EARLY PREDICTION FOR CHRONIC KIDNEY DISEASE DETECTION

TEAM SIZE : 4

TEAM LEADER : IMRAN A (20UCS4715)

TEAM MEMBERS : MURUGESAN P (2OUCS4724)

KARTHICK R (20UCS4717)

SIVAKARTHICKEYAN M (20UCS4731)

INTRODUCTION

Overview

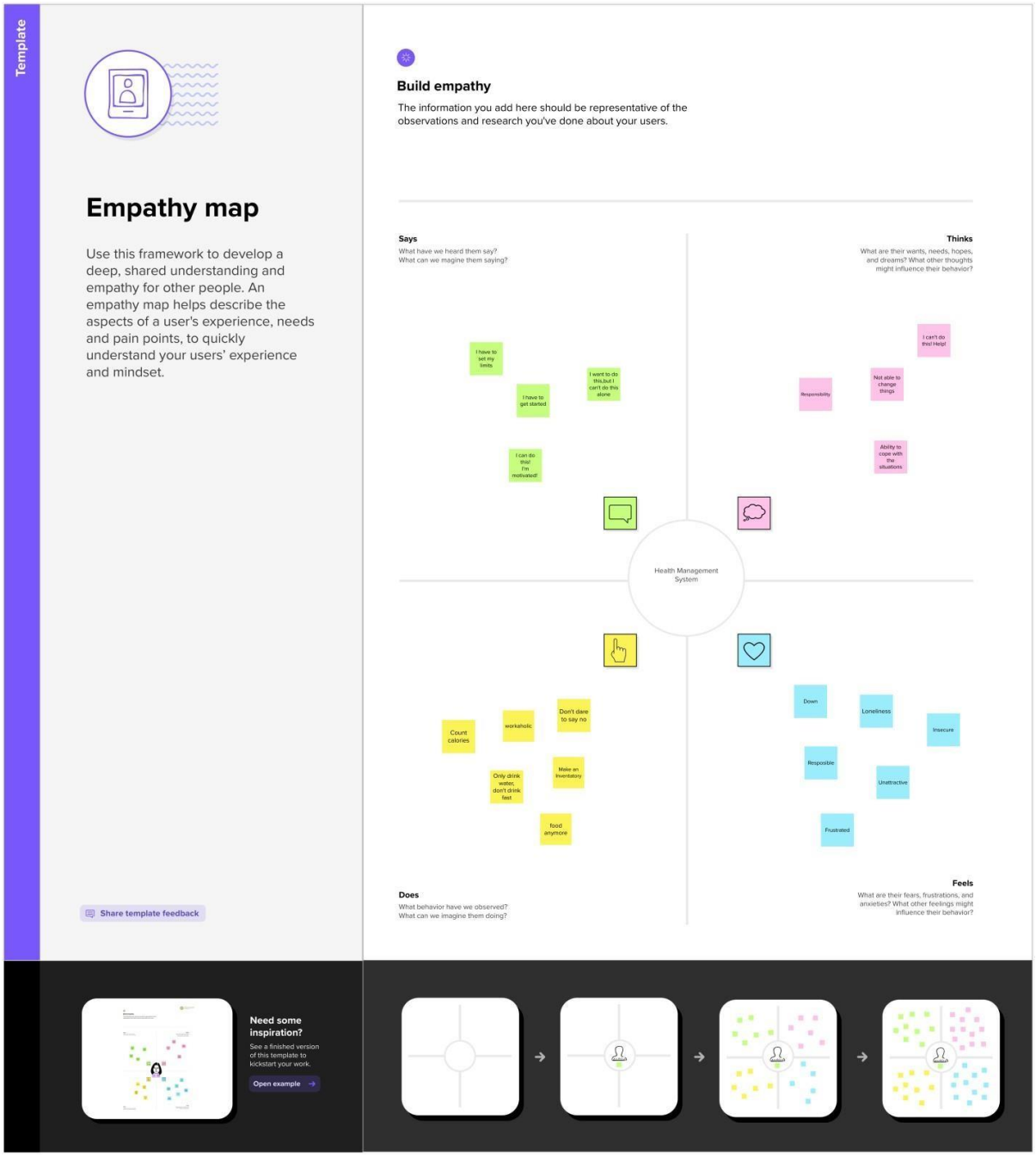
Every year, an increasing number of patients are diagnosed with late stages of renal disease. Chronic Kidney Disease, also known as Chronic Renal Disease, is characterized by abnormal kidney function or a break down of renal function that progresses over months or year. Kidney disease is often found during screening of persons who are known to be at risk for kidney issues, such as those with high blood pressure or diabetes, and those with a blood family who has chronic kidney disease (CKD).he primary goals of this research are to design and suggest a machine learning method for predicting CKD. Support Vector Machine (SVR), Random Forest (LR), Artificial Neural Network (ANN), and Decision Tree are four master teaching methodologies investigated (DT). The components are built using chronic kidney disease datasets, and the outcomes of these models are compared to select the optimal model for prediction.

Purpose

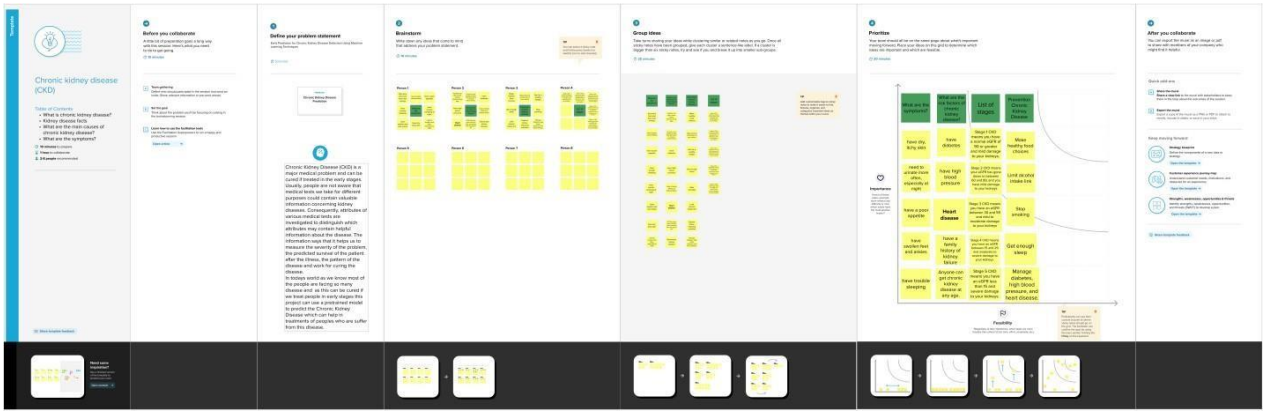
Kidney diseases (acute and chronic) are an important public health problem, and chronic kidney disease (CKD) is a noncommunicable disease of global significance. CKD is defined by the presence of kidney damage or reduced kidney function for a period of at least 3 months, with implications for health. The level of disease severity has been used to classify CKD into various stages, from persistent kidney damage only with preserved kidney function (estimated glomerular filtration rate (eGFR) >90ml/min/1.73m^2; stage 1) to persistent kidney damage accompanied by mild reduction in kidney function (eGFR 60-90; stage 2) to moderate to severe reduction in kidney function (eGFR 30-60; stage 3, and eGFR 15-30; stage 4). Stage 5 (eGFR <15) refers to the advanced stage of CKD also termed “kidney failure,” which can progress to end-stage kidney disease (ESKD), where dialysis therapy or kidney transplantation is essential to maintain life.

Problem definition & design thinking

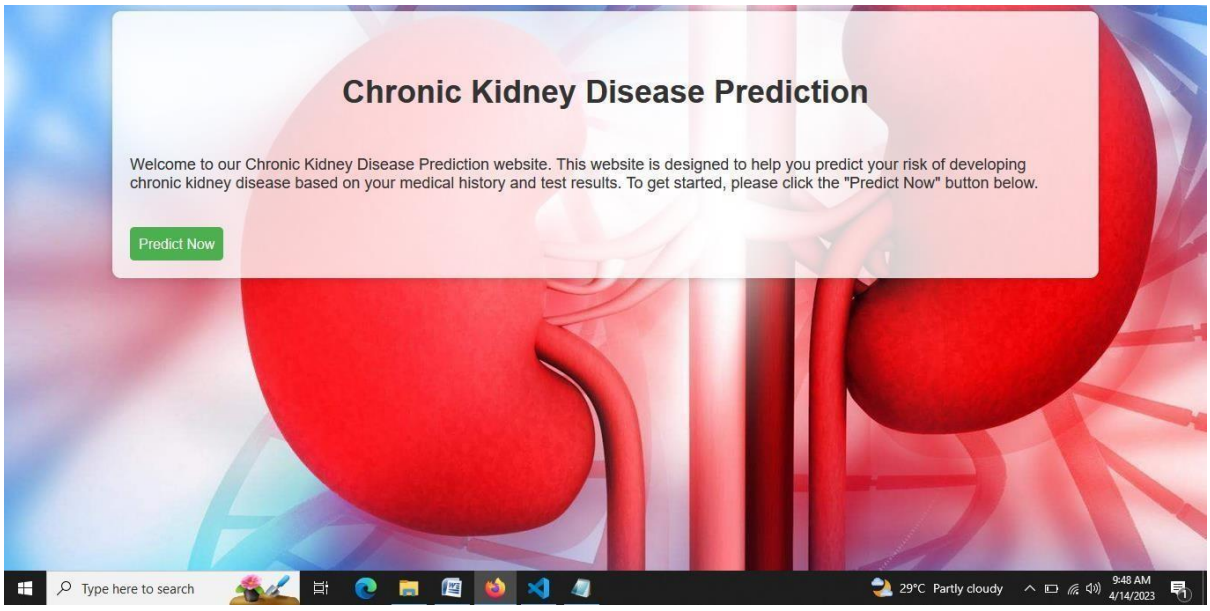
Empathy Map

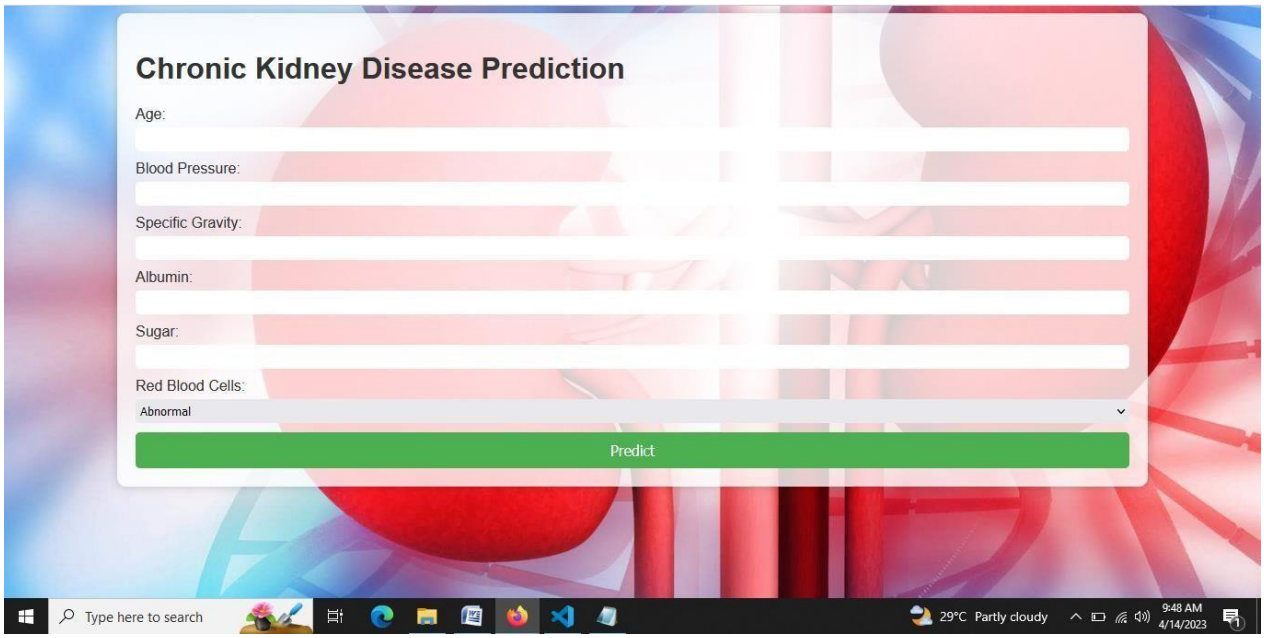


Ideation & Brainstorming Map



RESULT





ADAVANTAGES & DISADVANTAGES

Advantages :

- Managing underlying health conditions: CKD is often caused by underlying health conditions such as diabetes and high blood pressure. Managing these conditions can help slow the progression of CKD.
- Following a healthy diet: A healthy diet can help manage CKD by reducing the workload on the kidneys and reducing the risk of complications such as high blood pressure and diabetes. A healthcare professional can provide personalized dietary recommendations.
- Staying physically active: Regular exercise can help improve overall health and reduce the risk of complications from CKD.
- Taking medications as prescribed: Medications can help manage symptoms of CKD and underlying health conditions. It is important to take medications as prescribed and to talk to a healthcare professional about any concerns or side effects.

Disadvantages :

- Decreased kidney function: CKD causes a progressive loss of kidney function, which can lead to complications such as anemia, bone disease, and increased risk of infections.
- Increased risk of cardiovascular disease: CKD is associated with an increased risk of heart disease, stroke, and other cardiovascular problems.
- Increased healthcare costs: Individuals with CKD often require frequent medical care, which can be costly.
- Dietary restrictions: As kidney function declines, dietary restrictions may be necessary to help manage CKD. This can include limiting protein, sodium, and potassium intake, which can be challenging for some individuals.
- Decreased quality of life: CKD can have a significant impact on an individual's quality of life, including physical and emotional symptoms such as fatigue, depression, and anxiety.

APPLICATION

Your kidneys remove wastes and extra fluid from your body. Your kidneys also remove acid that is produced by the cells of your body and maintain a healthy balance of water, salts, and minerals such as sodium, calcium, phosphorus, and potassium—in your blood.

Each of your kidneys is made up of about a million filtering units called nephrons. Each nephron includes a filter, called the glomerulus, and a tubule. The nephrons work through a two-step process: the glomerulus filters your blood, and the tubule returns needed substances to your blood and removes wastes.

CONCLUSION

Early prediction is very crucial for both experts and patients to prevent and slow down the progress of chronic kidney disease to kidney failure. In this study three machinelearning models RF, SV, DT, and two feature selection methods RFECV and UFS were used to build proposed models. Te evaluation of models were done using tenfold crossvalidation. First, the four machine learning algorithms were applied to original datasets with all 19 features. Applying the models on the original dataset, we have got the highest accuracy with RF, SVM, and XGBoost. Te accuracy was 99.8% for the binary class and 82.56% for fve-class. DT produced lowest performance compared to RF. RF also produced the highest f1_score values. SVM and RF with RFECV produced the highest accuracy of 99.8%for binary class. XGBoost has 82.56% accuracy for fve-class datasets which is the highest.

FUTURE SCOPE

The future of early prediction of chronic kidney disease (CKD) is promising with the advancement of technology and research. Here are some potential future developments that could improve the early prediction of CKD:

- Biomarker research: Biomarkers are measurable indicators of a disease process, and researchers are currently exploring the use of biomarkers to detect early signs of CKD. Advances in biomarker research could lead to more accurate and reliable methods for predicting CKD.
- Artificial intelligence: Machine learning algorithms and artificial intelligence (AI) techniques are being developed to help predict CKD. These methods can analyze large amounts of data and identify patterns that may be indicative of early kidney damage.
- Personalized medicine: Personalized medicine involves tailoring medical treatment to an individual's unique genetic, environmental, and lifestyle factors..

APPENDIX

SOURCE CODE:

```
import pandas as pd
import numpy as np
from collections import Counter as c
import matplotlib.pyplot as plt
import os,sys
import scikitplot as skplt
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.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout

from sklearn.metrics import roc_curve,auc,confusion_matrix,classification_report,accuracy_score
#from sklearn.metrics import precision_recall_curve,average_precision_score,plot_precision_recall_curve,f1_curve

from keras.callbacks import ModelCheckpoint,EarlyStopping
```

```
from keras.models import Sequential,Model

from keras.optimizers import Adam
from sklearn.model_selection import KFold
import pickle
sns.set()

data=pd.read_csv("Chronickidneydisease.csv")

data.head(n=10)

data.shape

data.isnull().sum()

from sklearn.impute import SimpleImputer
imp_mode=SimpleImputer(missing_values=np.nan,strategy='most_frequent')

data_imputed=pd.DataFrame(imp_mode.fit_transform(data))
data_imputed.columns=data.columns
data_imputed

data_imputed.isnull().sum()

for i in data_imputed.columns:
    print("*****************************",i,"*****************************")
    print()
    print(set(data_imputed[i].tolist()))
    print()
```

```
print(data_imputed["rc"].mode())
print(data_imputed["wc"].mode())
print(data_imputed["pcv"].mode())

data_imputed["classification"]=data_imputed["classification"].apply(lambda x:"ckd" if x=="ckd\t" else x)

data_imputed["cad"]=data_imputed["cad"].apply(lambda x:"no" if x=="\tno" else x)

data_imputed["dm"]=data_imputed["dm"].apply(lambda x:"no" if x=="\tno" else x)
data_imputed["dm"]=data_imputed["dm"].apply(lambda x:"yes" if x=="\tyes" else x)


data_imputed["rc"]=data_imputed["rc"].apply(lambda x:"5.2" if x=="\t?" else x)

data_imputed["wc"]=data_imputed["wc"].apply(lambda x:"9800" if x=="\t6200" else x)
data_imputed["wc"]=data_imputed["wc"].apply(lambda x:"9800" if x=="\t8400" else x)
data_imputed["wc"]=data_imputed["wc"].apply(lambda x:"9800" if x=="\t?" else x)

data_imputed["pcv"]=data_imputed["pcv"].apply(lambda x:"41" if x=="\t43" else x)
data_imputed["pcv"]=data_imputed["pcv"].apply(lambda x:"41" if x=="\t?" else x)

for i in data_imputed.columns:
    print("*****************************",i,"*****************************")
    print()
    print(set(data_imputed[i].tolist()))
    print()
```

```
import matplotlib.pyplot as plt
import seaborn as sns

temp=data_imputed["classification"].value_counts()
temp_data=pd.DataFrame({'classification':temp.index,'values':temp.values})
print(sns.barplot(x='classification',y="values",data=temp_data))

data.dtypes

data_imputed.dtypes

data_imputed.dtypes

for i in data.select_dtypes(exclude=["object"]).columns:
    data_imputed[i]=data_imputed[i].apply(lambda x:float(x))

data_imputed.dtypes

sns.pairplot(data_imputed)

def distplots(col):
    sns.distplot(data[col])
    plt.show()

for i in list(data_imputed.select_dtypes(exclude=["object"]).columns)[1:]:
    distplots(i)

def boxplots(col):
    sns.boxplot(data[col])
    plt.show
```

```
for i in list(data_imputed.select_dtypes(exclude=["object"]).columns)[1:]:
    boxplots(i)

from sklearn import preprocessing

data_enco=data_imputed.apply(preprocessing.LabelEncoder().fit_transform)
data_enco

data_enco.to_csv("Kidney_Disease_Pre-processed.csv")

plt.figure(figsize=(20,20))
corr=data_enco.corr()
sns.heatmap(corr,annot=True)

x=data_enco.drop(["id","classification"],axis=1)
y=data_enco["classification"]

from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter

print(Counter(y))

from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter

ros = RandomOverSampler()

x_ros,y_ros=ros.fit_resample(x,y)
print(Counter(y_ros))
```

```
scaler=MinMaxScaler((-1,1))
x=scaler.fit_transform(x_ros)
y=y_ros

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)

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=7)


Run Cell | Run Above | Debug Cell
# In[46]:


import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout

```

```
183
184  from keras.callbacks import ModelCheckpoint,EarlyStopping
185  from keras.models import Sequential,Model
186
187  from keras.optimizers import Adam
188  from sklearn.model_selection import KFold
189
190  def model():
191      classifier=Sequential()
192      classifier.add(Dense(15,input_shape=(x_train.shape[1],),activation='relu'))
193      classifier.add(Dropout(0.2))
194      classifier.add(Dense(15,activation='relu'))
195      classifier.add(Dropout(0.4))
196      classifier.add(Dense(1,activation='sigmoid'))
197      classifier.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
198
199      return classifier
200
201  x_train.shape[1]
202
203  model=model()
204  model.summary()
205
206  history=model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=5,verbose=1)
207
208  import scikitplot as skplt
209  from sklearn.metrics import roc_curve,auc,confusion_matrix,classification_report,accuracy_score
210  from sklearn.metrics import precision_recall_curve,precision_recall_curve,average_precision_score,f1_score,confu
211
212  def plot_auc(t_y,p_y):
213      fpr,tpr,thresholds=roc_curve(t_y,p_y,pos_label=1)
```

C: > Users > Administrator > Downloads > Chronic Kidney Disease.py > ...

```
211
212  def plot_auc(t_y,p_y):
213      fpr,tpr,thresholds=roc_curve(t_y,p_y,pos_label=1)
214      fig,c_ax=plt.subplots(1,1,figsize=(9,9))
215      c_ax.plot(fpr,tpr,label='%s(AUC:%0.2f)' % ('classification',auc(fpr,tpr)))
216      c_ax.plot([0,1],[0,1],color='navy',lw=1,linestyle='--')
217      c_ax.set_xlabel('False Positive Rate')
218      c_ax.set_ylabel('True Positive Rate')
219
220  def plot_precision_recall_curve_helper(t_y,p_y):
221      fig,c_ax=plt.subplots(1,1,figsize=(9,9))
222      precision,recall,thresholds=precision_recall_curve(t_y,p_y,pos_label=1)
223      aps=average_precision_score(t_y,p_y)
224      c_ax.plot(recall,precision,label='%s(AP Score:%0.2f)' %('classification',aps))
225      c_ax.plot(recall,precision,color='red',lw=2)
226      c_ax.legend()
227      c_ax.set_xlabel('Recall')
228      c_ax.set_ylabel('Precision')
229
230  def plot_history(history):
231      f=plt.figure()
232      f.set_figwidth(15)
233
234      f.add_subplot(1,2,1)
235      plt.plot(history.history['val_loss'],label='val loss')
236      plt.plot(history.history['loss'],label='train loss')
237      plt.legend()
238      plt.title("Model Loss")
239
240
241      f.add_subplot(1,2,2)
242      plt.plot(history.history['val_accuracy'],label='val accuracy')
```

```
    plt.plot(history.history['val_accuracy'],label='val accuracy')
    plt.plot(history.history['accuracy'],label='train accuracy')
    plt.legend()
    plt.title("Model Accuracy")
    plt.show()


hist=plot_history(history)

plot_auc(y_test,model.predict(x_test,verbose=True))

plot_precision_recall_curve_helper(y_test,model.predict(x_test,verbose=True))

def calc_f1(prec,recall):
    return 2*(prec*recall)/(prec+recall) if recall and prec else 0

precision,recall,thresholds=precision_recall_curve(y_test,model.predict(x_test,verbose=True))
f1score=[calc_f1(precision[i],recall[i]) for i in range(len(thresholds))]
idx=np.argmax(f1score)
threshold=thresholds[idx]

print('*****************************************************************************************')
print('Precision:'+ str(precision[idx]))
print('Recall:'+ str(recall[idx]))
print('Thresholds:'+ str(thresholds[idx]))
print('F1 Score:'+ str(f1score[idx]))
```

```


plt.figure()
plt.plot(thresholds,f1score)
plt.title("F1-score vs Threshold")
plt.xlabel("thresholds")
plt.ylabel("F1-score")
plt.show()



lgr=LogisticRegression()
pickle.dump(lgr,open('CKDs.pkl','wb'))
```