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

INTRODUCTION:

1.1 OVERVIEW

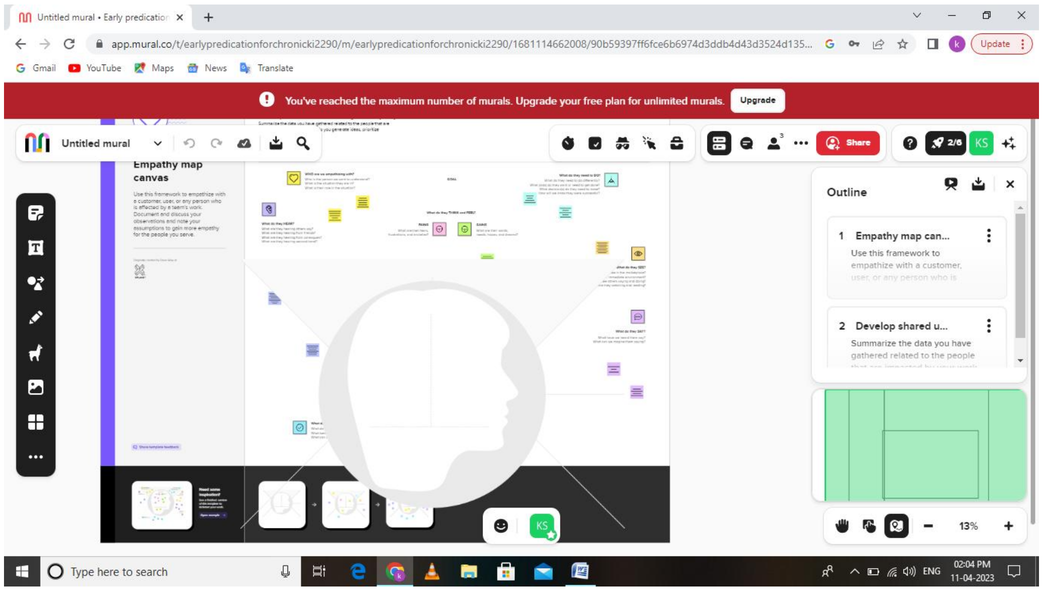
Chronic Kidney Disease is a serious lifelong condition that induced by either kidney pathology or reduced kidney functions. Early prediction and proper treatments can possibly stop, or slow the progression of this chronic disease to end-stage, where dialysis or kidney transplantation is the only way to save patient's life. In this study, we examine the ability of several machine-learning methods for early prediction of Chronic Kidney Disease. This matter has been studied widely; however, we are supporting our methodology by the use of predictive analytics, in which we examine the relationship in between data parameters as well as with the target class attribute. Predictive analytics enables us to introduce the optimal subset of parameters to feed machine learning to build a set of predictive models. This study starts with 24 parameters in addition to the class attribute and ends up by 30% of them as ideal subset to predict Chronic Kidney Disease. A total of 4 machine learning based classifiers have been evaluated within a supervised learning setting, achieving highest performance outcomes of AUC 0.995, sensitivity 0.9897, and specificity 1. The experimental procedure concludes that advances in machine learning, with assist of predictive analytics, represent a promising setting by which to recognize intelligent solutions, which in turn prove the ability of predication in the kidney disease domain and beyond.

1.2 PURPOSE

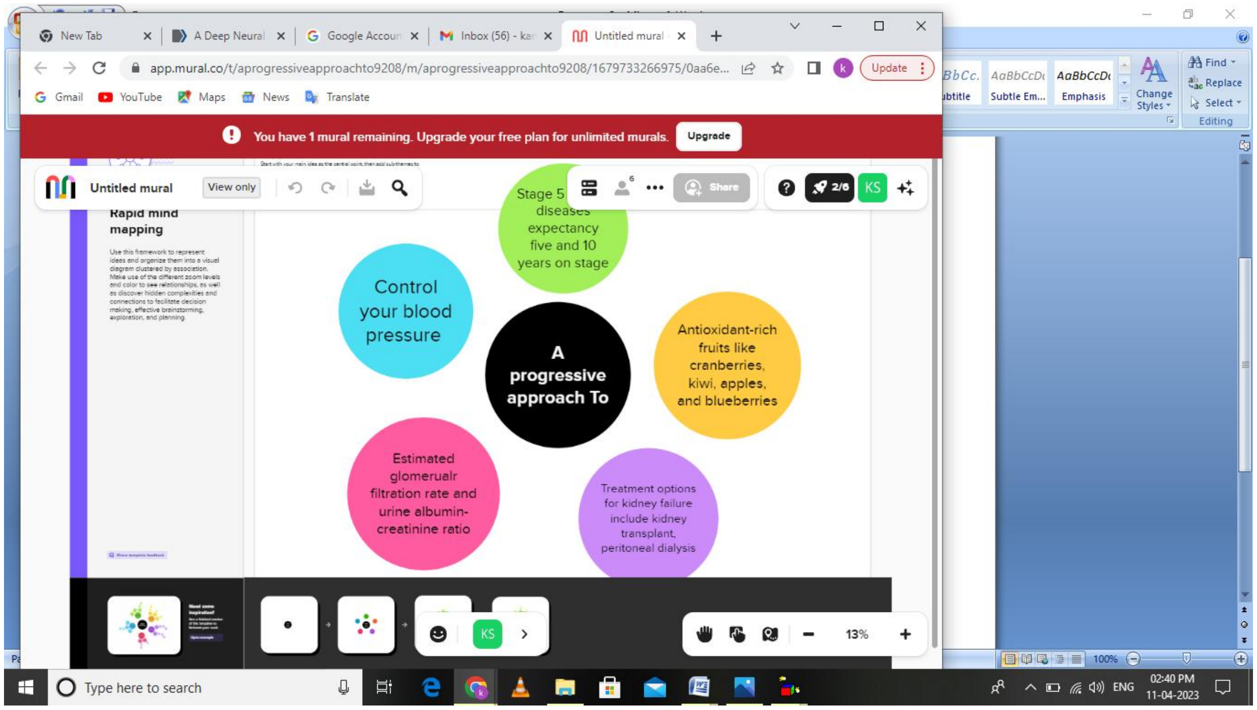
Chronic Kidney Disease (CKD) is a major medical problem and can be cured if treated in the early stages. Usually, people are not aware that medical tests we take for different purposes could contain valuable information concerning kidney diseases. Consequently, attributes of various medical tests are investigated to distinguish which attributes may contain helpful information about the disease. The information says that it helps us to measure the severity of the problem, the predicted survival of the patient after the illness, the pattern of the disease and work for curing the disease.In todays world as we know most of the people are facing so many disease and as this can be cured if we treat people in early stages this project can use a pretrained model to predict the Chronic Kidney Disease which can help in treatments of peoples who are suffer from this disease.

2. PROBLEM DEFINITION & DESIGN THINKING

2.1 EMPATHY MAP


Untitled mural • Early predication
app.mural.co/t/earlypredicationforchronicki2290/m/earlypredicationforchronicki2290/1681114662008/90b59397ff6fce6b6974d3ddb4d43d3524d135...
Gmail
YouTube
Maps
News
Translate
Update
You've reached the maximum number of murals. Upgrade your free plan for unlimited murals.
Upgrade
Untitled mural
Share
2/6
KS
Empathy map canvas
Use this framework to empathize with a customer, user, or any person who is affected by a team's work. Document and discuss your observations and note your assumptions to gain more empathy for the people you serve.
Outline
1 Empathy map can...
Use this framework to empathize with a customer, user, or any person who is
2 Develop shared u...
Summarize the data you have gathered related to the people
13%
Type here to search
ENG
02:04 PM
11-04-2023

2.2 IDEATION & BRAINSTORMING MAP

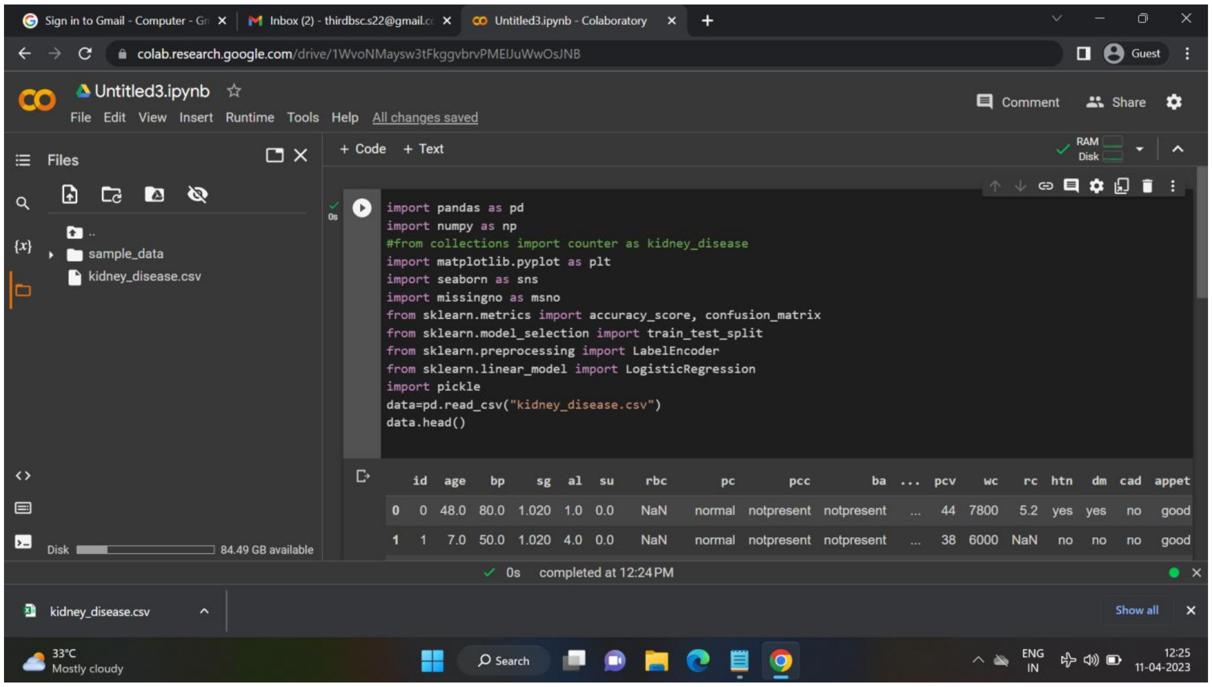


app.mural.co/t/aprogressiveapproachto9208/m/aprogressiveapproachto9208/1679733266975/0aa6e...
Untitled mural
View only
Rapid mind mapping
A progressive approach To
Stage 5 diseases expectancy five and 10 years on stage
Control your blood pressure
Antioxidant-rich fruits like cranberries, kiwi, apples, and blueberries
Estimated glomerualr filtration rate and urine albumin-creatinine ratio
Treatment options for kidney failure include kidney transplant, peritoneal dialysis
13%

You have 1 mural remaining. Upgrade your free plan for unlimited murals. Upgrade

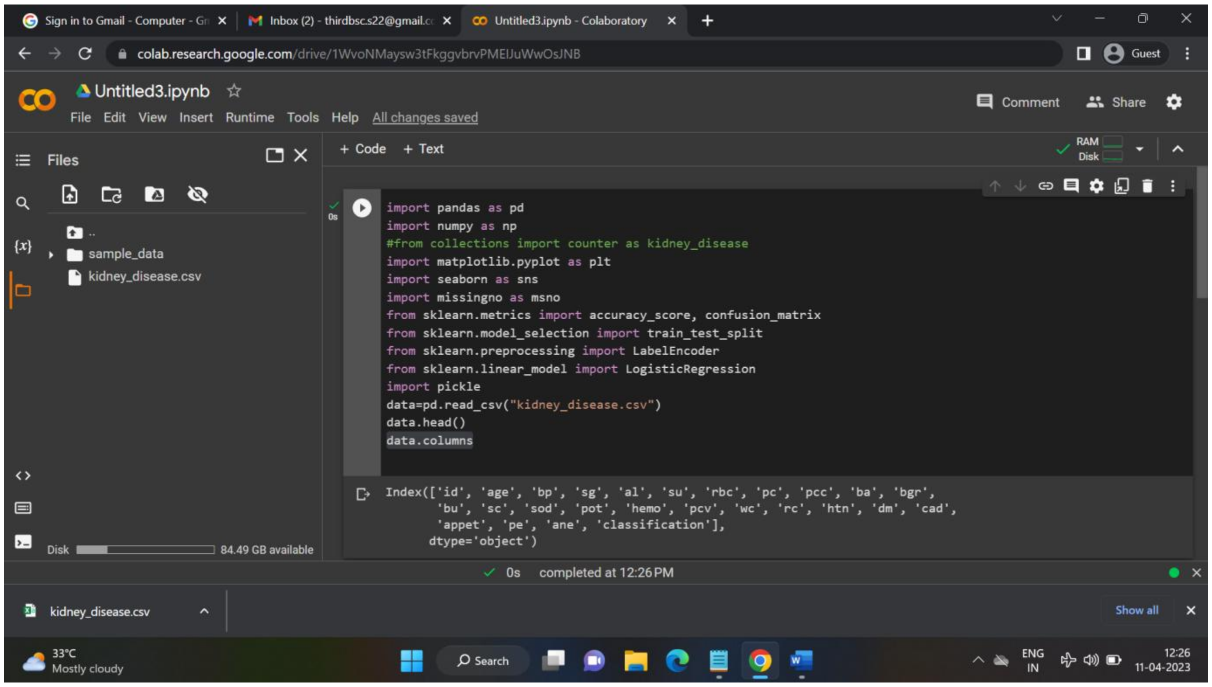
Use this framework to represent ideas and organize them into a visual diagram clustered by association. Make use of the different zoom levels and color to see relationships, as well as discover hidden complexities and connections to facilitate decision making, effective brainstorming, exploration, and planning.

3. RESULT:



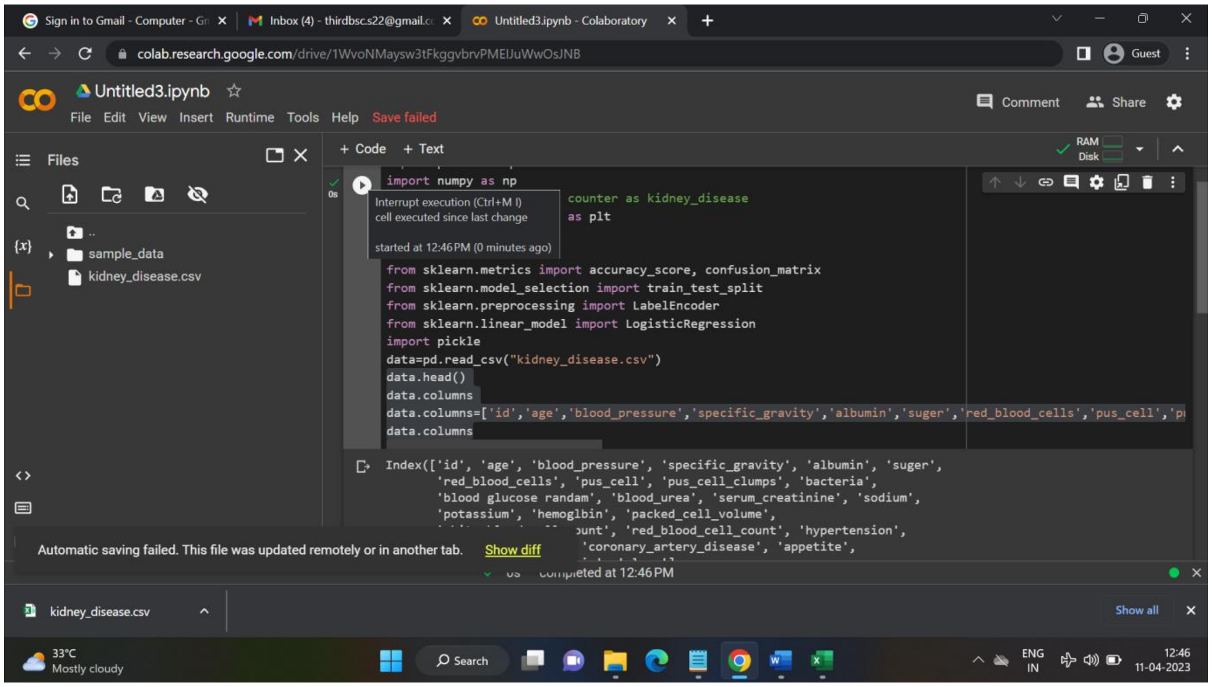
```
import pandas as pd
import numpy as np
#from collections import counter as kidney_disease
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
import pickle
data=pd.read_csv("kidney_disease.csv")
data.head()
```

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44	7800	5.2	yes	yes	no	good
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38	6000	NaN	no	no	no	good



```
import pandas as pd
import numpy as np
#from collections import counter as kidney_disease
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
import pickle
data=pd.read_csv("kidney_disease.csv")
data.head()
data.columns
```

```
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')
```

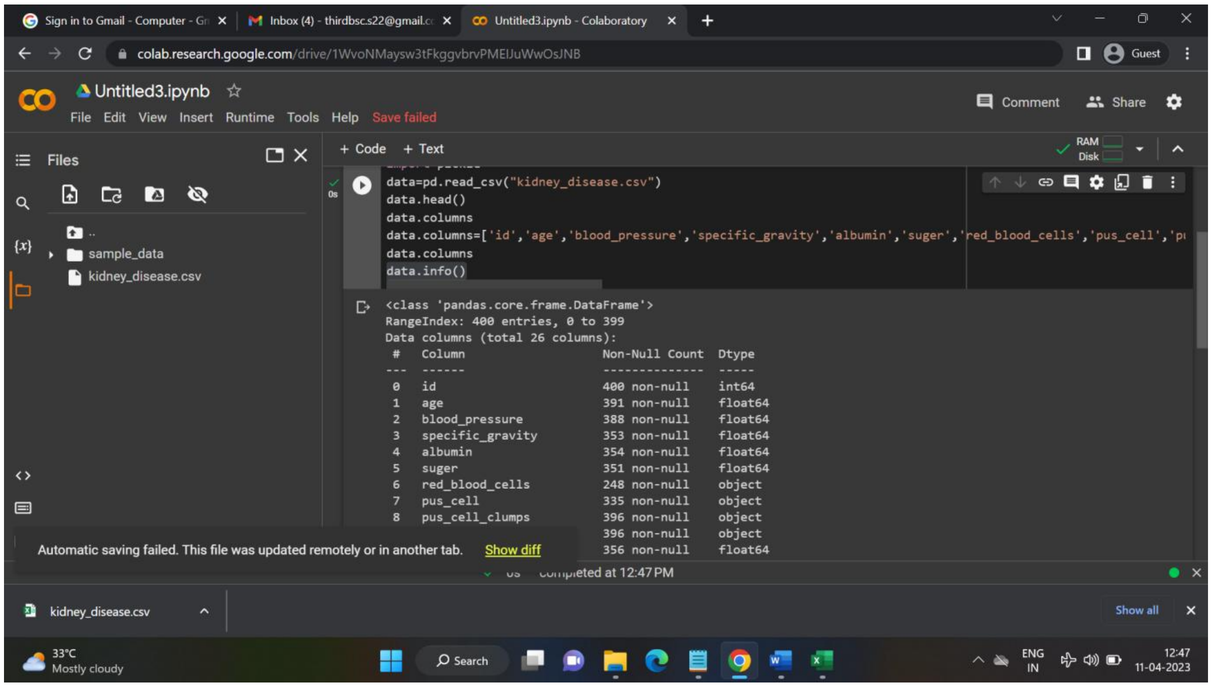


```
import numpy as np
                          counter as kidney_disease
                          as plt

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
import pickle
data=pd.read_csv("kidney_disease.csv")
data.head()
data.columns
data.columns=['id','age','blood_pressure','specific_gravity','albumin','suger','red_blood_cells','pus_cell','pu
data.columns
```

```
Index(['id', 'age', 'blood_pressure', 'specific_gravity', 'albumin', 'suger',
       'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria',
       'blood glucose randam', 'blood_urea', 'serum_creatinine', 'sodium',
       'potassium', 'hemoglbin', 'packed_cell_volume',
                        ount', 'red_blood_cell_count', 'hypertension',
                        'coronary_artery_disease', 'appetite',
```

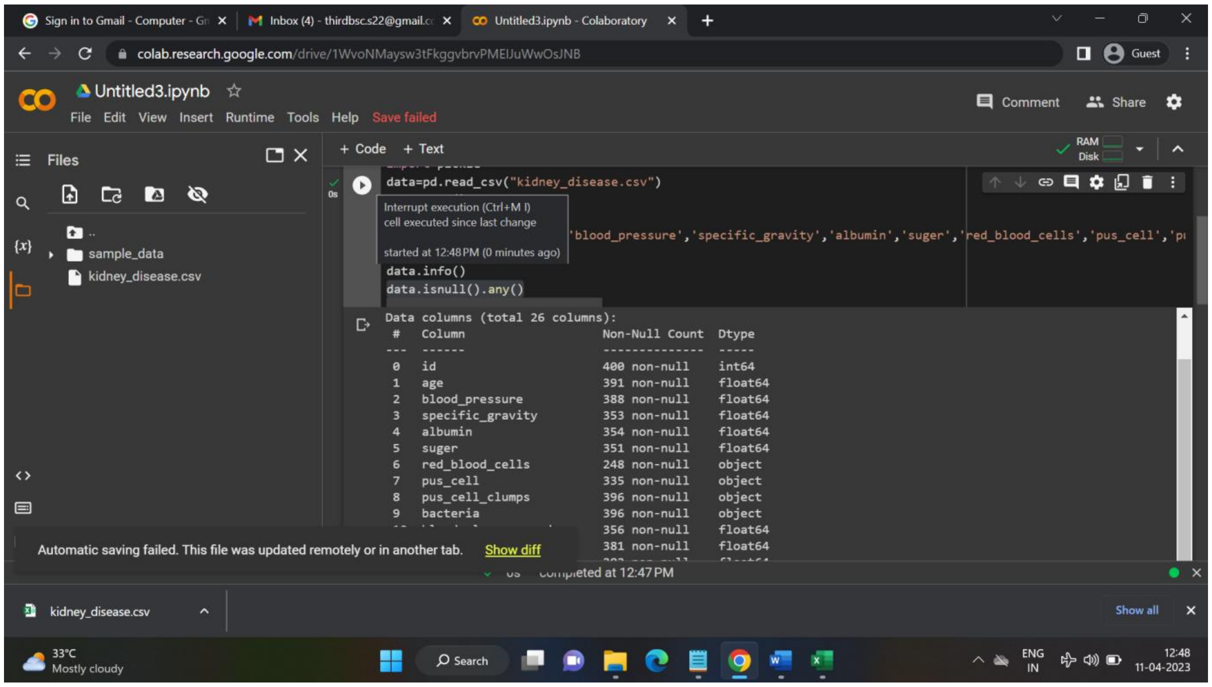
Automatic saving failed. This file was updated remotely or in another tab. Show diff



```
data=pd.read_csv("kidney_disease.csv")
data.head()
data.columns
data.columns=['id','age','blood_pressure','specific_gravity','albumin','suger','red_blood_cells','pus_cell','pu
data.columns
data.info()
```

```
<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   suger                    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
                              396 non-null    object
                              356 non-null    float64
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

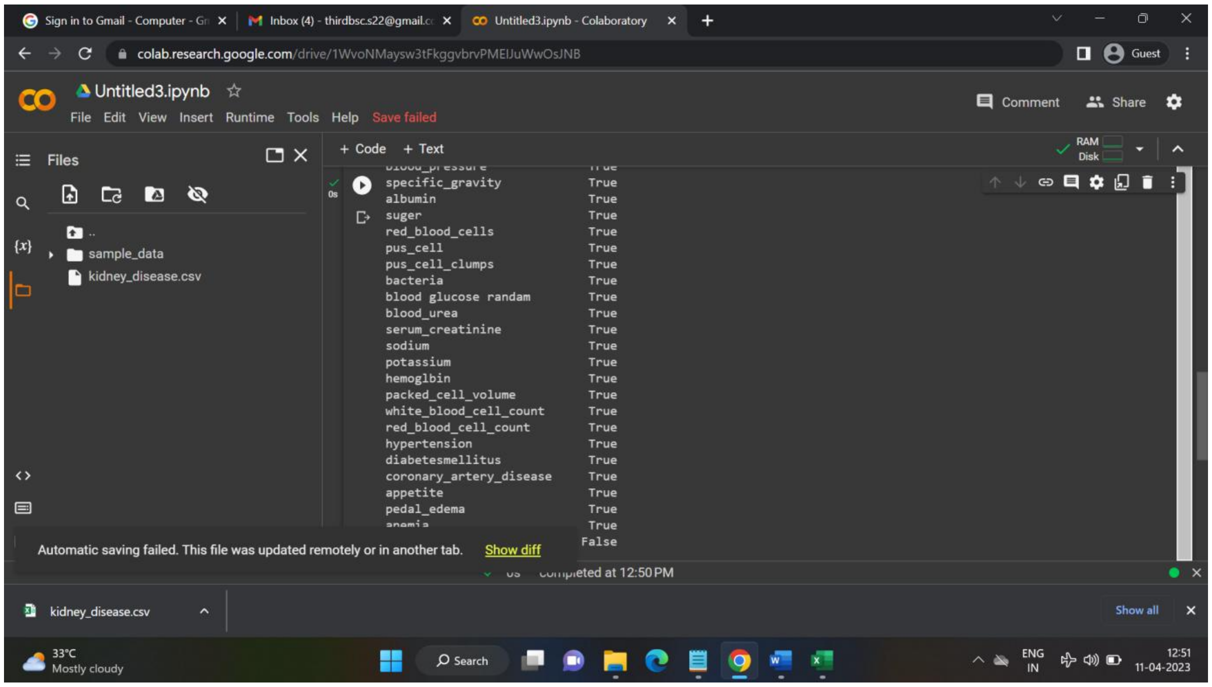


```
data=pd.read_csv("kidney_disease.csv")
'blood_pressure','specific_gravity','albumin','suger','red_blood_cells','pus_cell','pu
data.info()
data.isnull().any()
```

```
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   suger                    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
                              356 non-null    float64
                              381 non-null    float64
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

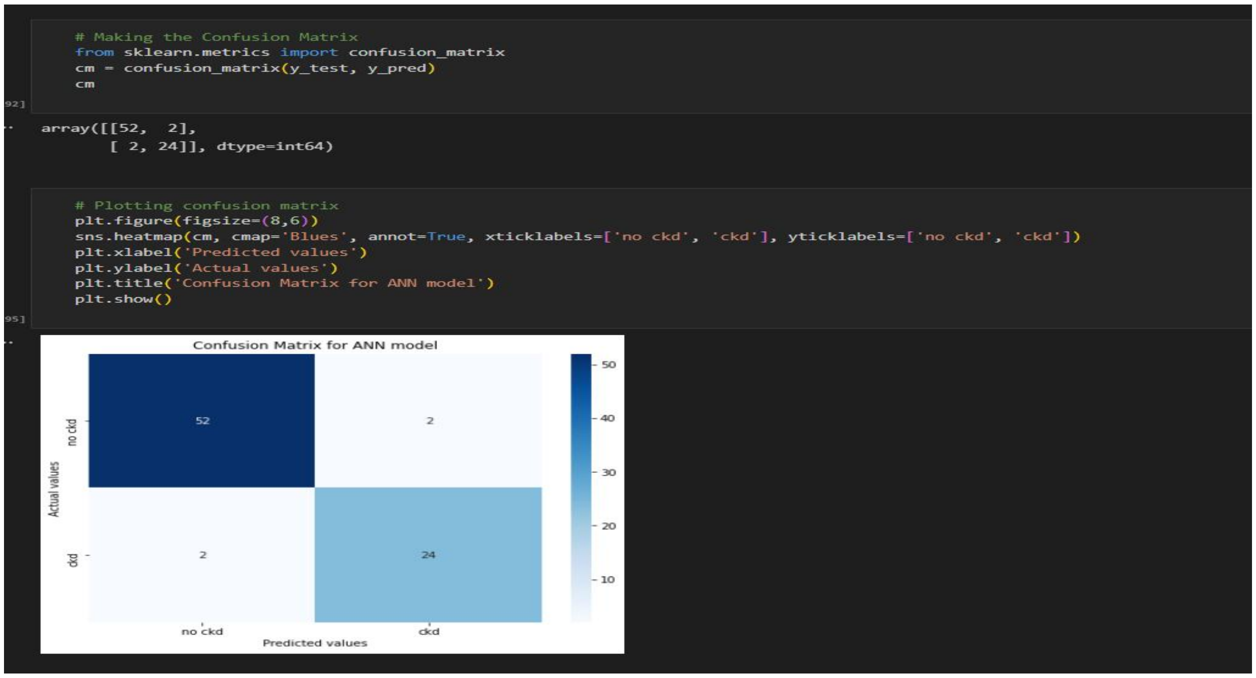
completed at 12:47 PM



```
specific_gravity         True
albumin                  True
suger                    True
red_blood_cells          True
pus_cell                 True
pus_cell_clumps          True
bacteria                 True
blood glucose randam     True
blood_urea               True
serum_creatinine         True
sodium                   True
potassium                True
hemoglbin                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
                        False
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

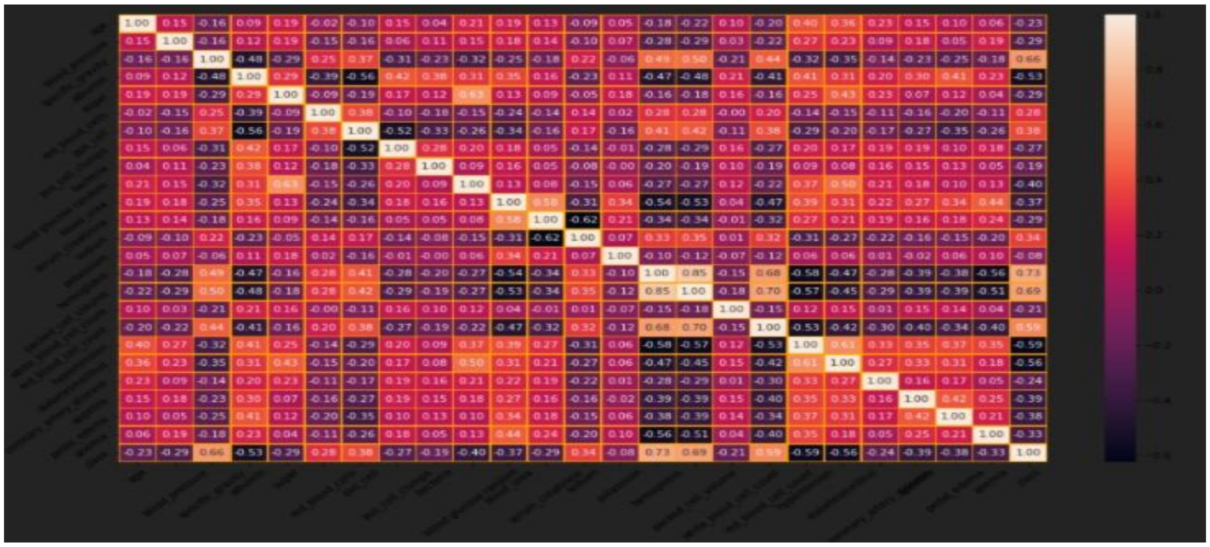
completed at 12:50 PM

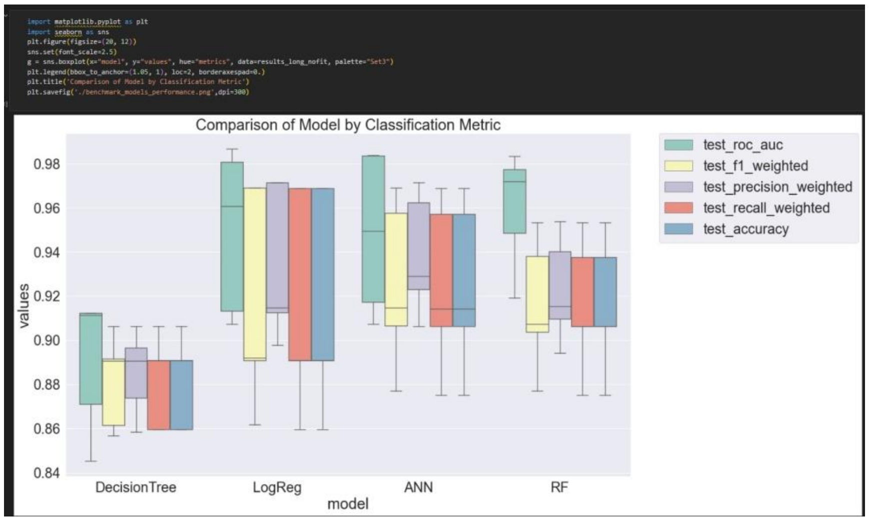


```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[52,  2],
       [ 2, 24]], dtype=int64)
```

```
# 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 ANN model')
plt.show()
```





```python
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(20, 12))
sns.set(font_scale=2.5)
g = sns.boxplot(x="model", y="values", hue="metrics", data=results_long_nofit, palette="Set3")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Comparison of Model by Classification Metric')
plt.savefig('./benchmark_models_performance.png',dpi=300)
```

4. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

Early detection of CKD allows proper management that could slow down CKD progression,prevent cardiovascular and other cormorbidities and enable timely initiation of dialysis

DISADVANTAGES:

A weakened immune system,which make it easier to developer infections.

Extra fluid in the body,which can use cause high blood pressure,swelling in the legs,or shortness of breath..

5. APPLICATIONS

While AI/ML is prevalent in our daily lives from finding optimal driving routes to facial recognition, uptake in nephrology research is slow. A search in PubMed for “machine learning” and “kidney” restricted to human studies resulted in only 207 results, half of which were published in the past two years and most do not comprise ML in its classical sense (Figure 1). Although, a comprehensive review of all articles is out of scope, we review pertinent recent literature on broader themes of applications of AI/ML in nephrology.

6. CONCLUSION

ML algorithms are a tool for unearthing the rules of big data, and prediction models which incorporate them have exceptional accuracy in predicting kidney disease patients' poor prognosis during clinical practice. The use of ML algorithms can help clinicians detect patients at high risk of kidney function progression in the early stages. In this way, they can receive treatment and management in time. In sum, we suggest the gradual incorporation of ML algorithm-based prediction models into clinical practice.

7. FUTURE SCOPE

8. APPENDEX:

```
import pandas as pd
import numpy as np
#from collections import counter as kidney_disease
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
import pickle
data=pd.read_csv("kidney_disease.csv")
```

```
data.head()
data.columns
data.columns=['id','age','blood_pressure','specific_gravity','albumin','su
ger','red_blood_cells','pus_cell','pus_cell_clumps','bacteria','blood
glucose
random','blood_urea','serum_creatinine','sodium','potassium','hemoglbin','
packed_cell_volume','white_blood_cell_count','red_blood_cell_count','hyper
tension','diabetesmellitus','coronary_artery_disease','appetite','pedal_ed
ema','anemia','class']
data.columns
data.info()
data.isnull().any()

data['bgr'].fillna(['bgr'].mean(),inplace=True)
data['bp'].fillna(data['bp'].mean(),inplace=True)
data['bu'].fillna(data['bu'].mean(),inplace=True)
data['hemo'].fillna(data['hemo'].mean(),inplace=True)
data['pcv'].fillna(data['pcv'].mean(),inplace=True)
data['pot'].fillna(data['pot'].mean(),inplace=True)
data['red_blood_cell_count'].fillna(data['red_blood_cell_count'].mean(),in
place=True)
data['sc'].fillna(data['sc'].mean(),inplace=True)
data['sod'].fillna(data['sod'].mean(),inplace=True)
data['white_blood_cell_count'].fillna(data['white_blood_cell_count'].mean()
,inplace=True)
data['age'].follna(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],inplce=Tr
ue)
data['appt'].fillna(data['appetite'].mode()[0],inplace=True)
data['albumin'].fillna(data['albumin'].mode()[0],inplace=True)
data['pus_cell'].fillna(data['pus_cell'].mode()[0],inplace=True)
data['red_blood_cells'].fillna(data['red_blood_cells'].mode()[0],inplace=T
rue)
data['coronary_artery_disease'].fillna(data['coronary_artery_disease'].mod
e[0],inplace=True)
```

```
data['anemia'].fillna(data['animea'].mode()[0],inplace=True)
data['su'].fillna(data['su'].mode()[0],inplace=True)
data['disabetesmellitus'].fillna(data['disabetesmellitus'].mode()[0],inpla
ce=True)
data['pedal_edema'].fillna(data['pedal_edema'].mode()[0],inplace=True)
data['specific_gravity'].fillna(data['specific_gravity'].mode()[0],inplace
=True)

catcols=set(data.dtypes[data.dtypes=='O'].index.values)
print(catcols)
for i in catcols:
print("Columns:",i)
print(c(data[i]))
print'*'*120+'\n')
catcols.remove('red_blood_cell_count')
catcols.remove('packed_cell_volume')
catcols.remove('white_blood_cell_count')
print(catcols)
catcols=['animia','pedal_edema','appetite','bacteria','class','coronary_ar
tery_disease','diabetesmellit','hypertension',pus_cell','pus_cell_clumps',
red_blood_cells']
from sklearn.preprocessing import LabelEncoder
for i in catcols:
print("LABEL ENCODING OF:",i)
LEi=LabelEncoder()
print(c(data[i]))
data[i]=LEi.fit_transform(data[i])
print(c(data[i]))
print("*"*100)
contcols=set(data.dtype[data.dtypes!='0'].index.values)
print(contcols)
for i in contcols:
print("Continous Columns:",i)
print(c(data[i]))
print('*'*120+'\n')
contcols.remove('specific_gravity')
contcols.remove('albumin')
contcols.remove('suger')
print(contcols)
contcols.add('red_blood_cell_count')
contcols.add('packed_cell_volume')
contcols.add('white_blood_cell_count')
print(contcols)
catcols.add('specific_gravity')
catcols.add('albumin')
catcols.add('suger')
print(catcols)
data['coronary_artery_disease']=data.coronary_artery_disease.replace('\tno
','no')
c(data['coronary_artery_disease'])
data.describe()
sns.distplot(data.age)
import matplotlib.pyplot as plt # import the matplotlib library
fig=plt.figure(figsize= (5,5)) #plot size
plt.scatter(data['age'],data['blood_pressure'],color='blue')
plt.xlabel('age') #set the label for x_axis
plt.ylabel('blood pressure') # set the label for y_axis
plt.tittle("age VS blood Scatter plot") #set a tittle for the axes
plt.figure(figsize=(20,15), facecolor='white')
plotnumber = 1
for column in contcols:
if plotnumber<=11
ax = plt.subplot(3,4,plotnumber)
plt.scatter(data['age'],data[column])
plt.xlabel(column,fontsize=20)
plotnumber+=1
plt.show()
f,ax=plt.subplots(figsize=(18,10))
sns.heatmap(data.corr(),annot=True,fmt=".2f",ax=ax,linewidths=0.5,lineco
lor="orange")
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.show()
sns.countplot(data['class'])
from sklearn.preprocessing import standardscaler
sc=standardscaler()
x_bal=sc.fit_transform(x)
selcols=['red_blood_cells',pus_cell','bloodglucose random','blood_urea',
'pedal_edema', 'anemia','diabetemellitus','coronary_artery_disease']
x=pd.DataFrame(data,column=selcols)
y=pd.DataFrame(data,columns=['class'])
print(x.shape)
print(y.shape)
from sklearn.model_selection import train_test_split
x_train,test,y_test=train_test_split(x,y,test_size=0.2,random_stream=2)
chronic kidney
import tensorflow
from tensorflow.keras.models import sequential
from tensorflow.keras.layers import Dense

classification = sequential()
classification.add(Dense(30,activation='relu'))
classification.add(Dense(128,activation='relu'))
classification.add(Dense(64,activation='relu'))
classification.add(Dense(32,activation='relu'))
classification.add(Dense(1,activation='sigmoid'))

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

classification.fit(x_train,y_train,batch_size=10,validation_split=0.2,epoc
hs=100)

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassification(n_estimators=10,criterion='entropy')


rfc.fit(x_train,y_train)
```

```
<ipython-input-255-b87bb2ba9825>:1: DataconversionWarning: A column-
vectory wa
(n_samples,), for exampleusing ravels
  rfc.fit (x _train,y_train)

RandomForestclassifier(citerion='entropy', n_estimators=10)

y_predict_train = rfc.predict(x_test)

y_predict_train = rfc.predict(x_train )
from sklearn.tree iport DecisionTreeclassifier

dtc =
DecisionTreeclassifier(max_depth=4,splitter='best',criterion='entropy')
dtc.fit(x_train,y_train)
decisionTreeclassifier(criterion='entropy', max_depth=4)
y_predict= dtc.predict(x_test)
y_predict

y_predict_train = dtc.predict(x_train)



from sklearn.linear_model import logisticRegression
lgr = LogesticRegression()
lgr.fit(x_train,y_train)
c:\user\saumya\anaconda3\lib\-packages\sklearn\utils\validation.py:72:
DataConversionwar
please change the shhape of y to (n_samples, ),for example using ravel().
return f(**kwargs)
LogesticRegression()

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)
[0]
array([0])
y_pred = dtc.predict([[1,1,121.000000,36.0,0,0,1,0]])
print(y_pred)
(y_pred)
[0]
array([0])
y_pred = rfc.predict([[1,1,121.000000,36.0,0,0,1,0]])
print(y_pred)
(y_pred)
[0]
array([0])
classification.save("ckd.hs")
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.resphape(1,-1)
 sample_value=sc.transform(sample-value)
test=classification.predict([[1,1,121.000000,36,0,0,0,1,0]])
if test==1:
print('prediction:High chance of CKD!')
else:
print('prediction:Low chance of CKD.')
prediction:Low chance of CKD.
from sklearn import model_selection

dfs=[]
models=[
        ('LonReg',LogistiucRegression()),
        ('RF',RandomForestClassifer()),
        ('DecisionTree',DecisionTreeClassifier()),
       ]
eresults = []
names = []
scoring = ['accuracy', 'precision_weighted', 'recall_weighed',
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-name))
results.append(cv_results)
names,append(name)
this_df = pd.DataFrame(cv_results)
this_df['model'] = name
dfs.append(this_df)
final = pd.concat(dfs,ignore_index=True)
return final
from sklearn.metrics import confusion_matrix
cm = confusuion_matrix(y_test, y_predict)
cm
array([[47, 7],
       [0,26]],dtype=int64)
plt.figure(figsize=(8,6))
sns.heatmap(cm, cmap='Blues', annot=Tree,xticlables=['no ckd',
'ckd'],yticklabls=['no ckd','ckd'])
plt.xlabel('Predicted values')
plt.ylable('Actual values')
plt.title('Confusion Matrix for Logistic Regression model')
plt.show()
formsklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_predict)
```

```
cm
array([[52, 2],
      [3,23]],dtype=int64)
plt.figure(figsize=(8,6))
sns.heatmap(cm,cmap='Blues',annot=True,xticklables=['no
ckd','ckd'],yticklables=['no ckd','ckd'])
plt.xlable(Predicted values')
plt.ylable('Actual values')
plt.title('confusion matrix for RandomForestClassifier')
plt.show()
python.txt
Displaying python.txt.
from sklearn.metrices import confusion_matrix
cm=confusion_matrix(y_test,y_predict)
cm
array([[52,2],
[1,25]],dtype=int64
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.tittle('Confusion Matrix for Decision ThreeClassifier')
plt.show()
print(classification_report(y_test,y_pred))
from sklearn.metrics import confusion_matrics
cm=confusion_matrix(y_test,y_pred)
cm
array([[52, 2],
[2, 24]],dtype=int64)
plt.figure(figsize=(8,6))
sns.heatmap(cm,cmap+'Blues',annot=True,xticlabels=['no
ckd','ckd'],yticklabels=['no ckd','ckd'])
plt.xlabel('Predicted values')
plt.ylabel('Actual values')
plt.tittle('Confusion Matrix for ANN model')
plt.show()
bootstraps=[]
for model in list(set(final.model.values)):
model_df=final.loc[final.model==model]
bootstraps.append(bootstrap)
bootstrap_df=pd.concat(bootstrap,ignore_index=True)
results_long=pd.melt(bootstrap_df,id_vars=['model'],var_name='metrices',va
lue_name='values')
time_metrices=['fit_time','score_time']
results_long_nofit=results_long.loc[~results_long['metrics'].isin(time_mat
rics)]
results_long_nofit=resultslong_nofit.sort_values(by='values')
results_long_nofit=results_long.loc[results_long['metrics'].isin(time_metr
ics)]
results_long_fit=results_long_fit.sort_values(by='values')
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(20,12))
sns.sert(font_scale=2.5)
g=sns.boxplot(x="model",y="values",hue"metrics",data=results_long_notfit,p
alette="Set3")
plt.legend(bbox_to_anchor=(1.05,1),loc=2,borderaxespad=0.)
plt.tittle('Coparison of_Model by Classification Metric')
plt.savefig('./benchmark_models_performance.png',dpi=300)
pickle.dump(lgr,open('CKD.pkl','wb'))
from flask import flask,render_template,request
import numpy as np
import pickle


app=Flask(__name__)
model=pickle.load(open('CKD'.pkl','rb')) #loading the model


@app.route('/')
def home():
```

```
return render_template('home.html')




@app.route('/prediction',methods=['POST','GET'])
def prediction():
return render _template('indexnew,html')
@app.riute('/Home',methods=['POST','GET'])
def my_home():
return render_template('home.html')
@app.route('/predict',methods=['POST"])# route to show thepredictions in a
web UI
def predict():

input_features=[float(x)for x in request.form.values()]
features_value=[np.array(input_features)]
features_name=['blood_urea','blood glucose random','anemia',
'coronary_artery_disease','pus_cell','red_blood_cells',
'diabetesmellitus','pedal_edema']
df=pd.DataFrame(features_value,columns=features_name)
output=model.predict(df)
return render_template('result.html',prediction_text=output)
if__name__=='__main__':
app.run(debug=True
(base D:\SmartBridge\Chronic Kidney Disease>python app.py
& Serving flask app "app"(lazy loading)
* Environment:production
WARNING:This is a development server.Do not use it in a producton
deployment.
Use a production WSGI server instead.
* Debug mode:off
* Running on http://127.0.0.1:5000/(Pres CTRL+C to quit)
```