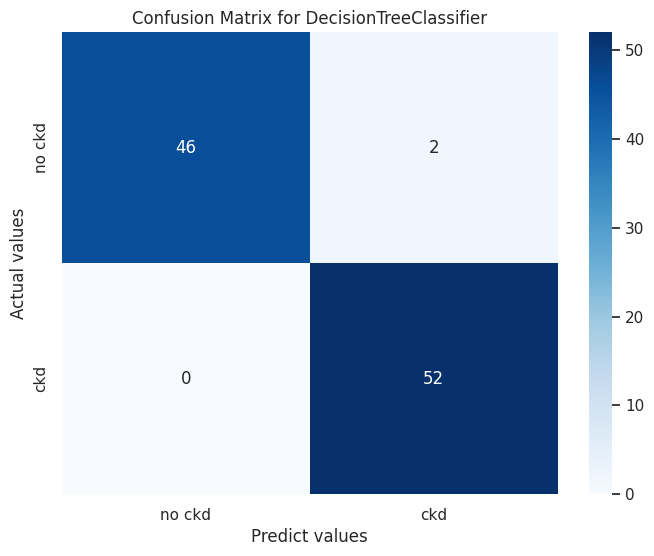


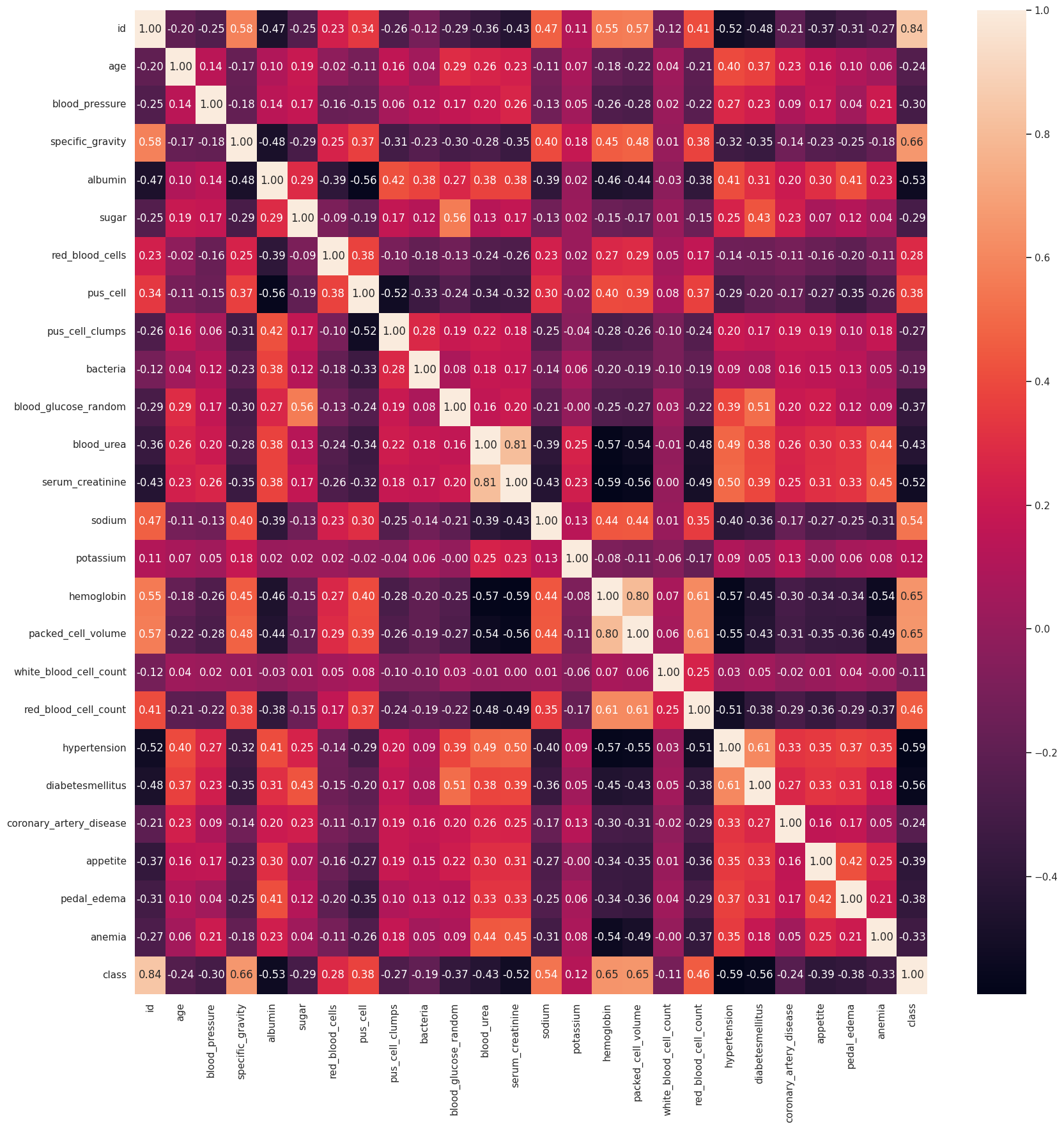
****

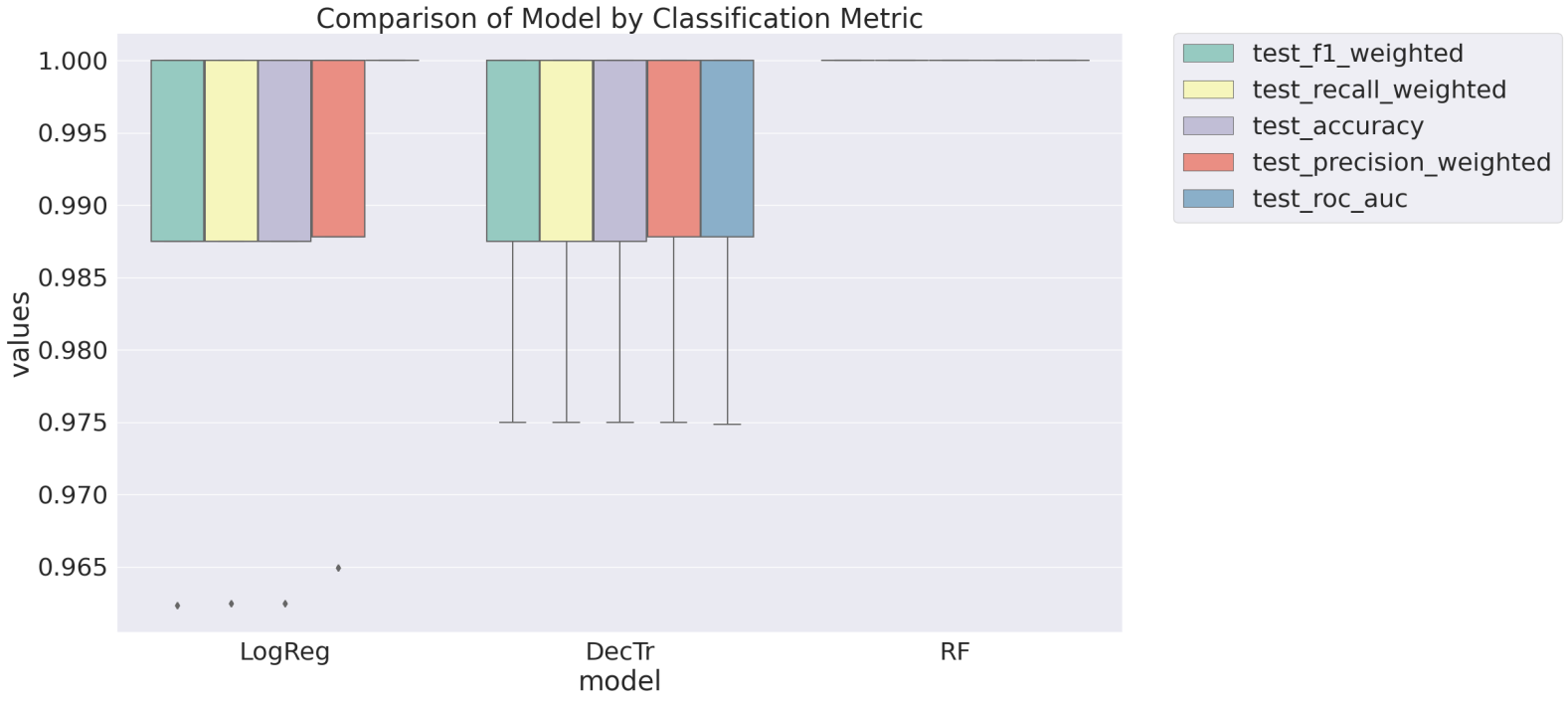




****







**Advantages:**

**Early Detection:**

Predicting CKD can help in its early detection, which can improve the chances of managing the disease and preventing complications. This can also help to slow the progression of the disease.

**Prevention:**

Predicting CKD can also help in preventing the development of the disease. By identifying risk factors early on, such as hypertension and diabetes, appropriate measures can be taken to prevent the onset of CKD.

**Improved Outcomes:**

Early detection and prevention of CKD can lead to improved outcomes, including a better quality of life, reduced healthcare costs, and reduced mortality rates.

**Patient Education:**

Predicting CKD can also help in educating patients about the disease and its risk factors. Patients can learn about the importance of maintaining a healthy lifestyle, regular check-ups, and managing any underlying conditions.

**Resource Allocation:**

Predicting CKD can also help in allocating resources for its prevention and management. Healthcare providers can identify high-risk patients and allocate resources accordingly, such as screening programs and patient education initiatives.

**Disadvantages:**

**False Positives and Negatives:**

Predictive models for CKD may not always accurately identify those at risk. False positives may lead to unnecessary testing and interventions, while false negatives may result in missed opportunities for early intervention.

**Cost:**

Predicting CKD can be costly, both in terms of financial resources and healthcare personnel time. This can be a barrier to implementation, particularly in resource-limited settings.

**Stigma:**

Predicting CKD may also result in stigmatization and discrimination towards individuals who are identified as high-risk, particularly if they are not experiencing any symptoms.

**Ethical Considerations:**

There are ethical considerations surrounding the use of predictive models for CKD, particularly regarding privacy, autonomy, and informed consent.

**Overdiagnosis and Overtreatment:**

Predictive models for CKD may lead to overdiagnosis and overtreatment of individuals who are at low risk. This can result in unnecessary testing and interventions that may cause harm or have adverse effects.

**Applications:**

**Diagnosis:**

CKD can be diagnosed through blood and urine tests that measure the kidney's ability to filter waste products from the body.

**Treatment:**

Treatment for CKD depends on the stage of the disease and may include medications to control blood pressure and blood sugar levels, as well as dietary changes.

**Monitoring:**

Regular monitoring of kidney function is essential for people with CKD, as it can help to detect any changes in the disease and guide treatment decisions.

**Prevention:**

People who are at risk of developing CKD, such as those with diabetes or high blood pressure, can take steps to prevent or delay the onset of the disease by managing their conditions effectively.

**Research:**

Researchers are studying CKD to better understand its causes and develop new treatments that can slow or stop the progression of the disease**.**

**Conclusion:**

Machine learning algorithms such as logistic regression, decision trees, and support vector machines can be effectively used for CKD prediction with high accuracy.

The risk factors for developing CKD include age, gender, hypertension, diabetes, obesity, smoking, and family history of kidney disease.

The use of biomarkers such as serum creatinine, blood urea nitrogen, and albuminuria can improve the accuracy of CKD prediction models.

Regular monitoring of blood pressure and blood glucose levels can help in early detection and prevention of CKD.

Lifestyle changes such as maintaining a healthy diet, regular exercise, and quitting smoking can help in reducing the risk of developing CKD.

The CKD prediction models developed in this project can be used by healthcare professionals to identify individuals at high risk for CKD and initiate early interventions.

In conclusion, the CKD prediction project highlights the importance of early detection and prevention of CKD through the use of machine learning algorithms and biomarkers. By identifying individuals at high risk for CKD, healthcare professionals can initiate early interventions and prevent or delay the onset of CKD.

**Future Scope:**

**Improved accuracy of predictive models:**

With the increasing availability of electronic health records and advanced data analytics techniques, it is expected that the accuracy of predictive models for CKD will continue to improve. This could lead to earlier detection and more effective management of the disease.

**Personalized risk assessment:**

Advances in genetic testing and personalized medicine may allow for more accurate risk assessment for CKD in individual patients. This could lead to targeted interventions to prevent or delay the onset of the disease.

**Integration with wearable devices and mobile health apps:**

The use of wearable devices and mobile health apps to monitor key health indicators such as blood pressure, heart rate, and glucose levels could provide valuable data for predicting CKD risk. Integration of this data with predictive models could lead to more accurate and timely risk assessment.

**Early intervention and prevention:**

As more is learned about the underlying causes of CKD, it may be possible to develop targeted interventions to prevent or delay the onset of the disease. This could involve lifestyle modifications, medication, or other therapies.

**Appendix :**

**Source Code :**

#Data collection & Preparation

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import missingno as msno

import os, sys

import pickle

from collections import Counter as c

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

sns.set()

df=pd.read\_csv("/content/drive/MyDrive/Colab Notebooks/kidney\_disease.csv")

head=df.head(n=7)

print(head)

#Data Preparation

old=df.columns

#rename the columns

df.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']

df.columns

df.shape

df.info()

df.isnull().any()

df.isnull().sum()

#output for data preparation

print(old)

print(df.columns)

print(df.shape)

print(df.info)

print(df.isnull().any())

print(df.isnull().sum())

#handling the missing value

#imputing null values

from sklearn.impute import SimpleImputer

imp\_mode=SimpleImputer(missing\_values=np.nan,strategy="most\_frequent")

df\_imputed=pd.DataFrame(imp\_mode.fit\_transform(df))

df\_imputed.columns=df.columns

impute=df\_imputed

#output for impute

print(impute)

#no null

print(impute.isnull().sum())

#finding unique values in the columns

for i in df\_imputed.columns:

  print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*",i,"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

  print()

  print(set(df\_imputed[i].tolist()))

  print()

print(df\_imputed['red\_blood\_cell\_count'].mode())

print(df\_imputed['white\_blood\_cell\_count'].mode())

print(df\_imputed['packed\_cell\_volume'].mode())

df\_imputed["red\_blood\_cell\_count"]=df\_imputed['red\_blood\_cell\_count'].apply(lambda x:'5.2' if x=='?' else x)

df\_imputed["white\_blood\_cell\_count"]=df\_imputed['white\_blood\_cell\_count'].apply(lambda x:'9800' if x=='?' else x)

df\_imputed["packed\_cell\_volume"]=df\_imputed["packed\_cell\_volume"].apply(lambda x:'41' if x=='?' else x)

#checking ? value to mode value & checking null values any()

#finding unique values in the columns

for i in df\_imputed.columns:

  print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*",i,"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

  print()

  print(set(df\_imputed[i].tolist()))

  print()

#data types actual df.dtypes

print(df.dtypes)

#data types in df\_imputed

print("data tpes in imputed variables:",df\_imputed.dtypes)

#exclude the object dtypes

print(df.select\_dtypes(exclude=["object"]).columns)

for i in df.select\_dtypes(exclude=["object"]).columns:

    df\_imputed[i]=df\_imputed[i].apply(lambda x: float(x))

#exclude object  df\_imputed changing to actual dtypes

actualdf=df\_imputed.dtypes

print("actual dtypes:",actualdf)

#label encoding catcorigal column to  numeric column

from sklearn import preprocessing

df\_enco=df\_imputed.apply(preprocessing.LabelEncoder().fit\_transform)

print(df\_enco.head(7))

#Exploratory Data Analysis

#Descriptve statistical Analysis

print(df\_enco.describe())

#Visual Analysis

#Univariate analysis

#sns.histplot(df\_enco.Age,kde=True)

def distplots(col):

    sns.distplot(df\_enco[col])

    plt.show()

for i in list(df\_enco.columns)[1:]:

    distplots(i)

#find and remove outliers of data

#def boxplots(col):

 # sns.boxplot(df\_enco[col])

  #plt.show()

#for i in list(df\_enco.columns)[1:]:

 # boxplots(i)

#Age vs Blood Pressure

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

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

plt.xlabel('age')#x-axis for label

plt.ylabel('blood\_pressure')#y-axis for label

plt.title("age vs blood scatter plot")

#Multivariate analysis sns.pairplot(df\_enco)

plt.figure(figsize=(30,25),facecolor='white')

plotnumber=1

for column in df\_enco:

  if plotnumber<=26:

    ax=plt.subplot(6,5,plotnumber)

    plt.scatter(df\_enco['age'],df\_enco[column])

    plt.xlabel(column,fontsize=20)

    plt.ylabel("density",fontsize=20)

  plotnumber+=1

plt.show()

#df\_enco.to\_csv("/content/drive/MyDrive/Colab Notebooks/kidney\_disease.csv")

temp=df\_enco["class"].value\_counts()

temp\_df=pd.DataFrame({'class':temp.index,'values':temp.values})

print(sns.barplot(x='class',y="values",data=temp\_df))

#Finding correlation between the independent columns

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

corr=df\_enco.corr()

sns.heatmap(corr,annot=True,fmt=".2f")

#Creating independent and dependent

x=df\_enco.drop(["id","class"],axis=1)

y=df\_enco["class"]

#Lets detect the label balance

from imblearn.over\_sampling import RandomOverSampler

from imblearn.under\_sampling import RandomUnderSampler

from collections import Counter

print(Counter(y))

#Lets balance the labels

ros=RandomOverSampler()

x\_ros,y\_ros=ros.fit\_resample(x,y)

print(Counter(y\_ros))

#Scaling the data

#Initial a MinMaxScaler and scale the features to between -1 and 1 to normalize them.

#The MinMaxScaler transforms features by scaling them to a given range

#The fit\_transform() method fits to the data and then transfprms it.we don't need to scale the labels.

#Scale the features to between -1 and 1

#Scaling is important in the algorithms such as support vector machine(SVM) and k-nearest neighbors(KNN) where distance

#between the data points is important.

from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler((-1,1))

x=scaler.fit\_transform(x\_ros)

y=y\_ros

#Applying PCA

#The code below has .95 for the number of components paramter

#It means that Scikit-learn chosse the minimum number of principal components such that 95%of thevariance is retained

import plotly.offline as py

py.init\_notebook\_mode(connected=True)

import plotly.graph\_objs as go

import plotly.tools as tls

from sklearn.decomposition import PCA

pca=PCA(.95)

x\_PCA=pca.fit\_transform(x)

print(x.shape)

print(x\_PCA.shape)

#with pca

#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\_PCA,y,test\_size=0.2,random\_state=2)

#Model building

#Training the model in multiple algorithm

#ANN model

#Importing the keras libraries and packages

import keras

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import Dropout

from keras.callbacks import ModelCheckpoint,EarlyStopping

from keras.models import Sequential,Model

from keras.optimizers import Adam

from sklearn.model\_selection import KFold

#creating model

#Creating ANN skelton view

def model():

  classifier=Sequential()

  classifier.add(Dense(34,input\_shape=(x\_train.shape[1],), activation='relu'))

  classifier.add(Dropout(0.2))

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

  classifier.add(Dropout(0.5))

  classifier.add(Dense(64,activation='relu'))

  classifier.add(Dropout(0.8))

  classifier.add(Dense(1,activation='sigmoid'))

#compiling the ANN model

  classifier.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])

  return classifier

#(x\_train.shape[1]+1)\*12

#Training the model

model=model()

model.summary()

history=model.fit(x\_train,y\_train,validation\_data=(x\_test,y\_test),epochs=100,verbose=1)

print(history)

#Random Forest Model

from sklearn.ensemble import RandomForestClassifier

rfc=RandomForestClassifier(n\_estimators=10,criterion='entropy')

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

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

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

print(y\_predict\_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)

print(y\_predict)

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

print(y\_predict\_train)

#Logistic Regression

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)

print(y\_predict)

#logistic regression

y\_pred=lgr.predict([[1,1,121.000000,36.0,0,0,1,0,0,1,100.0,98.0,7.7,111.0,3.2,12.5,44.0,8300]])

print(y\_pred)

(y\_pred)

#Decision Tree Classification

y\_pred=dtc.predict([[1,1,121.000000,36.0,0,0,1,0,0,1,100.0,98.0,7.7,111.0,3.2,12.5,44.0,8300]])

print(y\_pred)

#Random Forest Classifier

y\_pred=rfc.predict([[1,1,121.000000,36.0,0,0,1,0,0,1,100.0,98.0,7.7,111.0,3.2,12.5,44.0,8300]])

print(y\_pred)

(y\_pred)

model.save("ckd.h5")

#Testing the model

y\_pred=model.predict(x\_test)

y\_pred=(y\_pred>0.5)

print(y\_pred)

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

def predict\_exit(sample\_value):

  #convert list to numpy array

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

  #reshape because sample\_value contains only 1 record

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

  #feature scaling

  sample\_value=sc.transform(sample\_value)

  return classifier.predict(sample\_value)

test=model.predict([[1,1,121.000000,36.0,0,0,1,0,0,1,100.0,98.0,7.7,111.0,3.2,12.5,44.0,8300]])

if test==1:

  print('Prediction: High chance of CKD!')

else:

  print('Prediction: Low chance of CKD')

#Performance testing & Evaluate the results

#Testing model with multiple evalution metrics

#Compare the model

from sklearn import model\_selection

dfs=[]

models=[

        ('LogReg',LogisticRegression()),

        ('RF',RandomForestClassifier()),

        ('DecTr',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,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

#Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

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

print(cm)

#plotting confusion matrix

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()

#Making the confusion matrix

from sklearn.metrics import confusion\_matrix

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

print(cm)

#Plotting confusion\_matrix

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

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

plt.xlabel('Predict values')

plt.ylabel('Actual values')

plt.title('Confusion Matrix for RandomForestClassifier')

plt.show()

#Making the confusion matrix

from sklearn.metrics import confusion\_matrix

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

print(cm)

#Plotting confusion\_matrix

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

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

plt.xlabel('Predict values')

plt.ylabel('Actual values')

plt.title('Confusion Matrix for DecisionTreeClassifier')

plt.show()

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

#making the confusion matrix

from sklearn.metrics import confusion\_matrix

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

print(cm)

#Plotting confusion\_matrix

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

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

plt.xlabel('Predict values')

plt.ylabel('Actual values')

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

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'] # fit time metrics

## PERFORMANCE METRICS

results\_long\_nofit=results\_long.loc[~results\_long['metrics'].isin(time\_metrics)] # get df without fit data

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

## TIME METRICS

results\_long\_fit=results\_long.loc[results\_long['metrics'].isin(time\_metrics)] # df with fit data

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

#comparison of model by classification metric

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(dtc,open('CKD.pk1','wb'))

output:

id age bp sg al su rbc pc pcc ba \

0 0 48.0 80.0 1.020 1.0 0.0 NaN normal notpresent notpresent

1 1 7.0 50.0 1.020 4.0 0.0 NaN normal notpresent notpresent

2 2 62.0 80.0 1.010 2.0 3.0 normal normal notpresent notpresent

3 3 48.0 70.0 1.005 4.0 0.0 normal abnormal present notpresent

4 4 51.0 80.0 1.010 2.0 0.0 normal normal notpresent notpresent

5 5 60.0 90.0 1.015 3.0 0.0 NaN NaN notpresent notpresent

6 6 68.0 70.0 1.010 0.0 0.0 NaN normal notpresent notpresent

... pcv wc rc htn dm cad appet pe ane classification

0 ... 44 7800 5.2 yes yes no good no no ckd

1 ... 38 6000 NaN no no no good no no ckd

2 ... 31 7500 NaN no yes no poor no yes ckd

3 ... 32 6700 3.9 yes no no poor yes yes ckd

4 ... 35 7300 4.6 no no no good no no ckd

5 ... 39 7800 4.4 yes yes no good yes no ckd

6 ... 36 NaN NaN no no no good no no ckd

[7 rows x 26 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 400 entries, 0 to 399

Data columns (total 26 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 id 400 non-null int64

1 age 391 non-null float64

2 blood\_pressure 388 non-null float64

3 specific\_gravity 353 non-null float64

4 albumin 354 non-null float64

5 sugar 351 non-null float64

6 red\_blood\_cells 248 non-null object

7 pus\_cell 335 non-null object

8 pus\_cell\_clumps 396 non-null object

9 bacteria 396 non-null object

10 blood\_glucose\_random 356 non-null float64

11 blood\_urea 381 non-null float64

12 serum\_creatinine 383 non-null float64

13 sodium 313 non-null float64

14 potassium 312 non-null float64

15 hemoglobin 348 non-null float64

16 packed\_cell\_volume 330 non-null object

17 white\_blood\_cell\_count 295 non-null object

18 red\_blood\_cell\_count 270 non-null object

19 hypertension 398 non-null object

20 diabetesmellitus 398 non-null object

21 coronary\_artery\_disease 398 non-null object

22 appetite 399 non-null object

23 pedal\_edema 399 non-null object

24 anemia 399 non-null object

25 class 400 non-null object

dtypes: float64(11), int64(1), object(14)

memory usage: 81.4+ KB

Index(['id', 'age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr',

'bu', 'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',

'appet', 'pe', 'ane', 'classification'],

dtype='object')

Index(['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'],

dtype='object')

(400, 26)

<bound method DataFrame.info of id age blood\_pressure specific\_gravity albumin sugar \

0 0 48.0 80.0 1.020 1.0 0.0

1 1 7.0 50.0 1.020 4.0 0.0

2 2 62.0 80.0 1.010 2.0 3.0

3 3 48.0 70.0 1.005 4.0 0.0

4 4 51.0 80.0 1.010 2.0 0.0

.. ... ... ... ... ... ...

395 395 55.0 80.0 1.020 0.0 0.0

396 396 42.0 70.0 1.025 0.0 0.0

397 397 12.0 80.0 1.020 0.0 0.0

398 398 17.0 60.0 1.025 0.0 0.0

399 399 58.0 80.0 1.025 0.0 0.0

red\_blood\_cells pus\_cell pus\_cell\_clumps bacteria ... \

0 NaN normal notpresent notpresent ...

1 NaN normal notpresent notpresent ...

2 normal normal notpresent notpresent ...

3 normal abnormal present notpresent ...

4 normal normal notpresent notpresent ...

.. ... ... ... ... ...

395 normal normal notpresent notpresent ...

396 normal normal notpresent notpresent ...

397 normal normal notpresent notpresent ...

398 normal normal notpresent notpresent ...

399 normal normal notpresent notpresent ...

packed\_cell\_volume white\_blood\_cell\_count red\_blood\_cell\_count \

0 44 7800 5.2

1 38 6000 NaN

2 31 7500 NaN

3 32 6700 3.9

4 35 7300 4.6

.. ... ... ...

395 47 6700 4.9

396 54 7800 6.2

397 49 6600 5.4

398 51 7200 5.9

399 53 6800 6.1

hypertension diabetesmellitus coronary\_artery\_disease appetite \

0 yes yes no good

1 no no no good

2 no yes no poor

3 yes no no poor

4 no no no good

.. ... ... ... ...

395 no no no good

396 no no no good

397 no no no good

398 no no no good

399 no no no good

pedal\_edema anemia class

0 no no ckd

1 no no ckd

2 no yes ckd

3 yes yes ckd

4 no no ckd

.. ... ... ...

395 no no notckd

396 no no notckd

397 no no notckd

398 no no notckd

399 no no notckd

[400 rows x 26 columns]>

id False

age True

blood\_pressure True

specific\_gravity True

albumin True

sugar True

red\_blood\_cells True

pus\_cell True

pus\_cell\_clumps True

bacteria True

blood\_glucose\_random True

blood\_urea True

serum\_creatinine True

sodium True

potassium True

hemoglobin True

packed\_cell\_volume True

white\_blood\_cell\_count True

red\_blood\_cell\_count True

hypertension True

diabetesmellitus True

coronary\_artery\_disease True

appetite True

pedal\_edema True

anemia True

class False

dtype: bool

id 0

age 9

blood\_pressure 12

specific\_gravity 47

albumin 46

sugar 49

red\_blood\_cells 152

pus\_cell 65

pus\_cell\_clumps 4

bacteria 4

blood\_glucose\_random 44

blood\_urea 19

serum\_creatinine 17

sodium 87

potassium 88

hemoglobin 52

packed\_cell\_volume 70

white\_blood\_cell\_count 105

red\_blood\_cell\_count 130

hypertension 2

diabetesmellitus 2

coronary\_artery\_disease 2

appetite 1

pedal\_edema 1

anemia 1

class 0

dtype: int64

id age blood\_pressure specific\_gravity albumin sugar red\_blood\_cells \

0 0 48.0 80.0 1.02 1.0 0.0 normal

1 1 7.0 50.0 1.02 4.0 0.0 normal

2 2 62.0 80.0 1.01 2.0 3.0 normal

3 3 48.0 70.0 1.005 4.0 0.0 normal

4 4 51.0 80.0 1.01 2.0 0.0 normal

.. ... ... ... ... ... ... ...

395 395 55.0 80.0 1.02 0.0 0.0 normal

396 396 42.0 70.0 1.025 0.0 0.0 normal

397 397 12.0 80.0 1.02 0.0 0.0 normal

398 398 17.0 60.0 1.025 0.0 0.0 normal

399 399 58.0 80.0 1.025 0.0 0.0 normal

pus\_cell pus\_cell\_clumps bacteria ... packed\_cell\_volume \

0 normal notpresent notpresent ... 44

1 normal notpresent notpresent ... 38

2 normal notpresent notpresent ... 31

3 abnormal present notpresent ... 32

4 normal notpresent notpresent ... 35

.. ... ... ... ... ...

395 normal notpresent notpresent ... 47

396 normal notpresent notpresent ... 54

397 normal notpresent notpresent ... 49

398 normal notpresent notpresent ... 51

399 normal notpresent notpresent ... 53

white\_blood\_cell\_count red\_blood\_cell\_count hypertension diabetesmellitus \

0 7800 5.2 yes yes

1 6000 5.2 no no

2 7500 5.2 no yes

3 6700 3.9 yes no

4 7300 4.6 no no

.. ... ... ... ...

395 6700 4.9 no no

396 7800 6.2 no no

397 6600 5.4 no no

398 7200 5.9 no no

399 6800 6.1 no no

coronary\_artery\_disease appetite pedal\_edema anemia class

0 no good no no ckd

1 no good no no ckd

2 no poor no yes ckd

3 no poor yes yes ckd

4 no good no no ckd

.. ... ... ... ... ...

395 no good no no notckd

396 no good no no notckd

397 no good no no notckd

398 no good no no notckd

399 no good no no notckd

[400 rows x 26 columns]

id 0

age 0

blood\_pressure 0

specific\_gravity 0

albumin 0

sugar 0

red\_blood\_cells 0

pus\_cell 0

pus\_cell\_clumps 0

bacteria 0

blood\_glucose\_random 0

blood\_urea 0

serum\_creatinine 0

sodium 0

potassium 0

hemoglobin 0

packed\_cell\_volume 0

white\_blood\_cell\_count 0

red\_blood\_cell\_count 0

hypertension 0

diabetesmellitus 0

coronary\_artery\_disease 0

appetite 0

pedal\_edema 0

anemia 0

class 0

dtype: int64

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* id \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* age \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 11.0, 12.0, 14.0, 15.0, 17.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0, 27.0, 28.0, 29.0, 30.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 40.0, 41.0, 42.0, 43.0, 44.0, 45.0, 46.0, 47.0, 48.0, 49.0, 50.0, 51.0, 52.0, 53.0, 54.0, 55.0, 56.0, 57.0, 58.0, 59.0, 60.0, 61.0, 62.0, 63.0, 64.0, 65.0, 66.0, 67.0, 68.0, 69.0, 70.0, 71.0, 72.0, 73.0, 74.0, 75.0, 76.0, 78.0, 79.0, 80.0, 81.0, 82.0, 83.0, 90.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* blood\_pressure \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{100.0, 70.0, 140.0, 110.0, 80.0, 50.0, 180.0, 120.0, 90.0, 60.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* specific\_gravity \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{1.02, 1.025, 1.005, 1.015, 1.01}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* albumin \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{0.0, 1.0, 2.0, 3.0, 4.0, 5.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* sugar \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{0.0, 1.0, 2.0, 3.0, 4.0, 5.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* red\_blood\_cells \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'abnormal', 'normal'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* pus\_cell \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'abnormal', 'normal'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* pus\_cell\_clumps \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'notpresent', 'present'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* bacteria \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'notpresent', 'present'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* blood\_glucose\_random \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{22.0, 70.0, 74.0, 75.0, 76.0, 78.0, 79.0, 80.0, 81.0, 82.0, 83.0, 84.0, 85.0, 86.0, 87.0, 88.0, 89.0, 90.0, 91.0, 92.0, 93.0, 94.0, 95.0, 96.0, 97.0, 98.0, 99.0, 100.0, 101.0, 102.0, 103.0, 104.0, 105.0, 106.0, 107.0, 108.0, 109.0, 110.0, 111.0, 112.0, 113.0, 114.0, 115.0, 116.0, 117.0, 118.0, 119.0, 120.0, 121.0, 122.0, 123.0, 124.0, 125.0, 127.0, 128.0, 129.0, 130.0, 131.0, 132.0, 133.0, 134.0, 137.0, 138.0, 139.0, 140.0, 141.0, 143.0, 144.0, 146.0, 148.0, 150.0, 153.0, 156.0, 157.0, 158.0, 159.0, 160.0, 162.0, 163.0, 165.0, 169.0, 171.0, 172.0, 173.0, 176.0, 182.0, 184.0, 192.0, 201.0, 203.0, 204.0, 207.0, 208.0, 210.0, 213.0, 214.0, 215.0, 219.0, 220.0, 224.0, 226.0, 230.0, 233.0, 234.0, 238.0, 239.0, 241.0, 242.0, 246.0, 248.0, 250.0, 251.0, 252.0, 253.0, 255.0, 256.0, 261.0, 263.0, 264.0, 268.0, 269.0, 270.0, 273.0, 280.0, 288.0, 294.0, 295.0, 297.0, 298.0, 303.0, 307.0, 308.0, 309.0, 323.0, 341.0, 352.0, 360.0, 380.0, 410.0, 415.0, 423.0, 424.0, 425.0, 447.0, 463.0, 490.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* blood\_urea \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{1.5, 10.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0, 27.0, 28.0, 29.0, 30.0, 31.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 40.0, 41.0, 42.0, 44.0, 45.0, 46.0, 47.0, 48.0, 49.0, 50.0, 51.0, 52.0, 53.0, 54.0, 55.0, 56.0, 50.1, 58.0, 57.0, 60.0, 61.0, 64.0, 65.0, 66.0, 67.0, 68.0, 70.0, 71.0, 72.0, 73.0, 74.0, 75.0, 76.0, 77.0, 79.0, 80.0, 82.0, 85.0, 86.0, 87.0, 88.0, 89.0, 90.0, 92.0, 93.0, 94.0, 95.0, 96.0, 98.0, 98.6, 103.0, 106.0, 107.0, 111.0, 113.0, 114.0, 115.0, 118.0, 125.0, 132.0, 133.0, 137.0, 139.0, 142.0, 145.0, 146.0, 148.0, 150.0, 153.0, 155.0, 158.0, 162.0, 163.0, 164.0, 165.0, 166.0, 176.0, 180.0, 186.0, 191.0, 202.0, 208.0, 215.0, 217.0, 219.0, 223.0, 235.0, 241.0, 309.0, 322.0, 391.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* serum\_creatinine \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{0.8, 1.2, 1.4, 3.8, 1.8, 1.1, 1.9, 7.2, 4.0, 2.7, 2.1, 4.6, 4.1, 9.6, 5.2, 7.7, 7.3, 2.5, 2.0, 10.8, 3.0, 3.25, 15.0, 14.2, 24.0, 16.9, 18.0, 18.1, 1.5, 1.0, 32.0, 6.5, 0.5, 6.0, 7.5, 8.5, 48.1, 11.5, 12.0, 13.0, 13.5, 76.0, 16.4, 2.4, 2.9, 3.9, 3.4, 4.4, 5.9, 6.4, 11.9, 13.4, 2.8, 2.3, 3.3, 4.3, 1.3, 5.3, 6.3, 6.8, 0.6, 0.9, 0.4, 9.7, 9.2, 9.3, 0.7, 10.2, 1.7, 11.8, 12.2, 12.8, 13.8, 13.3, 2.2, 15.2, 3.2, 6.7, 5.6, 6.1, 7.1, 1.6, 2.6, 3.6}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* sodium \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{128.0, 129.0, 130.0, 131.0, 4.5, 132.0, 133.0, 135.0, 136.0, 134.0, 138.0, 139.0, 140.0, 141.0, 142.0, 137.0, 143.0, 145.0, 146.0, 147.0, 144.0, 150.0, 163.0, 104.0, 111.0, 113.0, 114.0, 115.0, 120.0, 122.0, 124.0, 125.0, 126.0, 127.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* potassium \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{2.5, 3.2, 3.7, 3.5, 4.0, 4.2, 5.8, 3.4, 6.4, 4.9, 4.1, 4.3, 5.2, 6.6, 7.6, 3.0, 4.6, 4.4, 4.5, 5.9, 5.5, 5.0, 5.4, 5.1, 5.6, 6.5, 39.0, 47.0, 3.6, 2.8, 2.7, 3.8, 3.3, 4.7, 4.8, 5.7, 5.3, 6.3, 2.9, 3.9}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* hemoglobin \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{3.1, 4.8, 5.6, 6.6, 7.6, 8.4, 7.7, 9.6, 10.8, 11.2, 11.3, 11.6, 12.2, 15.4, 12.4, 9.5, 12.6, 12.1, 12.7, 15.0, 15.6, 15.2, 16.1, 5.5, 6.0, 7.5, 8.0, 8.5, 9.0, 10.0, 10.5, 11.5, 11.0, 12.5, 12.0, 13.0, 13.5, 14.0, 14.5, 15.5, 16.5, 16.4, 16.9, 16.0, 16.6, 17.0, 17.1, 17.4, 17.5, 17.6, 7.9, 9.4, 9.9, 10.9, 10.4, 11.9, 11.4, 12.9, 13.9, 13.4, 14.4, 14.9, 15.9, 5.8, 6.8, 6.3, 7.3, 8.3, 8.2, 8.8, 8.7, 9.7, 9.8, 9.3, 9.2, 10.7, 10.3, 10.2, 11.8, 11.7, 12.3, 12.8, 13.8, 13.2, 13.7, 13.3, 14.3, 14.2, 14.8, 14.7, 15.7, 15.8, 15.3, 16.2, 16.3, 16.7, 16.8, 17.2, 17.3, 17.7, 17.8, 6.2, 6.1, 7.1, 8.6, 8.1, 9.1, 10.1, 10.6, 11.1, 13.6, 13.1, 14.1, 14.6, 15.1}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* packed\_cell\_volume \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'52', '21', '47', '22', '37', '30', '34', '?', '18', '51', '54', '35', '33', '9', '46', '45', '19', '53', '23', '26', '42', '17', '31', '20', '28', '48', '41', '38', '27', '43', '32', '39', '29', '15', '24', '50', '14', '49', '36', '44', '25', '40', '16'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* white\_blood\_cell\_count \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'14600', '9800', '7500', '10500', '10700', '15200', '5000', '12800', '12500', '6600', '4500', '26400', '9400', '11500', '21600', '5700', '8100', '12200', '5300', '6300', '12700', '7000', '11900', '11200', '5400', '11000', '7900', '11400', '5100', '7300', '18900', '5500', '8300', '7400', '12300', '10900', '5800', '13600', '5200', '4200', '6200', '2600', '14900', '6000', '9600', '19100', '12000', '4700', '12100', '6400', '9500', '15700', '7700', '13200', '6800', '5900', '8600', '2200', '11300', '9300', '8800', '10200', '8400', '11800', '4900', '12400', '4300', '?', '9000', '9100', '9700', '6900', '16300', '7200', '6700', '8000', '5600', '8500', '9200', '9900', '7800', '10800', '6500', '8200', '4100', '3800', '10400', '16700', '10300', '7100'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* red\_blood\_cell\_count \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'6.5', '2.1', '?', '5.8', '2.3', '4.9', '2.7', '4.4', '5', '4.1', '2.9', '5.4', '5.5', '4', '5.1', '3.7', '3.6', '3.5', '4.3', '3.3', '3', '8', '6.2', '5.9', '2.6', '3.1', '2.8', '6.3', '4.6', '3.8', '5.6', '3.9', '2.5', '4.7', '4.5', '6.1', '5.3', '3.4', '4.2', '6.4', '4.8', '3.2', '5.2', '2.4', '5.7', '6'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* hypertension \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'no', 'yes'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* diabetesmellitus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'no', 'yes'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* coronary\_artery\_disease \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'no', 'yes'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* appetite \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'poor', 'good'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* pedal\_edema \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'no', 'yes'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* anemia \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'no', 'yes'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'ckd', 'notckd'}

0 5.2

Name: red\_blood\_cell\_count, dtype: object

0 9800

Name: white\_blood\_cell\_count, dtype: object

0 41

Name: packed\_cell\_volume, dtype: object

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* id \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* age \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 11.0, 12.0, 14.0, 15.0, 17.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0, 27.0, 28.0, 29.0, 30.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 40.0, 41.0, 42.0, 43.0, 44.0, 45.0, 46.0, 47.0, 48.0, 49.0, 50.0, 51.0, 52.0, 53.0, 54.0, 55.0, 56.0, 57.0, 58.0, 59.0, 60.0, 61.0, 62.0, 63.0, 64.0, 65.0, 66.0, 67.0, 68.0, 69.0, 70.0, 71.0, 72.0, 73.0, 74.0, 75.0, 76.0, 78.0, 79.0, 80.0, 81.0, 82.0, 83.0, 90.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* blood\_pressure \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{100.0, 70.0, 140.0, 110.0, 80.0, 50.0, 180.0, 120.0, 90.0, 60.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* specific\_gravity \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{1.02, 1.025, 1.005, 1.015, 1.01}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* albumin \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{0.0, 1.0, 2.0, 3.0, 4.0, 5.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* sugar \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{0.0, 1.0, 2.0, 3.0, 4.0, 5.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* red\_blood\_cells \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'abnormal', 'normal'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* pus\_cell \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'abnormal', 'normal'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* pus\_cell\_clumps \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'notpresent', 'present'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* bacteria \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'notpresent', 'present'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* blood\_glucose\_random \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{22.0, 70.0, 74.0, 75.0, 76.0, 78.0, 79.0, 80.0, 81.0, 82.0, 83.0, 84.0, 85.0, 86.0, 87.0, 88.0, 89.0, 90.0, 91.0, 92.0, 93.0, 94.0, 95.0, 96.0, 97.0, 98.0, 99.0, 100.0, 101.0, 102.0, 103.0, 104.0, 105.0, 106.0, 107.0, 108.0, 109.0, 110.0, 111.0, 112.0, 113.0, 114.0, 115.0, 116.0, 117.0, 118.0, 119.0, 120.0, 121.0, 122.0, 123.0, 124.0, 125.0, 127.0, 128.0, 129.0, 130.0, 131.0, 132.0, 133.0, 134.0, 137.0, 138.0, 139.0, 140.0, 141.0, 143.0, 144.0, 146.0, 148.0, 150.0, 153.0, 156.0, 157.0, 158.0, 159.0, 160.0, 162.0, 163.0, 165.0, 169.0, 171.0, 172.0, 173.0, 176.0, 182.0, 184.0, 192.0, 201.0, 203.0, 204.0, 207.0, 208.0, 210.0, 213.0, 214.0, 215.0, 219.0, 220.0, 224.0, 226.0, 230.0, 233.0, 234.0, 238.0, 239.0, 241.0, 242.0, 246.0, 248.0, 250.0, 251.0, 252.0, 253.0, 255.0, 256.0, 261.0, 263.0, 264.0, 268.0, 269.0, 270.0, 273.0, 280.0, 288.0, 294.0, 295.0, 297.0, 298.0, 303.0, 307.0, 308.0, 309.0, 323.0, 341.0, 352.0, 360.0, 380.0, 410.0, 415.0, 423.0, 424.0, 425.0, 447.0, 463.0, 490.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* blood\_urea \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{1.5, 10.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0, 26.0, 27.0, 28.0, 29.0, 30.0, 31.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 40.0, 41.0, 42.0, 44.0, 45.0, 46.0, 47.0, 48.0, 49.0, 50.0, 51.0, 52.0, 53.0, 54.0, 55.0, 56.0, 50.1, 58.0, 57.0, 60.0, 61.0, 64.0, 65.0, 66.0, 67.0, 68.0, 70.0, 71.0, 72.0, 73.0, 74.0, 75.0, 76.0, 77.0, 79.0, 80.0, 82.0, 85.0, 86.0, 87.0, 88.0, 89.0, 90.0, 92.0, 93.0, 94.0, 95.0, 96.0, 98.0, 98.6, 103.0, 106.0, 107.0, 111.0, 113.0, 114.0, 115.0, 118.0, 125.0, 132.0, 133.0, 137.0, 139.0, 142.0, 145.0, 146.0, 148.0, 150.0, 153.0, 155.0, 158.0, 162.0, 163.0, 164.0, 165.0, 166.0, 176.0, 180.0, 186.0, 191.0, 202.0, 208.0, 215.0, 217.0, 219.0, 223.0, 235.0, 241.0, 309.0, 322.0, 391.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* serum\_creatinine \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{0.8, 1.2, 1.4, 3.8, 1.8, 1.1, 1.9, 7.2, 4.0, 2.7, 2.1, 4.6, 4.1, 9.6, 5.2, 7.7, 7.3, 2.5, 2.0, 10.8, 3.0, 3.25, 15.0, 14.2, 24.0, 16.9, 18.0, 18.1, 1.5, 1.0, 32.0, 6.5, 0.5, 6.0, 7.5, 8.5, 48.1, 11.5, 12.0, 13.0, 13.5, 76.0, 16.4, 2.4, 2.9, 3.9, 3.4, 4.4, 5.9, 6.4, 11.9, 13.4, 2.8, 2.3, 3.3, 4.3, 1.3, 5.3, 6.3, 6.8, 0.6, 0.9, 0.4, 9.7, 9.2, 9.3, 0.7, 10.2, 1.7, 11.8, 12.2, 12.8, 13.8, 13.3, 2.2, 15.2, 3.2, 6.7, 5.6, 6.1, 7.1, 1.6, 2.6, 3.6}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* sodium \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{128.0, 129.0, 130.0, 131.0, 4.5, 132.0, 133.0, 135.0, 136.0, 134.0, 138.0, 139.0, 140.0, 141.0, 142.0, 137.0, 143.0, 145.0, 146.0, 147.0, 144.0, 150.0, 163.0, 104.0, 111.0, 113.0, 114.0, 115.0, 120.0, 122.0, 124.0, 125.0, 126.0, 127.0}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* potassium \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{2.5, 3.2, 3.7, 3.5, 4.0, 4.2, 5.8, 3.4, 6.4, 4.9, 4.1, 4.3, 5.2, 6.6, 7.6, 3.0, 4.6, 4.4, 4.5, 5.9, 5.5, 5.0, 5.4, 5.1, 5.6, 6.5, 39.0, 47.0, 3.6, 2.8, 2.7, 3.8, 3.3, 4.7, 4.8, 5.7, 5.3, 6.3, 2.9, 3.9}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* hemoglobin \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{3.1, 4.8, 5.6, 6.6, 7.6, 8.4, 7.7, 9.6, 10.8, 11.2, 11.3, 11.6, 12.2, 15.4, 12.4, 9.5, 12.6, 12.1, 12.7, 15.0, 15.6, 15.2, 16.1, 5.5, 6.0, 7.5, 8.0, 8.5, 9.0, 10.0, 10.5, 11.5, 11.0, 12.5, 12.0, 13.0, 13.5, 14.0, 14.5, 15.5, 16.5, 16.4, 16.9, 16.0, 16.6, 17.0, 17.1, 17.4, 17.5, 17.6, 7.9, 9.4, 9.9, 10.9, 10.4, 11.9, 11.4, 12.9, 13.9, 13.4, 14.4, 14.9, 15.9, 5.8, 6.8, 6.3, 7.3, 8.3, 8.2, 8.8, 8.7, 9.7, 9.8, 9.3, 9.2, 10.7, 10.3, 10.2, 11.8, 11.7, 12.3, 12.8, 13.8, 13.2, 13.7, 13.3, 14.3, 14.2, 14.8, 14.7, 15.7, 15.8, 15.3, 16.2, 16.3, 16.7, 16.8, 17.2, 17.3, 17.7, 17.8, 6.2, 6.1, 7.1, 8.6, 8.1, 9.1, 10.1, 10.6, 11.1, 13.6, 13.1, 14.1, 14.6, 15.1}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* packed\_cell\_volume \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'52', '21', '47', '22', '37', '30', '34', '18', '51', '54', '35', '33', '9', '46', '45', '19', '53', '23', '26', '42', '17', '31', '20', '28', '48', '41', '38', '27', '43', '32', '39', '29', '15', '24', '50', '14', '49', '36', '44', '25', '40', '16'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* white\_blood\_cell\_count \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'14600', '9800', '7500', '10500', '10700', '15200', '5000', '12800', '12500', '6600', '4500', '26400', '9400', '11500', '21600', '5700', '8100', '12200', '5300', '6300', '12700', '7000', '11900', '11200', '5400', '11000', '7900', '11400', '5100', '7300', '18900', '5500', '8300', '7400', '12300', '10900', '5800', '13600', '5200', '4200', '6200', '2600', '14900', '6000', '9600', '19100', '12000', '4700', '12100', '6400', '9500', '15700', '7700', '13200', '6800', '5900', '8600', '2200', '11300', '9300', '8800', '10200', '8400', '11800', '4900', '12400', '4300', '9000', '9100', '9700', '6900', '16300', '7200', '6700', '8000', '5600', '8500', '9200', '9900', '7800', '10800', '6500', '8200', '4100', '3800', '10400', '16700', '10300', '7100'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* red\_blood\_cell\_count \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'6.5', '2.1', '5.8', '2.3', '4.9', '2.7', '4.4', '5', '4.1', '2.9', '5.4', '5.5', '4', '5.1', '3.7', '3.6', '3.5', '4.3', '3.3', '3', '8', '6.2', '5.9', '2.6', '3.1', '2.8', '6.3', '4.6', '3.8', '5.6', '3.9', '2.5', '4.7', '4.5', '6.1', '5.3', '3.4', '4.2', '6.4', '4.8', '3.2', '5.2', '2.4', '5.7', '6'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* hypertension \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'no', 'yes'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* diabetesmellitus \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'no', 'yes'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* coronary\_artery\_disease \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'no', 'yes'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* appetite \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'poor', 'good'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* pedal\_edema \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'no', 'yes'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* anemia \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'no', 'yes'}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* class \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{'ckd', 'notckd'}

id int64

age float64

blood\_pressure float64

specific\_gravity float64

albumin float64

sugar float64

red\_blood\_cells object

pus\_cell object

pus\_cell\_clumps object

bacteria object

blood\_glucose\_random float64

blood\_urea float64

serum\_creatinine float64

sodium float64

potassium float64

hemoglobin float64

packed\_cell\_volume object

white\_blood\_cell\_count object

red\_blood\_cell\_count object

hypertension object

diabetesmellitus object

coronary\_artery\_disease object

appetite object

pedal\_edema object

anemia object

class object

dtype: object

data tpes in imputed variables: id object

age object

blood\_pressure object

specific\_gravity object

albumin object

sugar object

red\_blood\_cells object

pus\_cell object

pus\_cell\_clumps object

bacteria object

blood\_glucose\_random object

blood\_urea object

serum\_creatinine object

sodium object

potassium object

hemoglobin object

packed\_cell\_volume object

white\_blood\_cell\_count object

red\_blood\_cell\_count object

hypertension object

diabetesmellitus object

coronary\_artery\_disease object

appetite object

pedal\_edema object

anemia object

class object

dtype: object

Index(['id', 'age', 'blood\_pressure', 'specific\_gravity', 'albumin', 'sugar',

'blood\_glucose\_random', 'blood\_urea', 'serum\_creatinine', 'sodium',

'potassium', 'hemoglobin'],

dtype='object')

actual dtypes: id float64

age float64

blood\_pressure float64

specific\_gravity float64

albumin float64

sugar float64

red\_blood\_cells object

pus\_cell object

pus\_cell\_clumps object

bacteria object

blood\_glucose\_random float64

blood\_urea float64

serum\_creatinine float64

sodium float64

potassium float64

hemoglobin float64

packed\_cell\_volume object

white\_blood\_cell\_count object

red\_blood\_cell\_count object

hypertension object

diabetesmellitus object

coronary\_artery\_disease object

appetite object

pedal\_edema object

anemia object

class object

dtype: object

id age blood\_pressure specific\_gravity albumin sugar red\_blood\_cells \

0 0 40 3 3 1 0 1

1 1 5 0 3 4 0 1

2 2 54 3 1 2 3 1

3 3 40 2 0 4 0 1

4 4 43 3 1 2 0 1

5 5 52 4 2 3 0 1

6 6 60 2 1 0 0 1

pus\_cell pus\_cell\_clumps bacteria ... packed\_cell\_volume \

0 1 0 0 ... 30

1 1 0 0 ... 24

2 1 0 0 ... 17

3 0 1 0 ... 18

4 1 0 0 ... 21

5 1 0 0 ... 25

6 1 0 0 ... 22

white\_blood\_cell\_count red\_blood\_cell\_count hypertension \

0 69 30 1

1 53 30 0

2 67 30 0

3 59 17 1

4 65 24 0

5 69 22 1

6 87 30 0

diabetesmellitus coronary\_artery\_disease appetite pedal\_edema anemia \

0 1 0 0 0 0

1 0 0 0 0 0

2 1 0 1 0 1

3 0 0 1 1 1

4 0 0 0 0 0

5 1 0 0 1 0

6 0 0 0 0 0

class

0 0

1 0

2 0

3 0

4 0

5 0

6 0

[7 rows x 26 columns]

id age blood\_pressure specific\_gravity albumin \

count 400.000000 400.000000 400.000000 400.000000 400.00000

mean 199.500000 43.932500 2.645000 2.542500 0.90000

std 115.614301 16.284684 1.273983 1.086806 1.31313

min 0.000000 0.000000 0.000000 0.000000 0.00000

25% 99.750000 34.000000 2.000000 2.000000 0.00000

50% 199.500000 47.000000 3.000000 3.000000 0.00000

75% 299.250000 56.000000 3.000000 3.000000 2.00000

max 399.000000 75.000000 9.000000 4.000000 5.00000

sugar red\_blood\_cells pus\_cell pus\_cell\_clumps bacteria \

count 400.000000 400.000000 400.000000 400.000000 400.000000

mean 0.395000 0.882500 0.810000 0.105000 0.055000

std 1.040038 0.322418 0.392792 0.306937 0.228266

min 0.000000 0.000000 0.000000 0.000000 0.000000

25% 0.000000 1.000000 1.000000 0.000000 0.000000

50% 0.000000 1.000000 1.000000 0.000000 0.000000

75% 0.000000 1.000000 1.000000 0.000000 0.000000

max 5.000000 1.000000 1.000000 1.000000 1.000000

... packed\_cell\_volume white\_blood\_cell\_count red\_blood\_cell\_count \

count ... 400.000000 400.000000 400.000000

mean ... 25.375000 61.460000 26.657500

std ... 8.087515 27.136957 8.601754

min ... 0.000000 0.000000 0.000000

25% ... 20.000000 47.000000 23.000000

50% ... 27.000000 68.500000 30.000000

75% ... 30.000000 87.000000 30.000000

max ... 41.000000 88.000000 44.000000

hypertension diabetesmellitus coronary\_artery\_disease appetite \

count 400.000000 400.00000 400.000000 400.000000

mean 0.367500 0.34250 0.085000 0.205000

std 0.482728 0.47514 0.279231 0.404207

min 0.000000 0.00000 0.000000 0.000000

25% 0.000000 0.00000 0.000000 0.000000

50% 0.000000 0.00000 0.000000 0.000000

75% 1.000000 1.00000 0.000000 0.000000

max 1.000000 1.00000 1.000000 1.000000

pedal\_edema anemia class

count 400.000000 400.000000 400.000000

mean 0.190000 0.150000 0.375000

std 0.392792 0.357519 0.484729

min 0.000000 0.000000 0.000000

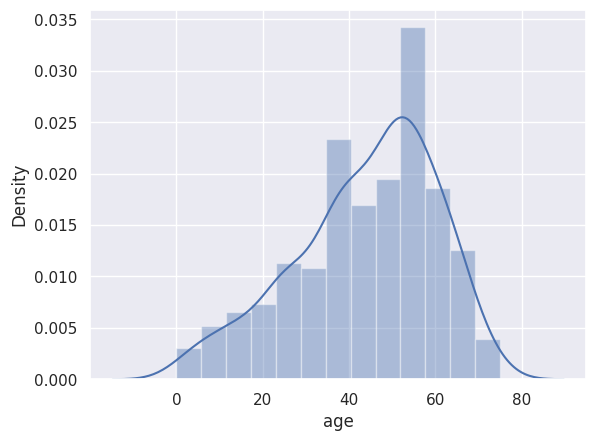
25% 0.000000 0.000000 0.000000

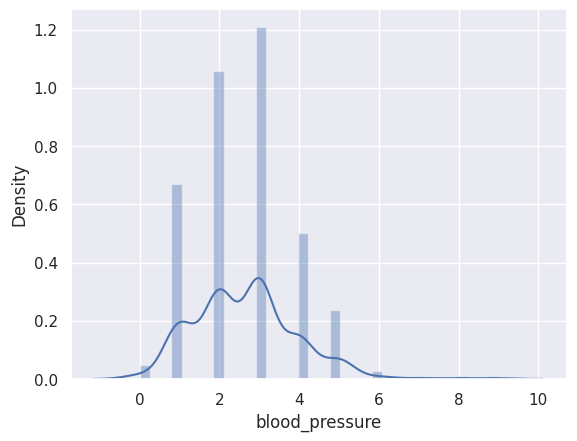
50% 0.000000 0.000000 0.000000

75% 0.000000 0.000000 1.000000

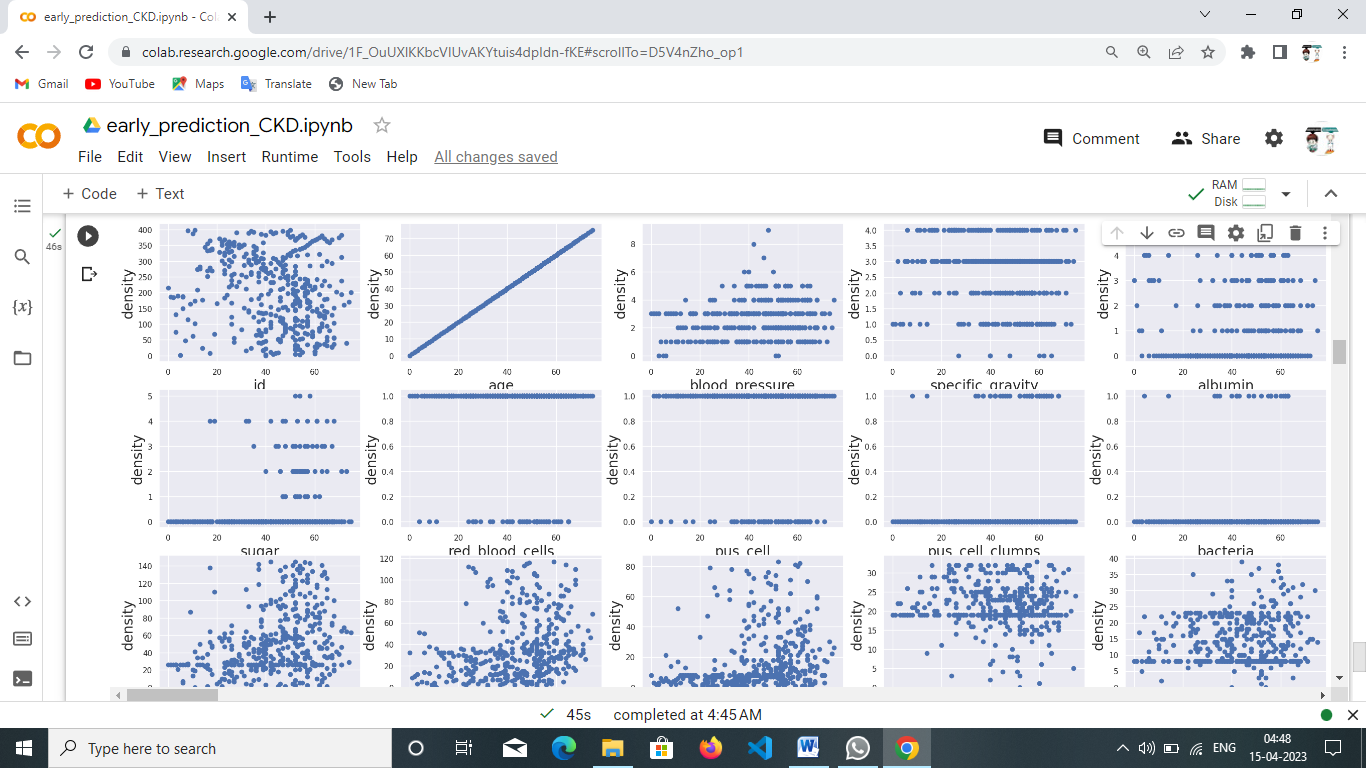
max 1.000000 1.000000 1.000000

[8 rows x 26 columns]









Axes(0.125,0.11;0.775x0.77)

Counter({0: 250, 1: 150})

Counter({0: 250, 1: 250})

(500, 24)

(500, 18)

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense (Dense) (None, 34) 646

dropout (Dropout) (None, 34) 0

dense\_1 (Dense) (None, 128) 4480

dropout\_1 (Dropout) (None, 128) 0

dense\_2 (Dense) (None, 64) 8256

dropout\_2 (Dropout) (None, 64) 0

dense\_3 (Dense) (None, 1) 65

=================================================================

Total params: 13,447

Trainable params: 13,447

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Epoch 1/100

13/13 [==============================] - 2s 24ms/step - loss: 0.7194 - accuracy: 0.5675 - val\_loss: 0.6035 - val\_accuracy: 0.9500

Epoch 2/100

13/13 [==============================] - 0s 8ms/step - loss: 0.6135 - accuracy: 0.6700 - val\_loss: 0.5171 - val\_accuracy: 0.9600

Epoch 3/100

13/13 [==============================] - 0s 8ms/step - loss: 0.5326 - accuracy: 0.7875 - val\_loss: 0.4244 - val\_accuracy: 0.9500

Epoch 4/100

13/13 [==============================] - 0s 7ms/step - loss: 0.4427 - accuracy: 0.8275 - val\_loss: 0.3284 - val\_accuracy: 0.9400

Epoch 5/100

13/13 [==============================] - 0s 7ms/step - loss: 0.3670 - accuracy: 0.8800 - val\_loss: 0.2390 - val\_accuracy: 0.9400

Epoch 6/100

13/13 [==============================] - 0s 6ms/step - loss: 0.2668 - accuracy: 0.9275 - val\_loss: 0.1671 - val\_accuracy: 0.9500

Epoch 7/100

13/13 [==============================] - 0s 8ms/step - loss: 0.2109 - accuracy: 0.9450 - val\_loss: 0.1229 - val\_accuracy: 0.9500

Epoch 8/100

13/13 [==============================] - 0s 9ms/step - loss: 0.1565 - accuracy: 0.9600 - val\_loss: 0.0936 - val\_accuracy: 0.9600

Epoch 9/100

13/13 [==============================] - 0s 7ms/step - loss: 0.1497 - accuracy: 0.9550 - val\_loss: 0.0695 - val\_accuracy: 0.9600

Epoch 10/100

13/13 [==============================] - 0s 9ms/step - loss: 0.1136 - accuracy: 0.9775 - val\_loss: 0.0508 - val\_accuracy: 0.9800

acy: 0.9975 - val\_loss: 0.0028 - val\_accuracy: 1.0000

Epoch 95/100

13/13 [==============================] - 0s 7ms/step - loss: 0.0028 - accuracy: 1.0000 - val\_loss: 0.0013 - val\_accuracy: 1.0000

Epoch 96/100

13/13 [==============================] - 0s 7ms/step - loss: 0.0010 - accuracy: 1.0000 - val\_loss: 0.0013 - val\_accuracy: 1.0000

Epoch 97/100

13/13 [==============================] - 0s 8ms/step - loss: 0.0011 - accuracy: 1.0000 - val\_loss: 0.0013 - val\_accuracy: 1.0000

Epoch 98/100

13/13 [==============================] - 0s 7ms/step - loss: 0.0027 - accuracy: 1.0000 - val\_loss: 0.0012 - val\_accuracy: 1.0000

Epoch 99/100

13/13 [==============================] - 0s 8ms/step - loss: 0.0031 - accuracy: 1.0000 - val\_loss: 0.0017 - val\_accuracy: 1.0000

Epoch 100/100

13/13 [==============================] - 0s 7ms/step - loss: 0.0021 - accuracy: 1.0000 - val\_loss: 0.0019 - val\_accuracy: 1.0000

<keras.callbacks.History object at 0x7fbd9c610790>

[1 1 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 1 0 1 0 0 0 1 0 0 1 1 0 0 1 1 1 1 1 0 0

0 1 1 0 0 0 0 1 0 1 1 1 1 0 1 0 0 1 1 0 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1 0 1

0 1 1 0 0 1 1 1 0 0 0 1 0 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1

0 1 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 1 1 1 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 0

0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 1 1 1 0 1 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1

0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 0 1 0 1 0 1 1 1

0 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1 1 1 1 0

1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 1 0 0 1 0 1 0 0 1 0

0 1 0 1 1 1 0 0 0 0 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 0 0 1

0 1 1 0 1 0 1 1 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0 0 1

0 1 1 1 0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 1 0 0]

[0 1 1 1 0 1 0 1 1 1 1 1 0 0 1 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0

1 0 1 0 0 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 1 1 1 1 0 1 1 1 0 0 0 1 0 0 1 0

1 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 0 1 0 1 1 0 1 1 0]

[1 1 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 1 0 1 0 0 0 1 0 0 1 1 0 0 1 1 1 1 1 0 0

0 1 1 0 0 0 0 1 0 1 1 1 1 0 1 0 0 1 1 0 1 0 1 0 1 1 0 0 0 1 1 0 0 0 1 0 1

0 1 1 0 0 1 1 1 0 0 0 1 0 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1

0 1 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 1 1 1 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 0

0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 1 1 1 0 1 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1

0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 0 1 0 1 0 1 1 1

0 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1 1 1 1 0

1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 1 0 0 1 0 1 0 0 1 0

0 1 0 1 1 1 0 0 0 0 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 0 0 1

0 1 1 0 1 0 1 1 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0 0 1

0 1 1 1 0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 1 0 0]

[0 1 1 1 0 1 0 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0

1 1 1 0 0 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 1 1 1 1 0 1 1 1 0 0 0 1 0 0 1 0

1 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 0 1 0 1 1 0 1 1 0]

[1]

[0]

[0]

4/4 [==============================] - 0s 4ms/step

[[False]

[ True]

[ True]

[ True]

[False]

[ True]

[False]

[ True]

[ True]

[ True]

[False]

[ True]

[ True]

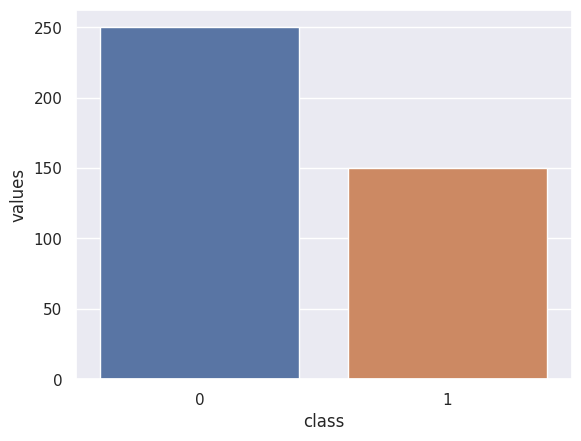
[False]

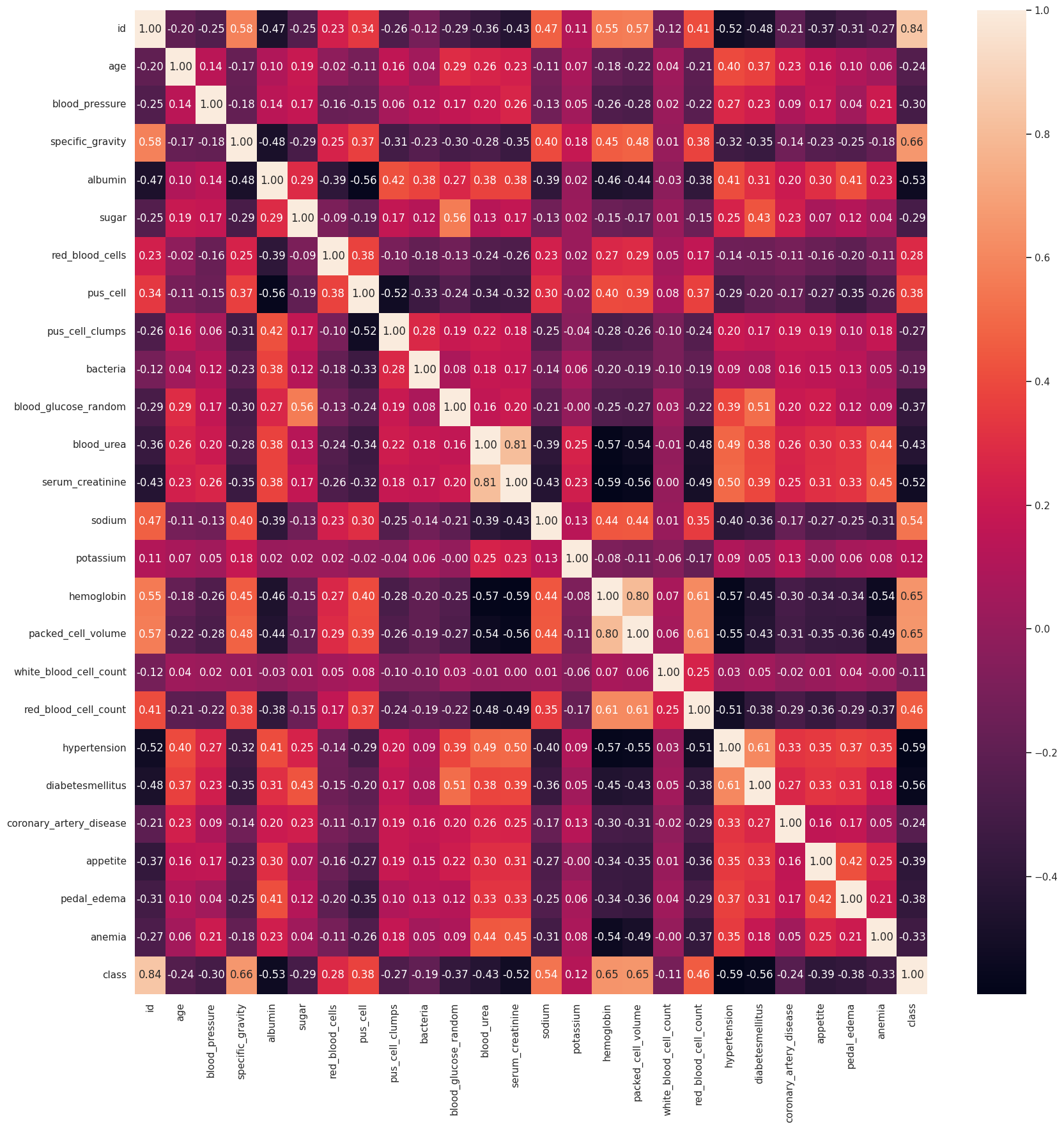
[ True]

[ True]

[False]]

1/1 [==============================] - 0s 52m step





Prediction: Low chance of CKD

LogReg

precision recall f1-score support

NO CKD 1.00 0.96 0.98 48

CKD 0.96 1.00 0.98 52

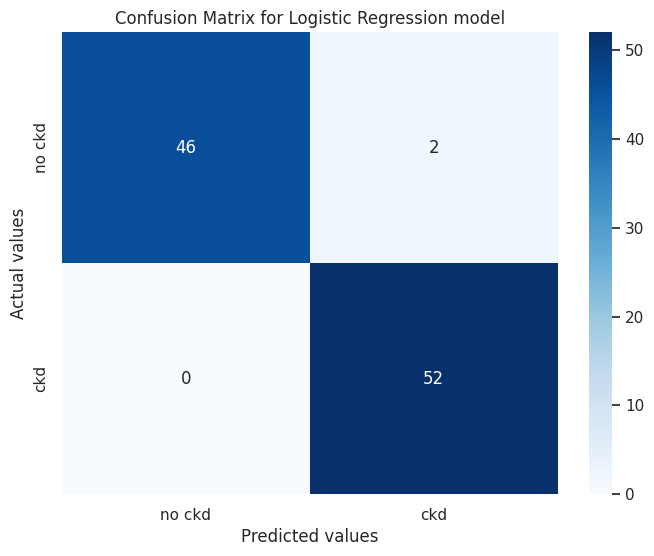
accuracy 0.98 100

macro avg 0.98 0.98 0.98 100

weighted avg 0.98 0.98 0.98 100

[[46 2]

[ 0 52]]



RF

precision recall f1-score support

NO CKD 1.00 1.00 1.00 48

CKD 1.00 1.00 1.00 52

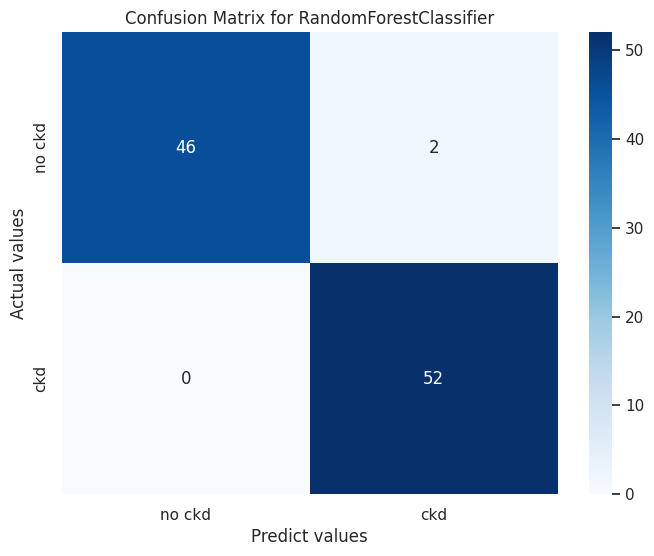
accuracy 1.00 100

macro avg 1.00 1.00 1.00 100

weighted avg 1.00 1.00 1.00 100

[[46 2]

[ 0 52]]



DecTr

precision recall f1-score support

NO CKD 1.00 1.00 1.00 48

CKD 1.00 1.00 1.00 52

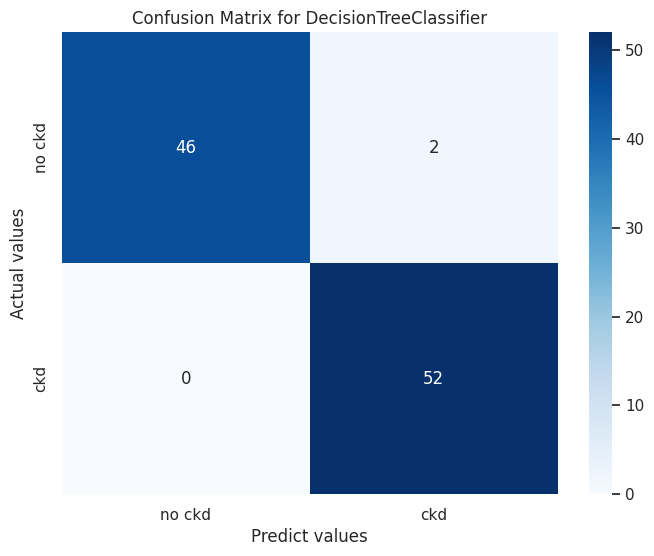
accuracy 1.00 100

macro avg 1.00 1.00 1.00 100

weighted avg 1.00 1.00 1.00 100

[[46 2]

[ 0 52]]



ANN

precision recall f1-score support

0 1.00 1.00 1.00 48

1 1.00 1.00 1.00 52

accuracy 1.00 100

macro avg 1.00 1.00 1.00 100

weighted avg 1.00 1.00 1.00 100

[[48 0]

[ 0 52]]

