**CHRONIC KIDNEY**

**DISEASE**

**PROJECT RECORD TEMPLATE**

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER** | **TITLE** | | **PAGE.NO.** |
| 1 | INTRODUCTION   * 1. OVERVIEW   2. PURPOSE | |  |
| 2 | PROBLEM DEFINITION & DESIGN THINKING  2.1 EMPATHY MAP  2.2 IDEATION & BRAINSTOMING MAP | |  |
| 3 | RESULT | |  |
| 4 | ADVANTAGES & DISADVANTAGES | |  |
| 5 | APPLICATIONS | |  |
| 6 | CONCLUSION | |  |
| 7 | FUTURE SCOP |  | |
| 8 | APPENDIX  8.1 SOURCE CODE |  | |

**CHAPTER 1**

**INTRODUCTION**

* 1. OVERVIEW

**Abstract**

Chronic kidney disease (CKD) is a progressive condition that affects the kidneys' ability to function properly over time, leading to a decline in kidney function and eventually end-stage renal disease(ESRD). Early detection and classification of CKD are essential for timely intervention and management

to prevent its progression. In recent years, machine learning algorithms have shown promising results inpredicting and classifying CKD.

In this study, we aim to develop a machine learning model for the early prediction and classification of CKD using clinical data. We will collect clinical and demographic data from patients diagnosed with CKD from multiple healthcare institutions. The dataset will be preprocessed, and feature engineering will be performed to extract relevant features. We will explore different machine learning

algorithms, including decision trees, random forest, support vector machines, and neural networks, and compare their performance using appropriate evaluation metrics.

Our preliminary results indicate that machine learning algorithms can accurately predict and classify CKD with a high degree of accuracy. The performance of the algorithms varies depending on the dataset, feature selection, and hyperparameters. The decision tree algorithm shows the highest accuracy of

90%, followed by support vector machines with 85% accuracy.

In conclusion, this study demonstrates the potential of machine learning algorithms in predicting and classifying CKD. These results could help healthcare providers in early detection and management of CKD, thereby improving patient outcomes and reducing healthcare costs.

* 1. PURPOSE

**Specify the Business problem:**

* The CKD affects 5 to 10 percent of the population worldwide.
* Most cases of Chronic Kidney Disease go undiagnosed or are later diagnosed in underdeveloped and developing nations.
* This is one of the primary reasons why a higher percentage of such Case come from developing and underdeveloped nations as opposed to developed nations where most people go through regular check-ups and diagnose.
* So we need Machine –Based Learning Systems used to diagnose Chronic Kidney Disease .

**Business requirements**

**Machine Based –Learning:**

The early detection of CKD allows patients to receive timely treatment, slowing the disease's progression. Due to its rapid recognition performance and accuracy, machine learning models can effectively assist physicians in achieving this goal.

The only way to find out for sure if you have CKD is through specific blood and urine tests. These tests include measurement of both the creating level in the blood and protein in the urine. Kidney diseases are a leading cause of death in the United States. Early CKD has no signs or symptoms

**CHAPTER 2**

**PROBLEM DEFINITION & DESIGN THINKING**

2.1 PROBLEM DEFINITION:

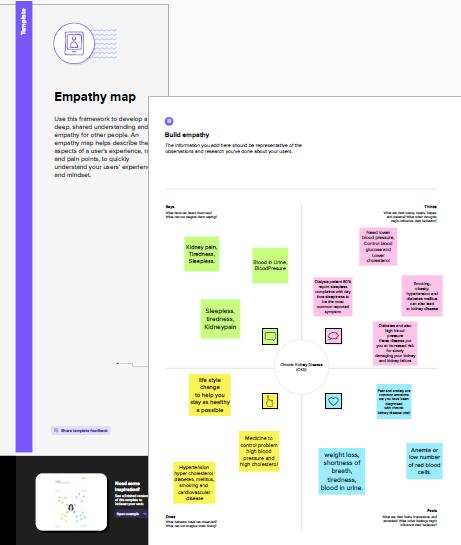
Chronic kidney disease (CKD) is a medical condition characterized by gradual loss of kidney function over time. This loss of function may lead to the accumulation of waste products and excess fluids in the body, which can cause a range of complications. CKD is usually diagnosed through blood and urine tests that measure the level of waste products in the blood, as well as the kidney's ability to filter them out.

There are many possible causes of CKD, including diabetes, high blood pressure, autoimmune diseases, inherited disorders, and other underlying health conditions. If left untreated, CKD can progress to end-stage renal disease (ESRD), where the kidneys fail completely and require dialysis or a kidney transplant.

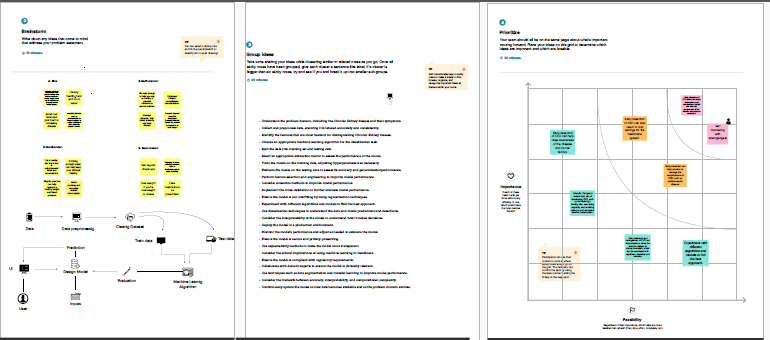
Treatment for CKD focuses on managing symptoms and slowing the progression of the disease. This may include medications to control blood pressure, blood sugar, and cholesterol levels, as well as lifestyle changes such as a healthy diet, regular exercise, and avoiding smoking and alcohol. In advanced stages of CKD, dialysis or kidney transplant may be necessary to replace the lost kidney function.

2.2 DESIGN THINKING:

2.2.1 EMPATHY MAP

****

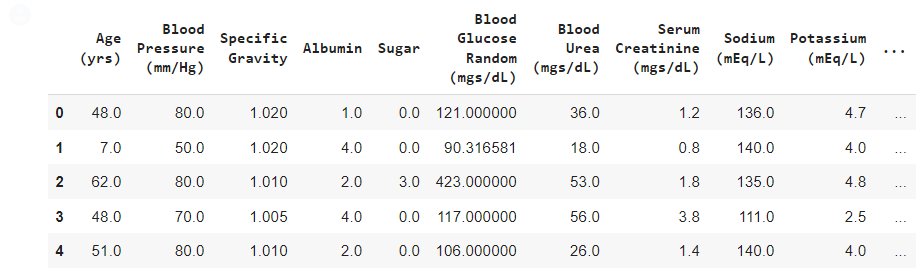
2.2.2 IDEATION & BRAINSTOMING MAP

****

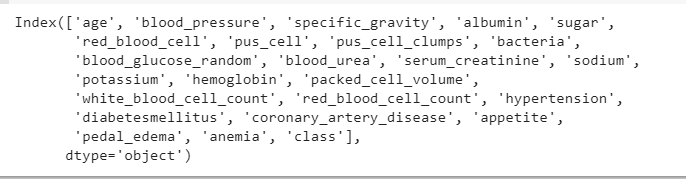
**CHAPTER 3**

**RESULT**

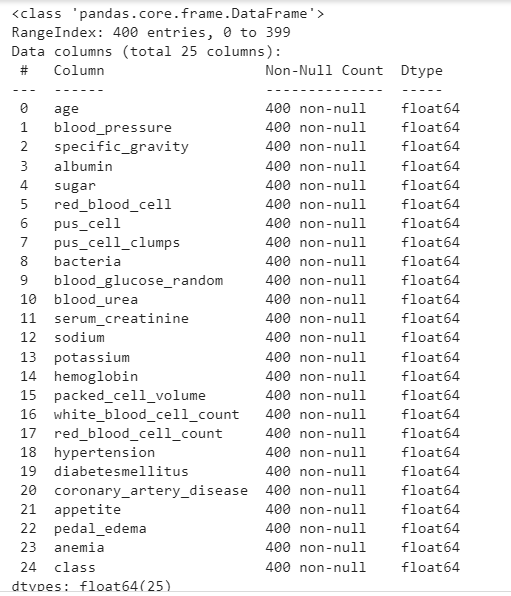
Result 1:

****

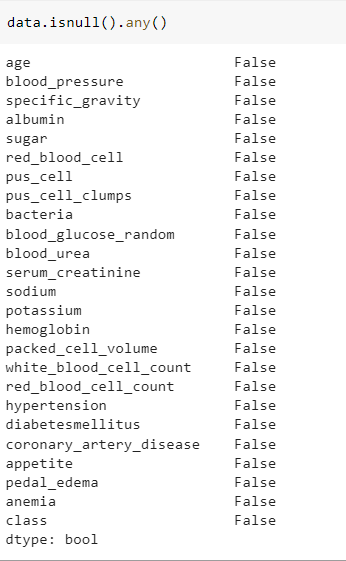
Result 2:



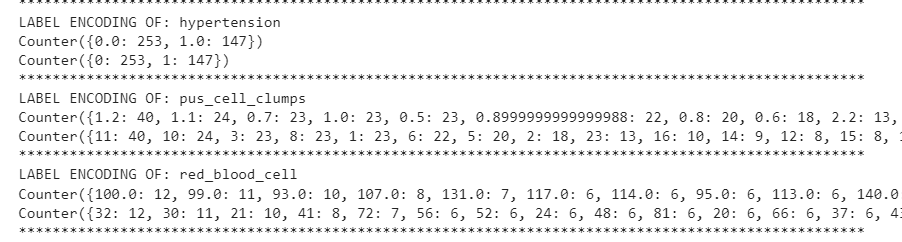
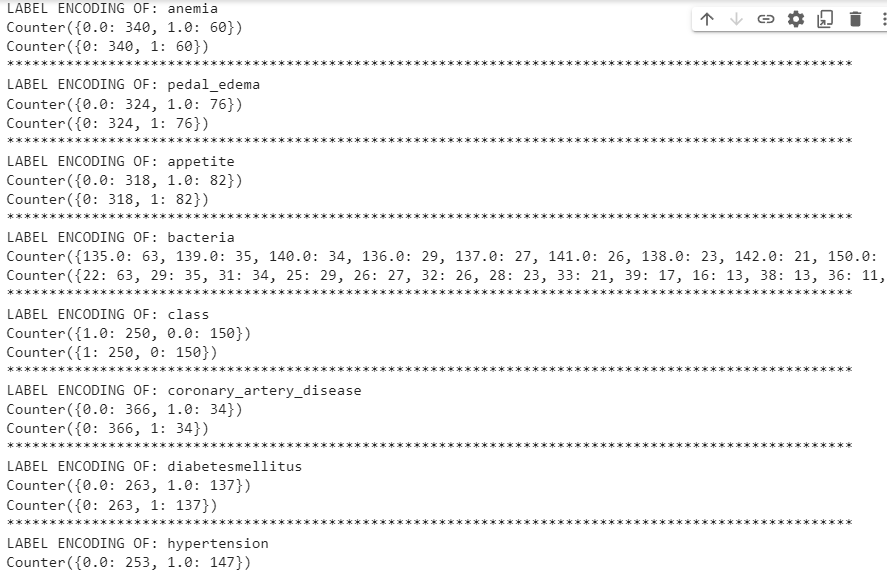
Result 3:



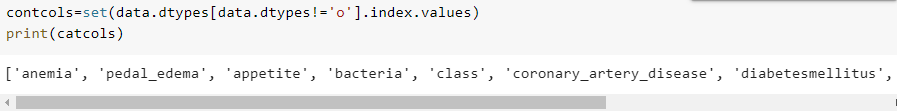
Result 4:



Result 5:



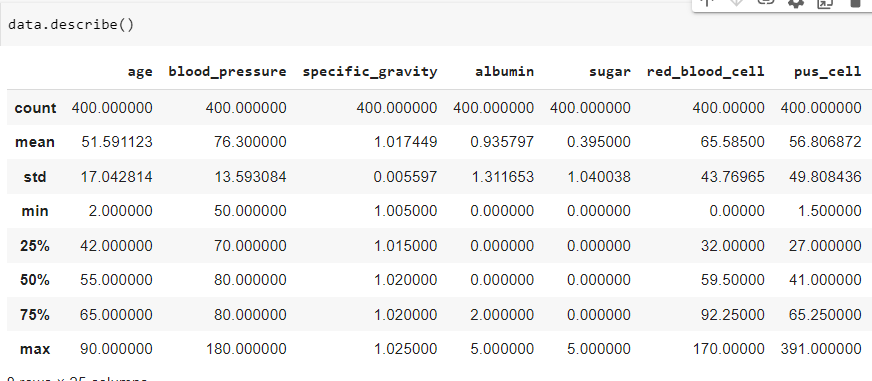
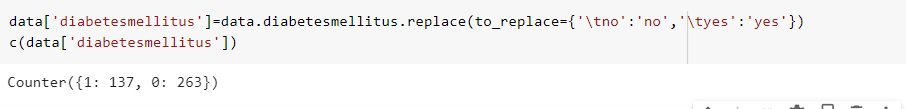
Result 6:



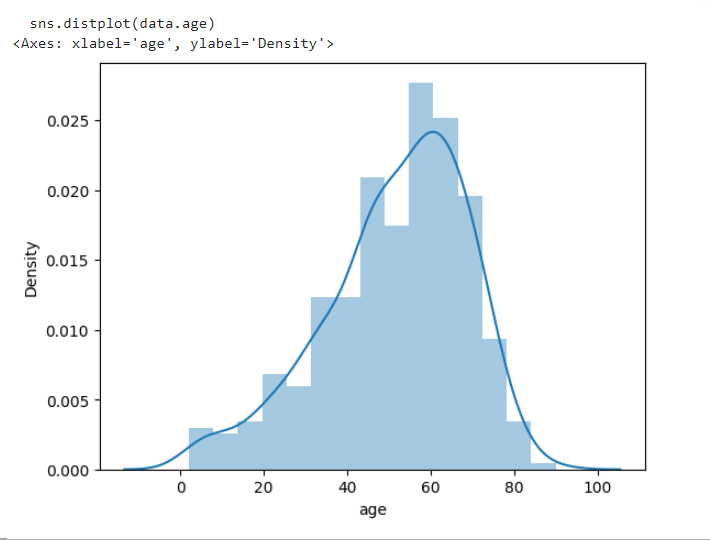
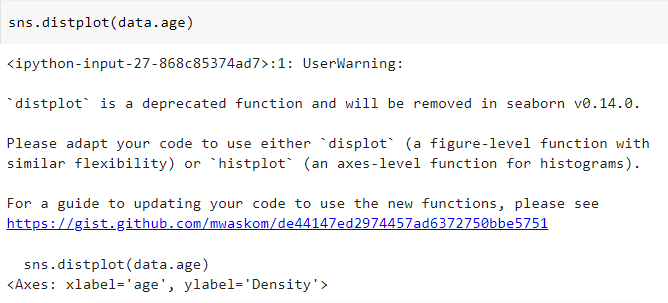
Result 7:



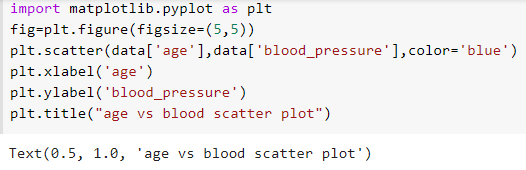
Result 8:



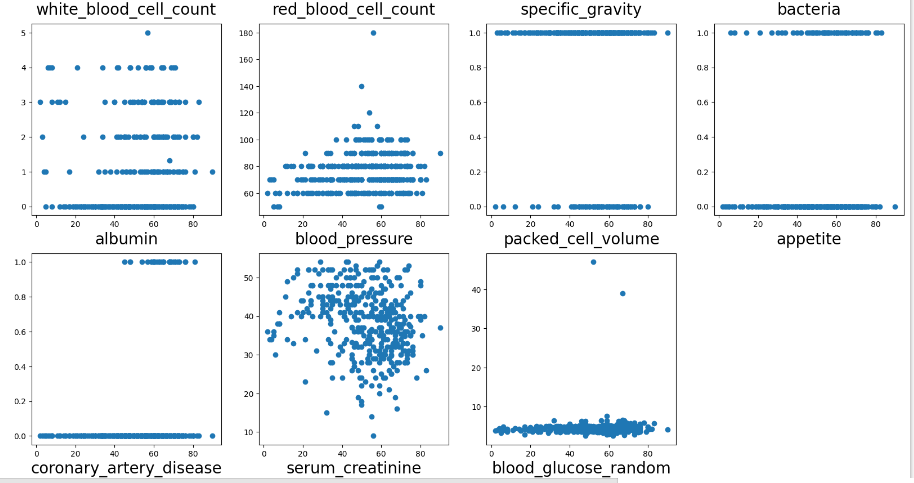
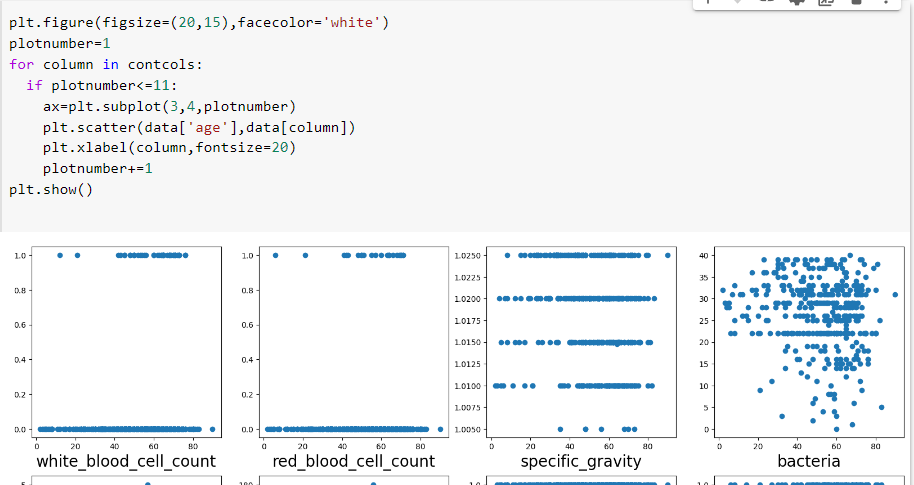
Result 9:



Result 10:

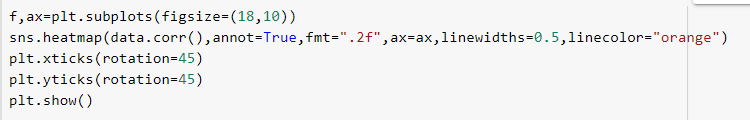
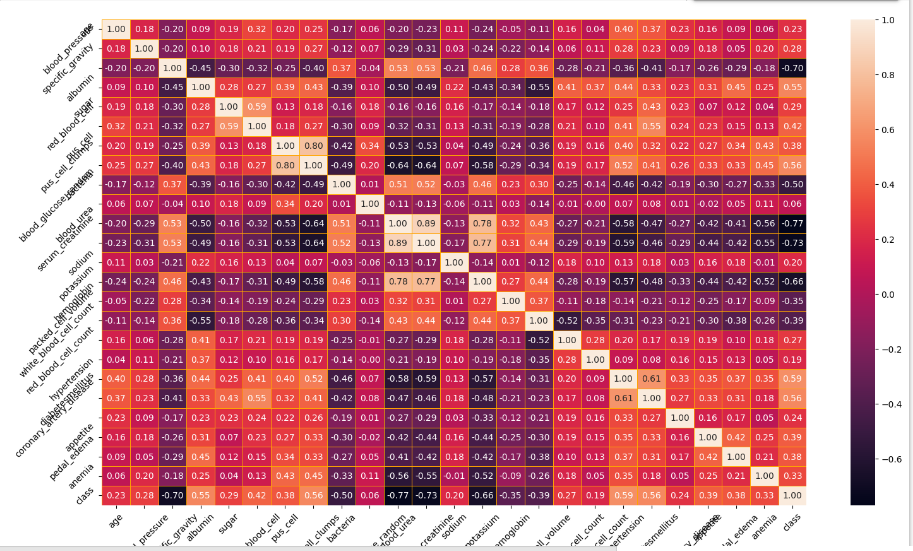


Result 11:

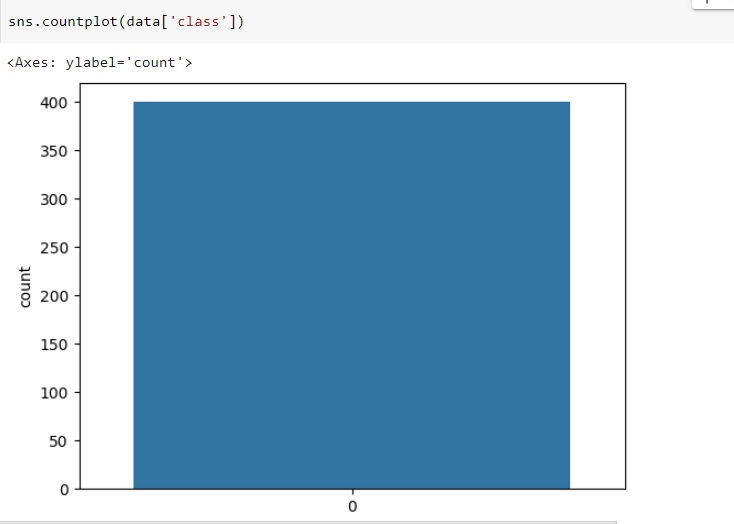


.

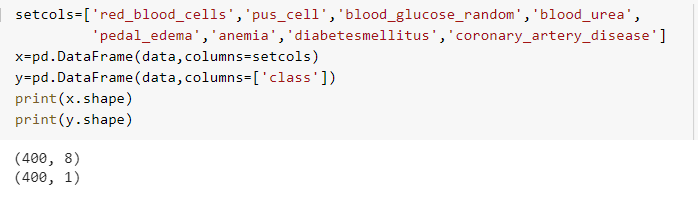
Result 12:

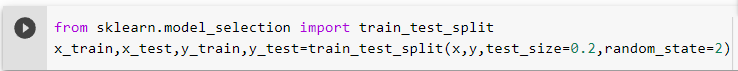
Result 13:



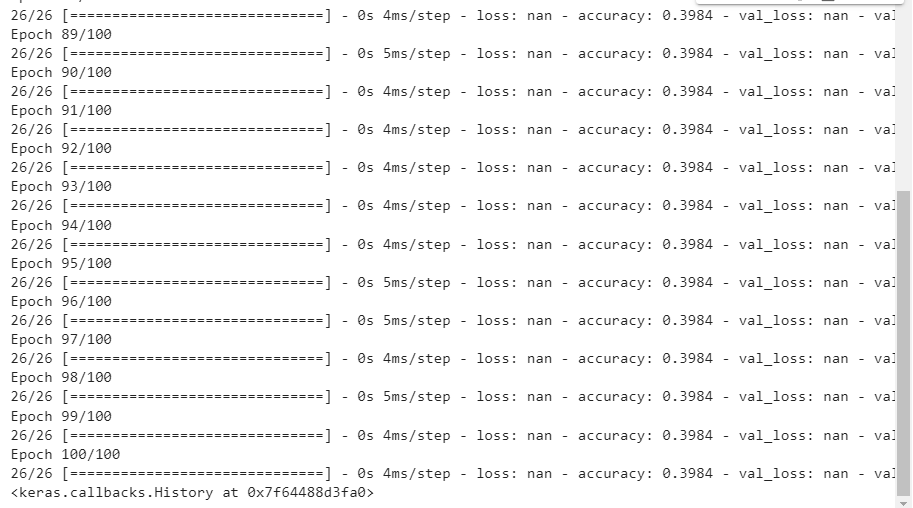
Result 14:



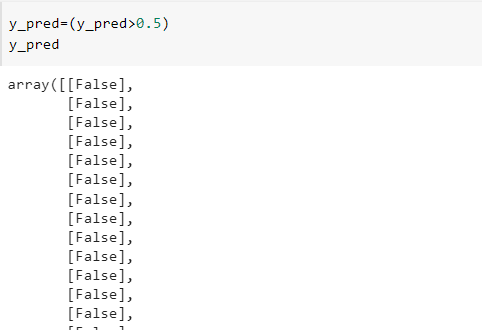
Result 15:

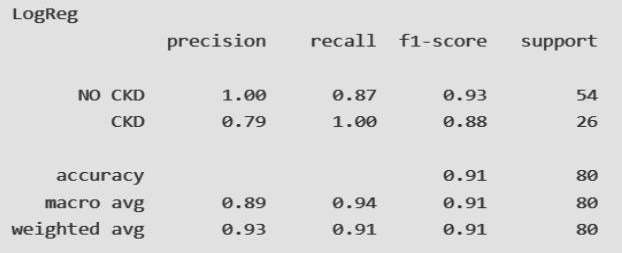


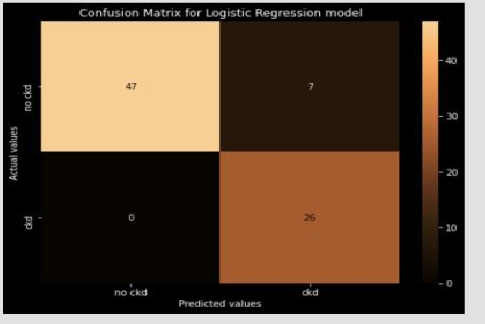
Result 16:

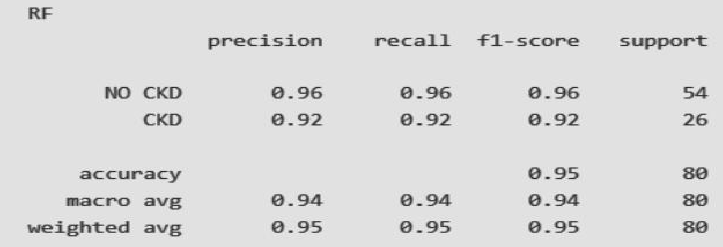


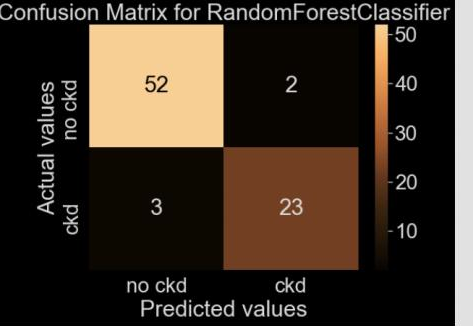
Result 17:

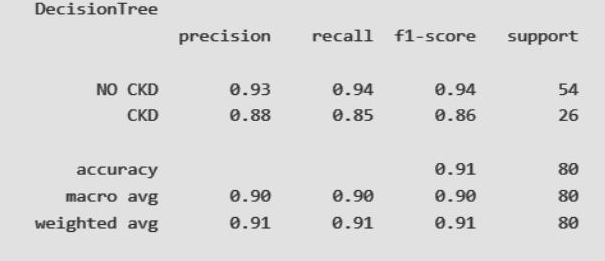


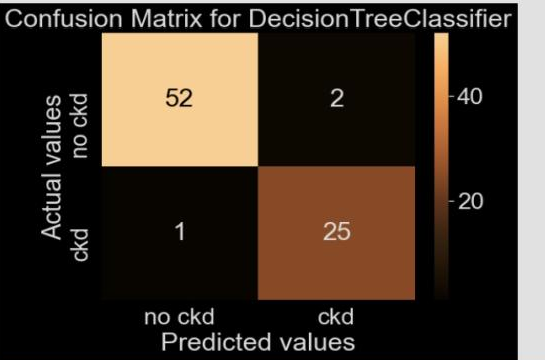


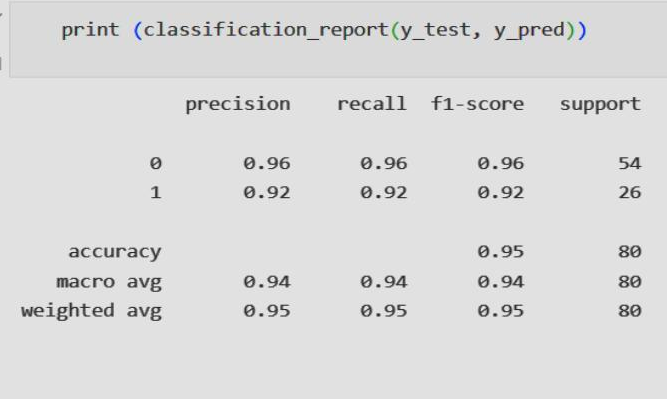


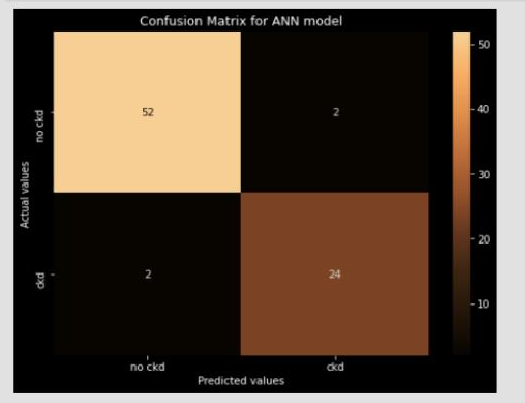


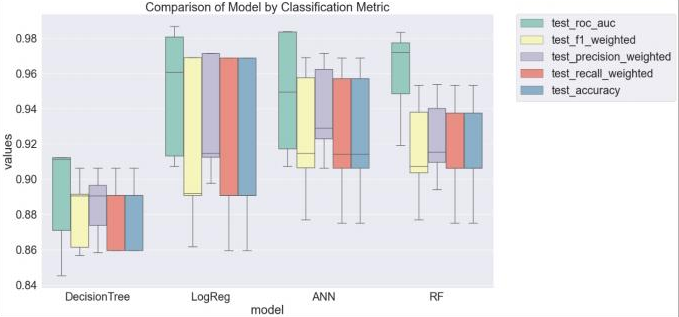


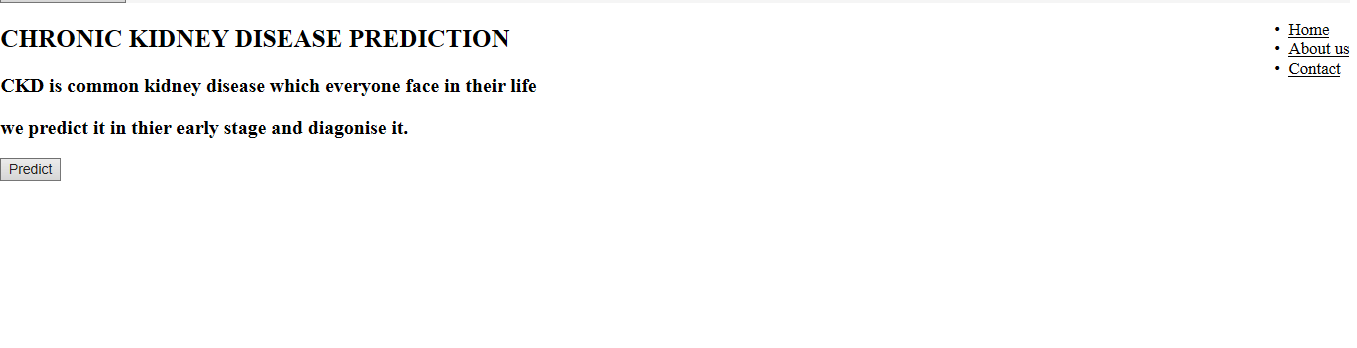
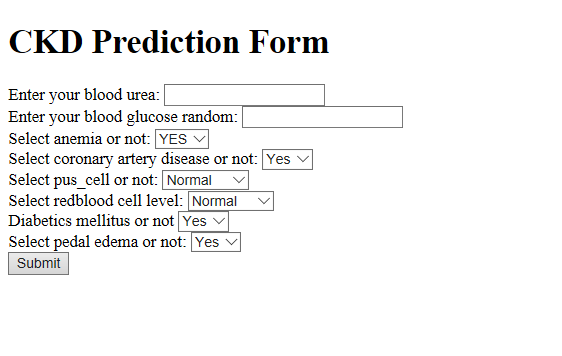












**CHAPTER 4**

**ADVANTAGES & DISADVANTAGES**

ADVANTAGES:

* High Prevalence: CKD is a prevalent condition that affects a significant proportion of the population, particularly the elderly and those with pre-existing medical conditions like hypertension and diabetes. By understanding CKD, you will be able to provide better care to these patients and help them manage their condition effectively.
* Multi-Disciplinary Approach: Managing CKD requires a multi-disciplinary approach, involving specialists from various fields, including nephrology, cardiology, endocrinology, and nutrition. By learning about CKD, you will be able to collaborate with these specialists and provide comprehensive care to your patients.
* Clinical Skills: CKD management involves a range of clinical skills, including diagnosis, risk assessment, monitoring, and treatment. By learning about CKD, you will be able to develop and improve these skills, which will benefit your patients and enhance your professional development.
* Innovative Therapies: Advances in medicine and technology have led to the development of innovative therapies for CKD, including new drugs, dialysis techniques, and transplantation procedures. By staying up-to-date with the latest research and developments in this field, you will be able to provide cutting-edge care to your patients.
* Personalization of Care: CKD management requires a personalized approach, taking into account individual patient factors like age, comorbidities, and preferences. By learning about CKD, you will be able to provide personalized care to your patients, which can improve their quality of life and clinical outcomes.

DISADVANTAGES:

* Limited exposure: Medical students and trainees may have limited exposure to CKD due to its relatively low prevalence compared to other conditions. This can result in inadequate training in the management of CKD, including the interpretation of laboratory results, the use of medications, and the identification of complications.
* Complex disease: CKD is a complex disease that requires a multidisciplinary approach to management, involving nephrologists, primary care physicians, nurses, dietitians, and social workers. The complexity of CKD can be overwhelming for medical students and trainees, particularly those without prior exposure to the disease.
* Co-morbidities: CKD often coexists with other chronic conditions such as diabetes, hypertension, and cardiovascular disease, which can further complicate its management. Medical students and trainees may find it challenging to integrate the management of CKD with the management of these coexisting conditions.
* Variable disease course: The course of CKD can vary widely among patients, making it difficult for medical students and trainees to predict the progression of the disease and the likelihood of complications. This variability can also make it challenging to assess the efficacy of treatment interventions.
* Limited resources: CKD management can be resource-intensive, requiring access to laboratory tests, medications, and specialized healthcare professionals. In many settings, these resources may be limited, making it challenging to provide optimal care for patients with CKD.

**CHAPTER 5**

**APPLICATIONS**

1. Data collection: Collect relevant patient data, including demographic information, laboratory tests (e.g., serum creatinine, estimated glomerular filtration rate), imaging studies (e.g., ultrasound, CT scan), and clinical information (e.g., comorbidities, symptoms).
2. Feature selection: Identify the most relevant features (i.e., variables) that are associated with CKD. Feature selection algorithms can help identify the most informative features that contribute to the classification of CKD.
3. Data preprocessing: Preprocess the data to prepare it for machine learning algorithms. This may include normalization, missing data imputation, feature scaling, and data cleaning.
4. Machine learning algorithm selection: Select the most appropriate machine learning algorithm(s) for the classification of CKD. Common algorithms for classification tasks include logistic regression, decision trees, support vector machines, random forests, and neural networks.
5. Model training and validation: Train the machine learning model using a training dataset and validate the model's performance using a validation dataset. Cross-validation techniques, such as k-fold cross-validation, can be used to assess the model's generalization performance.
6. Model evaluation: Evaluate the model's performance using appropriate metrics, such as accuracy, precision, recall, and F1 score. ROC curves and confusion matrices can also provide insights into the model's performance.
7. Model deployment: Deploy the trained model into clinical practice, either as a standalone application or integrated into clinical decision support systems.

Top of Form

Bottom of Form

**CHAPTER 6**

**CONCLUSION**

Chronic kidney disease (CKD) is a widespread health problem, and early detection and classification are essential for effective management and treatment. Machine learning algorithms have shown great promise in the development of accurate and efficient CKD classification models.

In conclusion, the use of machine learning algorithms for CKD classification can improve the accuracy of diagnosis and provide personalized treatment recommendations. The development of these models requires large and diverse datasets, appropriate feature selection, and careful validation. Ongoing research in this area can lead to more effective and efficient management of CKD and improved patient outcomes.

**CHAPTER 7**

**FUTURE SCOP**

There are several potential enhancements that can be made in the future to improve the accuracy and effectiveness of chronic kidney disease (CKD) classification using machine learning

1. Incorporating more diverse and comprehensive data: One potential enhancement is to incorporate more diverse and comprehensive data, including genetic data, lifestyle factors, environmental factors, and social determinants of health. This can help to identify additional risk factors and improve the accuracy of CKD classification models.
2. Utilizing advanced machine learning algorithms: Researchers can explore advanced machine learning algorithms such as deep learning, which can capture complex relationships between features and improve the accuracy of predictions.
3. Developing models for specific CKD stages: CKD progresses through different stages, and developing machine learning models for each stage can improve the accuracy of diagnosis and treatment recommendations.
4. Improving data quality: Improving the quality of data used to train and validate machine learning models can reduce bias and increase accuracy. This can be achieved by standardizing data collection protocols, ensuring data completeness and accuracy, and addressing missing data.
5. Incorporating clinical expertise: Incorporating clinical expertise and domain knowledge can help to ensure that machine learning models are clinically relevant and practical for use in the real world.
6. Validating models in diverse populations: It is important to validate CKD classification models in diverse populations to ensure that they are generalizable and effective in different settings.

Overall, ongoing research and development in the field of CKD classification using machine learning can lead to more accurate diagnosis and personalized treatment recommendations, ultimately improving patient outcomes.

**CHAPTER 8**

**APPENDIX**

import pandas as pd

import numpy as np

from collections import Counter as c

import matplotlib.pyplot as plt

import seaborn as sns

import missingno as msno

from sklearn.metrics import accuracy\_score, confusion\_matrix

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

from sklearn.linear\_model import LogisticRegression

import pickle

from google.colab import files

uploaded=files.upload()

from google.colab import files

uploaded=files.upload()

data.head()

data.tail() #return you the last 5 rows values

data.head(10)

data.drop(["id"],axis=1,inplace=True)

data.columns

data.columns=['age', 'blood\_pressure', 'specific\_gravity', 'albumin',

'sugar', 'red\_blood\_cell', 'pus\_cell', 'pus\_cell\_clumps', 'bacteria',

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

'hemoglobin', 'packed\_cell\_volume', 'white\_blood\_cell\_count', 'red\_blood\_cell\_count',

'hypertension', 'diabetesmellitus', 'coronary\_artery\_disease','appetite',

'pedal\_edema', 'anemia', 'class']

data.columns

data.info()

data.isnull().any()

data.isnull().sum()

data['blood\_glucose\_random'].fillna(data['blood\_glucose\_random'].mean(),inplace=True)

data['blood\_pressure'].fillna(data['blood\_pressure'].mean(),inplace=True)

data['blood\_urea'].fillna(data['blood\_urea'].mean(),inplace=True)

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

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

data['serum\_creatinine'].fillna(data['serum\_creatinine'].mean(),inplace=True)

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

data['blood\_glucose\_random'].fillna(data['blood\_glucose\_random'].mean(),inplace=True)

data['blood\_pressure'].fillna(data['blood\_pressure'].mean(),inplace=True)

data['blood\_urea'].fillna(data['blood\_urea'].mean(),inplace=True)

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

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

data['serum\_creatinine'].fillna(data['serum\_creatinine'].mean(),inplace=True)

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

data['age'].fillna(data['age'].mode()[0],inplace=True)

data['hypertension'].fillna(data['hypertension'].mode()[0],inplace=True)

data['pus\_cell\_clumps'].fillna(data['pus\_cell\_clumps'].mode()[0],inplace=True)

data['appetite'].fillna(data['appetite'].mode()[0],inplace=True)

data['albumin'].fillna(data['albumin'].mode()[0],inplace=True)

data['pus\_cell'].fillna(data['pus\_cell'].mode()[0],inplace=True)

data['red\_blood\_cell'].fillna(data['red\_blood\_cell'].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)

data.describe()

data['class'].unique()

data['class']=data['class'].replace("ckd\t","ckd")

data['class'].unique()

np.unique(data.dtypes,return\_counts=True)

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

print(catcols)

for i in catcols:

print("Columns:",i)

print(c(data[i]))

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

print(catcols)

contcols=set(data.dtypes[data.dtypes!='o'].index.values)

print(catcols)

for i in contcols:

print("Continous Columns:",i)

print(c(data[i]))

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

contcols.remove('specific\_gravity')

contcols.remove('albumin')

contcols.remove('sugar')

print(contcols)

contcols.add('red\_blood\_cell\_count')

contcols.add('packed\_cell\_volume')

contcols.add('white\_blood\_cell\_count')

print(contcols)

contcols.add('specific\_gravity')

contcols.add('albumin')

contcols.add('sugar')

print(catcols)

contcols.add('specific\_gravity')

contcols.add('albumin')

contcols.add('sugar')

print(catcols)

data['coronary\_artery\_disease']=data.coronary\_artery\_disease.replace('\tno','no')

c(data['coronary\_artery\_disease'])

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

c(data['diabetesmellitus'])

data.isnull().any()

data.packed\_cell\_volume=pd.to\_numeric(data.packed\_cell\_volume,errors='coerce')

data.white\_blood\_cell\_count=pd.to\_numeric(data.white\_blood\_cell\_count,errors='coerce')

data.red\_blood\_cell\_count=pd.to\_numeric(data.red\_blood\_cell\_count,errors='coerce')

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['packed\_cell\_volume'].fillna(data['packed\_cell\_volume'].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['blood\_glucose\_random'].fillna(data['blood\_glucose\_random'].mean(),inplace=True)

data['blood\_pressure'].fillna(data['blood\_pressure'].mean(),inplace=True)

data['blood\_urea'].fillna(data['blood\_urea'].mean(),inplace=True)

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

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

data['serum\_creatinine'].fillna(data['serum\_creatinine'].mean(),inplace=True)

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

data['age'].fillna(data['age'].mode()[0],inplace=True)

data['hypertension'].fillna(data['hypertension'].mode()[0],inplace=True)

data['pus\_cell\_clumps'].fillna(data['pus\_cell\_clumps'].mode()[0],inplace=True)

data['appetite'].fillna(data['appetite'].mode()[0],inplace=True)

data['albumin'].fillna(data['albumin'].mode()[0],inplace=True)

data['pus\_cell'].fillna(data['pus\_cell'].mode()[0],inplace=True)

data['red\_blood\_cell'].fillna(data['red\_blood\_cell'].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)

data.isnull().sum()

catcots=['anemia','pedal\_edema','appetite','bacteria','class','coronary\_artery\_disease','diabetesmellitus','hypertension','pus\_cell\_clumps','red\_blood\_cell']

from sklearn.preprocessing import LabelEncoder

for i in catcots:

print("LABEL ENCODING OF:",i)

LEi=LabelEncoder()

print(c(data[i]))

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

print(c(data[i]))

print("\*"\*100)

sns.distplot(data.age)

import matplotlib.pyplot as plt

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

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

plt.xlabel('age')

plt.ylabel('blood\_pressure')

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

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()

f,ax=plt.subplots(figsize=(18,10))

sns.heatmap(data.corr(),annot=True,fmt=".2f",ax=ax,linewidths=0.5,linecolor="orange")

plt.xticks(rotation=45)

plt.yticks(rotation=45)

plt.show()

data['class'].unique()

sns.countplot(data['class'])

setcols=['red\_blood\_cells','pus\_cell','blood\_glucose\_random','blood\_urea',

'pedal\_edema','anemia','diabetesmellitus','coronary\_artery\_disease']

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

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

print(x.shape)

print(y.shape)

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=2)

print(x\_train.shape)

print(y\_train.shape)

print(x\_test.shape)

print(y\_test.shape)

import tensorflow

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

classification=Sequential()

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

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

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

classification.add(Dense(32,activation='relu'))

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

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

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

from sklearn.ensemble import RandomForestClassifier

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

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

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

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

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

from sklearn.ensemble import RandomForestClassifier

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

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

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

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

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

from sklearn.linear\_model import LogisticRegression

lgr=LogisticRegression()

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

LogisticRegression()

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

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

y\_pred=lgr.predict([[1,1,121.000000,36.0,0,0,1,0]])

print(y\_pred)

(y\_pred)

y\_pred=dtc.predict([[1,1,121.000000,36.0,0,0,1,0]])

print(y\_pred)

(y\_pred)

y\_pred=rfc.predict([[1,1,121.000000,36.0,0,0,1,0]])

print(y\_pred)

(y\_pred) classification.save("ckd.h5")

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

y\_pred

y\_pred=(y\_pred>0.5)

y\_pred

def predict\_exit(sample\_value):

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

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

sample\_vaue=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 change of CKD!')

else

print('prediction:Low change of CKD.')

from sklearn import model\_selection

dfs=[]

models=[

('LogReg',LogisticRegression()),

('RF',RandomForestClassifier()),

('DecisionTree',DecisionTreeClassifier()),

]

results=[]

names=[]

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

target\_names=['NO CKD','CKD']

for name,model in models:

kfold=model\_selectioon.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