



EARLY PREDICTION FOR CHRONIC KIDNEY DISEASE DETECTION

TEAM SIZE : 4

TEAM LEADER : IMRAN A (20UCS4715)

TEAM MEMBERS : MURUGESN P (20UCS4724)

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). The 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 machine learning 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) $>90\text{ml/min/1.73m}^2$; 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


Template



Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

[Share template feedback](#)

**Build empathy**

The information you add here should be representative of the observations and research you've done about your users.

Says

What have we heard them say?
What can we imagine them saying?

I have to set my terms

I have to get listed

I can do that I'm motivated

I want to do this but I can't do this alone

Does

What behavior have we observed?
What can we imagine them doing?

Count calories

workout

Don't dare to say no

Only drink water, don't drink fast

Make an appointment

Find anyone

Feels

What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

Down

Loneliness

Insecure

Responsible

Unattractive

Frustrated

Thinks

What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?




Responsibility


Not able to change things


Ability to cope with the situation

I can't do that! Help!





Health Management System



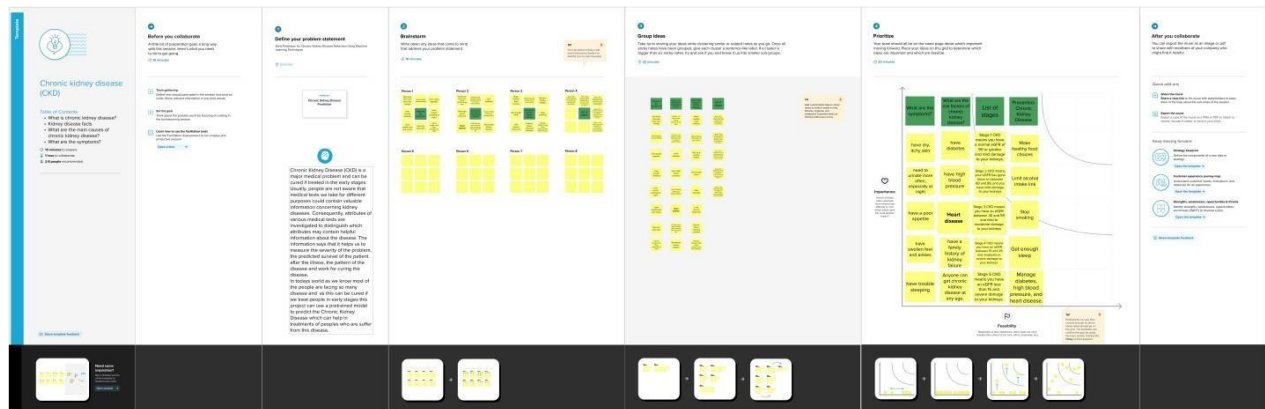




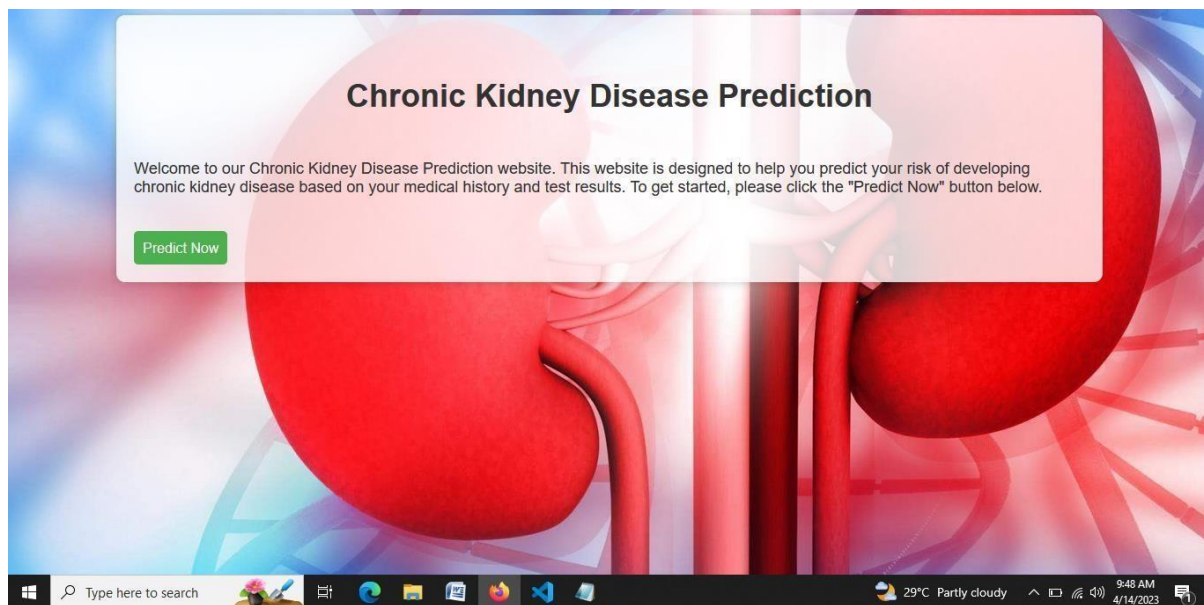
Need some inspiration?
See a finished version of this template to kickstart your work.
[Open example](#)



Ideation & Brainstorming Map



RESULT



Chronic Kidney Disease Prediction

Age:

Blood Pressure:

Specific Gravity:

Albumin:

Sugar:

Red Blood Cells:

ADVANTAGES & 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:

```
6
7 import pandas as pd
8 import numpy as np
9 from collections import Counter as c
10 import matplotlib.pyplot as plt
11 import os,sys
12 import scikitplot as skplt
13 import seaborn as sns
14 import missingno as msno
15 from sklearn.metrics import accuracy_score,confusion_matrix
16 from sklearn.model_selection import train_test_split
17 from sklearn.preprocessing import LabelEncoder
18 from sklearn.preprocessing import MinMaxScaler
19 from sklearn.linear_model import LogisticRegression
20 from imblearn.over_sampling import RandomOverSampler
21 from imblearn.under_sampling import RandomUnderSampler
22 from collections import Counter
23 import keras
24 from keras.models import Sequential
25 from keras.layers import Dense
26 from keras.layers import Dropout
27
28 from sklearn.metrics import roc_curve, auc, confusion_matrix, classification_report, accuracy_score
29 #from sklearn.metrics import precision_recall_curve, average_precision_score, plot_precision_recall_curve, f1_curve
30
31 from keras.callbacks import ModelCheckpoint, EarlyStopping
```

```
32 from keras.models import Sequential, Model
33
34 from keras.optimizers import Adam
35 from sklearn.model_selection import KFold
36 import pickle
37 sns.set()
38
39 data=pd.read_csv("ChronicKidneyDisease.csv")
40
41 data.head(n=10)
42
43 data.shape
44
45 data.isnull().sum()
46
47 from sklearn.impute import SimpleImputer
48 imp_mode=SimpleImputer(missing_values=np.nan, strategy='most_frequent')
49
50 data_imputed=pd.DataFrame(imp_mode.fit_transform(data))
51 data_imputed.columns=data.columns
52 data_imputed
53
54 data_imputed.isnull().sum()
55 |
56 for i in data_imputed.columns:
57     print("*****", i, "*****")
58     print()
59     print(set(data_imputed[i].tolist()))
60     print()
61
62
```



```

66 print(data_imputed["rc"].mode())
67 print(data_imputed["wc"].mode())
68 print(data_imputed["pcv"].mode())
69
70 data_imputed["classification"]=data_imputed["classification"].apply(lambda x:"ckd" if x=="ckd\t" else x)
71
72 data_imputed["cad"]=data_imputed["cad"].apply(lambda x:"no" if x=="\tno" else x)
73
74 data_imputed["dm"]=data_imputed["dm"].apply(lambda x:"no" if x=="\tno" else x)
75 data_imputed["dm"]=data_imputed["dm"].apply(lambda x:"yes" if x=="\tyes" else x)
76
77
78 data_imputed["rc"]=data_imputed["rc"].apply(lambda x:"5.2" if x=="\t?" else x)
79
80 data_imputed["wc"]=data_imputed["wc"].apply(lambda x:"9800" if x=="\t6200" else x)
81 data_imputed["wc"]=data_imputed["wc"].apply(lambda x:"9800" if x=="\t8400" else x)
82 data_imputed["wc"]=data_imputed["wc"].apply(lambda x:"9800" if x=="\t?" else x)
83
84 data_imputed["pcv"]=data_imputed["pcv"].apply(lambda x:"41" if x=="\t43" else x)
85 data_imputed["pcv"]=data_imputed["pcv"].apply(lambda x:"41" if x=="\t?" else x)
86
87 for i in data_imputed.columns:
88     print("*****",i,"*****")
89     print()
90     print(set(data_imputed[i].tolist()))
91     print()
92

```

```

93 import matplotlib.pyplot as plt
94 import seaborn as sns
95
96 temp=data_imputed["classification"].value_counts()
97 temp_data=pd.DataFrame({'classification':temp.index,'values':temp.values})
98 print(sns.barplot(x='classification',y="values",data=temp_data))
99
100 data.dtypes
101
102 data_imputed.dtypes
103
104 data_imputed.dtypes
105
106 for i in data.select_dtypes(exclude=["object"]).columns:
107     data_imputed[i]=data_imputed[i].apply(lambda x:float(x))
108
109 data_imputed.dtypes
110
111 sns.pairplot(data_imputed)
112
113 def distplots(col):
114     sns.distplot(data[col])
115     plt.show()
116
117 for i in list(data_imputed.select_dtypes(exclude=["object"]).columns)[1:]:
118     distplots(i)
119
120 def boxplots(col):
121     sns.boxplot(data[col])
122     plt.show()
123

```

```

123
124 for i in list(data_imputed.select_dtypes(exclude=["object"]).columns)[1:]:
125     boxplots(i)
126
127 from sklearn import preprocessing
128
129 data_enco=data_imputed.apply(preprocessing.LabelEncoder().fit_transform)
130 data_enco
131
132 data_enco.to_csv("Kidney_Disease_Pre-processed.csv")
133
134 plt.figure(figsize=(20,20))
135 corr=data_enco.corr()
136 sns.heatmap(corr,annot=True)
137
138 x=data_enco.drop(["id","classification"],axis=1)
139 y=data_enco["classification"]
140
141 from imblearn.over_sampling import RandomOverSampler
142 from imblearn.under_sampling import RandomUnderSampler
143 from collections import Counter
144
145 print(Counter(y))
146
147 from imblearn.over_sampling import RandomOverSampler
148 from imblearn.under_sampling import RandomUnderSampler
149 from collections import Counter
150
151 ros = RandomOverSampler()
152
153 x_ros,y_ros=ros.fit_resample(x,y)
154 print(Counter(y_ros))

```

```

C:\Users\Administrator\Downloads> Chronic Kidney Disease.py ...
155
156 scaler=MinMaxScaler((-1,1))
157 x=scaler.fit_transform(x_ros)
158 y=y_ros
159
160 import plotly.offline as py
161 py.init_notebook_mode(connected=True)
162 import plotly.graph_objs as go
163 import plotly.tools as tls
164 from sklearn.decomposition import PCA
165
166 pca=PCA(.95)
167 X_PCA=pca.fit_transform(x)
168
169 print(x.shape)
170 print(X_PCA.shape)
171
172 from sklearn.model_selection import train_test_split
173 x_train,x_test,y_train,y_test=train_test_split(X_PCA,y,test_size=0.2,random_state=7)
174
175
176 Run Cell | Run Above | Debug Cell
177 # In[46]:
178
179 import keras
180 from keras.models import Sequential
181 from keras.layers import Dense
182 from keras.layers import Dropout
183

```

```

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')

```

```

242     plt.plot(history.history['val_accuracy'],label='val accuracy')
243     plt.plot(history.history['accuracy'],label='train accuracy')
244     plt.legend()
245     plt.title("Model Accuracy")
246     plt.show()
247
248
249     hist=plot_history(history)
250
251     plot_auc(y_test,model.predict(x_test,verbose=True))
252
253     plot_precision_recall_curve_helper(y_test,model.predict(x_test,verbose=True))
254
255     def calc_f1(prec,recall):
256         return 2*(prec*recall)/(prec+recall) if recall and prec else 0
257
258     precision,recall,thresholds=precision_recall_curve(y_test,model.predict(x_test,verbose=True))
259     f1score=[calc_f1(precision[i],recall[i]) for i in range(len(thresholds))]
260     idx=np.argmax(f1score)
261     threshold=thresholds[idx]
262
263     print('*****')
264     print('Precision: '+ str(precision[idx]))
265     print('Recall: '+ str(recall[idx]))
266     print('Thresholds: '+ str(thresholds[idx]))
267     print('F1 Score: '+ str(f1score[idx]))
268
269
270
271
272

```

```

269
270
271
272
273     plt.figure()
274     plt.plot(thresholds,f1score)
275     plt.title("F1-score vs Threshold")
276     plt.xlabel("thresholds")
277     plt.ylabel("F1-score")
278     plt.show()
279
280
281
282
283     lgr=LogisticRegression()
284     pickle.dump(lgr,open('CKDs.pkl','wb'))
285
286
287
288
289
290
291
292

```