EARLY PREDICTION FOR CHRONIC KIDNEY DISEASE:
A PROGRESSIVE APPROACH TO HEALTH MANAGEMENT

INTRODUCTION

OVERVIEW:

CKD is the significant contributor to morbidity and mortality from non-communicable disease that can affected by 10-15% of the global population.

Early and accurate detection of the stages of CKD is believed to be vital to minimize impacts of patient's health complication such as Hypertension, anemia(low blood count) mineral bone disorder, poor nutritional health, acid base abnormalities and neurological complications.

The prediction model used include Decision Tree(DT), Support vector machine(SVM), Navie Bayes(NB).A report from 1990 to 2013 indicated that the global yearly life loss caused by CKD increased by 90% & it is the 13th leading cause of death in the world[1].

According to the report of the world kidney day of 2019,atleast 2.4 million people die every year.

According to WHO report of 2017,the number of deaths in Ethiopia due to kidney disease was 4,875.The age-adjusted death rate is 8.46 per 100,000 of the population & the death rate increased to 12.70 per 100,000 that has ranked the country 109 in 2018.

Predictive analysis using ML techniques can be helpful through an early detection of CKD for efficient & timely interventions. In this study,(SVM), (DT),(NB) have been used to detect CKD.

PURPOSE

In 2006, CDC established the Chronic Kidney Disease (CKD) Initiative to **provide public health strategies for promoting kidney health**. These strategies seek to: Prevent and control risk factors for CKD. Raise awareness of CKD and its complications.

Academic achievement in children with chronic kidney disease: a report from the CKiD cohort

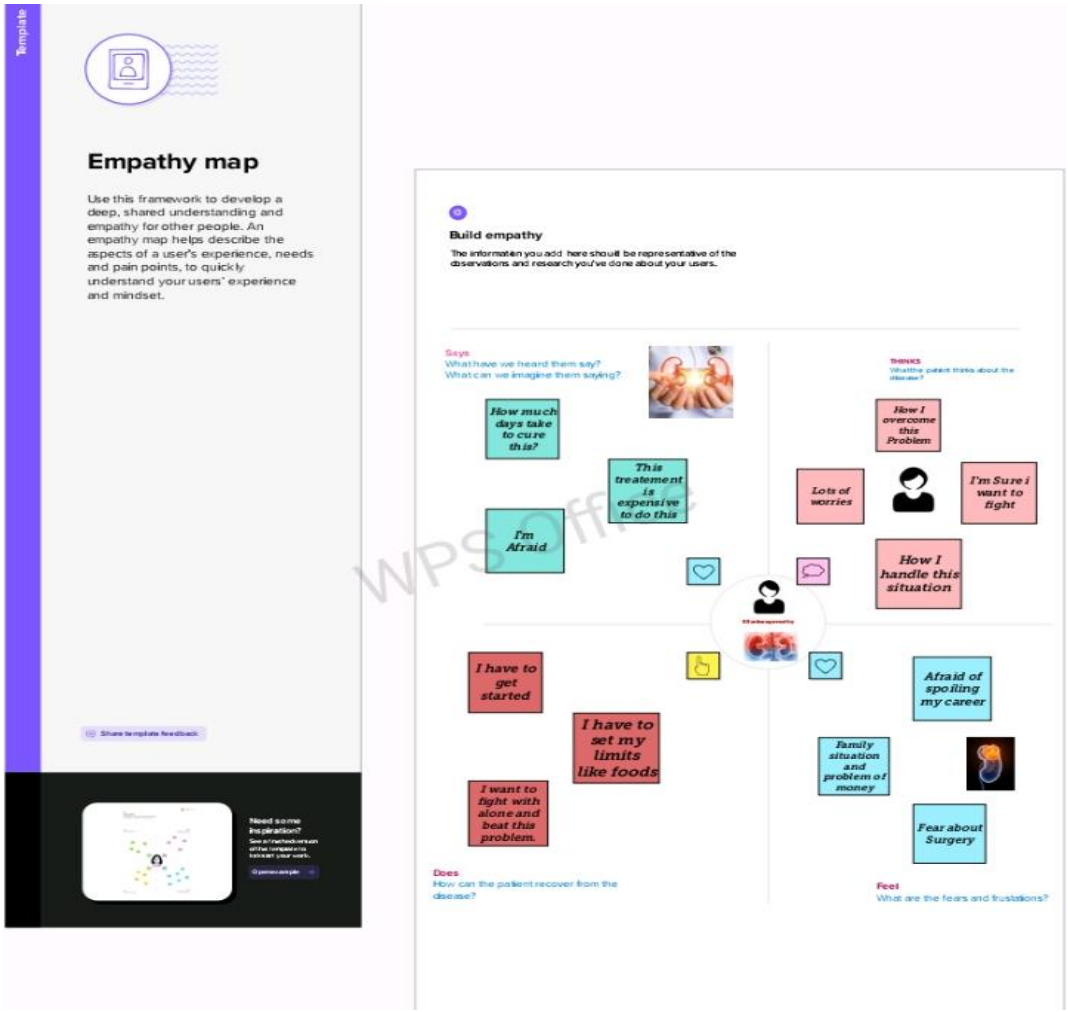
WIAT-II-A data were available for 319 children in the CKiD cohort. Low total academic achievement was present in 34% percent of the sample. There was no significant effect of CKD-related medical variables on academic achievement. Mathematics had the lowest distribution of achievement scores. In univariate models, low achievement was significantly related to days of missed school (p = 0.006) and presence of individualized education plan ($p < 0.0001$).

Low academic achievement was seen in over one-third of children with CKD, with the most difficulty observed in the domain of mathematics. Providers and educators should monitor for academic difficulties in this

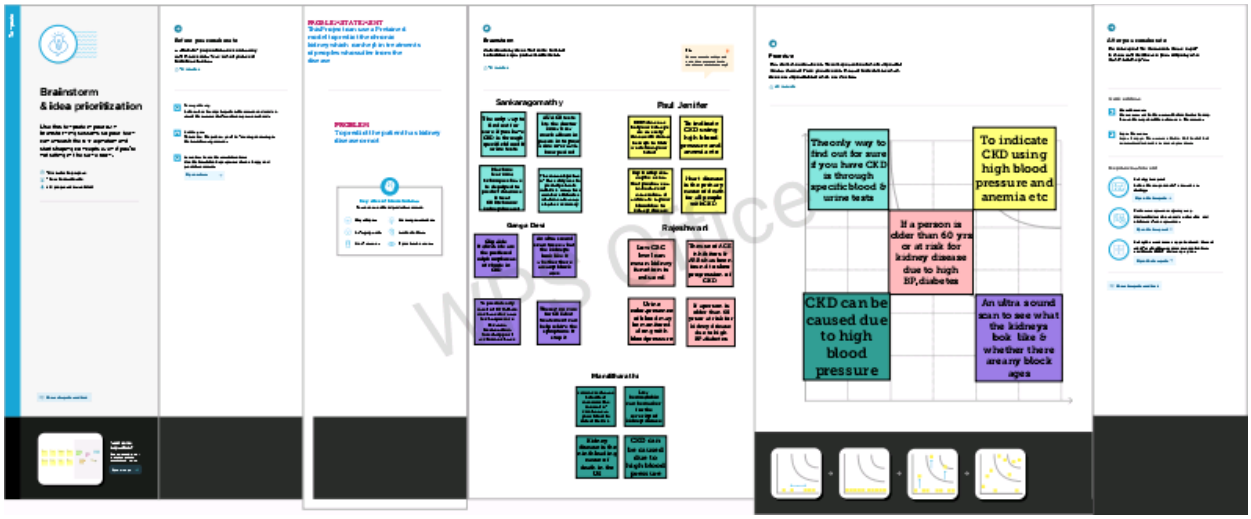
population in order to facilitate early educational assistance and promote positive educational outcomes.

PROBLEM DEFINITION & DESIGN THINKING

Empathy Map



Ideation & Brainstorming Map



RESULT

Importing the Libraries

```
[3]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     %matplotlib inline
     import seaborn as sns
     pd.pandas.set_option('display.max_columns',None)
```

Read the Dataset

```
[6]  dataset = pd.read_csv('/content/chronickidneydisease.csv')
```

```
[8]  dataset.head()
```

	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	classificat
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	36.0	1.2	NaN	NaN	15.4	44	7800	5.2	yes	yes	no	good	no	no	
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	18.0	0.8	NaN	NaN	11.3	38	6000	NaN	no	no	no	good	no	no	
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	53.0	1.8	NaN	NaN	9.6	31	7500	NaN	no	yes	no	poor	no	yes	
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	56.0	3.8	111.0	2.5	11.2	32	6700	3.9	yes	no	no	poor	yes	yes	
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	26.0	1.4	NaN	NaN	11.6	35	7300	4.6	no	no	no	good	no	no	

DATA PREPARATION

Rename the columns

```
dataset.columns = ['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', 'haemoglobin', 'packed_cell_volume', 'white_blood_cell_count', 'red_blood_cell_count',
                   'hypertension', 'diabetes_mellitus', 'coronary_artery_disease', 'appetite', 'peda_edema',
                   'aanemia', 'class']
```

```
[11] dataset.head()
```

	age	blood_pressure	specific_gravity	albumin	sugar	red_blood_cells	pus_cell	pus_cell_clumps	bacteria	blood_glucose_random	blood_urea	serum_creatinine
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	36.0	1.2
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	18.0	0.8
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	53.0	1.8
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	56.0	3.8
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	26.0	1.4

Handling missing values

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   age                     391 non-null    float64
 1   blood_pressure          388 non-null    float64
 2   specific_gravity        353 non-null    float64
 3   albumin                 354 non-null    float64
 4   sugar                   351 non-null    float64
 5   red_blood_cells         248 non-null    object
 6   pus_cell                335 non-null    object
 7   pus_cell_clumps         396 non-null    object
 8   bacteria                396 non-null    object
 9   blood_glucose_random    356 non-null    float64
 10  blood_urea              381 non-null    float64
 11  serum_creatinine        383 non-null    float64
 12  sodium                  313 non-null    float64
 13  potassium               312 non-null    float64
 14  haemoglobin             348 non-null    float64
 15  packed_cell_volume      330 non-null    object
 16  white_blood_cell_count  295 non-null    object
 17  red_blood_cell_count    270 non-null    object
 18  hypertension            398 non-null    object
 19  diabetes_mellitus       398 non-null    object
```

```
dataset.isnull().sum()
```

```
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
haemoglobin                 52
packed_cell_volume          70
white_blood_cell_count     105
red_blood_cell_count       130
hypertension                 2
diabetes_mellitus            2
coronary_artery_disease      2
appetite                     1
peda_edema                   1
aanemia                      1
class                        0
```

```
dataset['diabetes_mellitus'].replace(to_replace = {'\tno':'no','\tyes':'yes',' yes':'yes'},inplace=True)

dataset['coronary_artery_disease'] = dataset['coronary_artery_disease'].replace(to_replace = '\tno', value='no')

dataset['class'] = dataset['class'].replace(to_replace = 'ckd\t', value = 'ckd')

dataset['class'] = dataset['class'].replace(to_replace = 'notckd', value = 'not ckd')
for col in cat_col:
    print('{} has {} values  '.format(col, dataset[col].unique()))
    print('\n')
```

```
red_blood_cells has [nan 'normal' 'abnormal'] values


pus_cell has ['normal' 'abnormal' nan] values


pus_cell_clumps has ['notpresent' 'present' nan] values


bacteria has ['notpresent' 'present' nan] values


hypertension has ['yes' 'no' nan] values
```

Handling Categorical & Numerical Columns

```
cat_cols = [col for col in dataset.columns if dataset[col].dtype == 'object']
num_cols = [col for col in dataset.columns if dataset[col].dtype != 'object']
num_cols = num_cols[:-1]
print("Categorical data : ",cat_cols)
print("Numerical data : ",num_cols)
```

```
Categorical data :  ['red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria', 'hypertension', 'diabetes_mellitus', 'coronary_artery_disease', 'appetite', 'pe
Numerical data :  ['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar', 'blood_glucose_random', 'blood_urea', 'serum_creatinine', 'sodium', 'potassium
```

EXPLORATORY DATA ANALYSIS

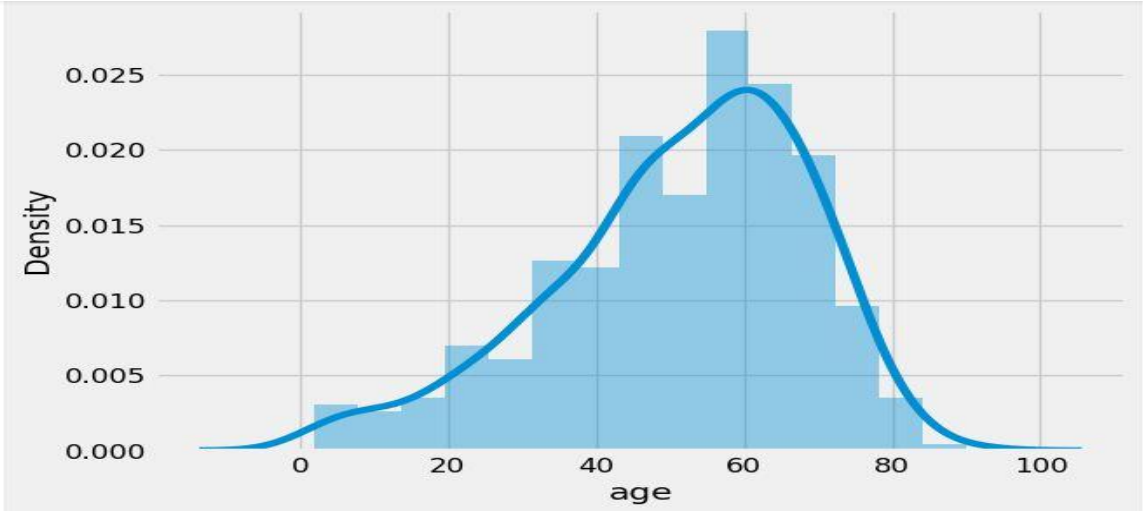
Descriptive statistical Analysis

```
dataset.describe()
```

	age	blood_pressure	specific_gravity	albumin	sugar	blood_glucose_random	blood_urea	serum_creatinine	sodium	potassium	haemoglobin
count	391.000000	388.000000	353.000000	354.000000	351.000000	356.000000	381.000000	383.000000	313.000000	312.000000	348.000000
mean	51.483376	76.469072	1.017408	1.016949	0.450142	148.036517	57.425722	3.072454	137.528754	4.627244	12.526437
std	17.169714	13.683637	0.005717	1.352679	1.099191	79.281714	50.503006	5.741126	10.408752	3.193904	2.912587
min	2.000000	50.000000	1.005000	0.000000	0.000000	22.000000	1.500000	0.400000	4.500000	2.500000	3.100000
25%	42.000000	70.000000	1.010000	0.000000	0.000000	99.000000	27.000000	0.900000	135.000000	3.800000	10.300000
50%	55.000000	80.000000	1.020000	0.000000	0.000000	121.000000	42.000000	1.300000	138.000000	4.400000	12.650000
75%	64.500000	80.000000	1.020000	2.000000	0.000000	163.000000	66.000000	2.800000	142.000000	4.900000	15.000000
max	90.000000	180.000000	1.025000	5.000000	5.000000	490.000000	391.000000	76.000000	163.000000	47.000000	17.800000

Univariate Analysis

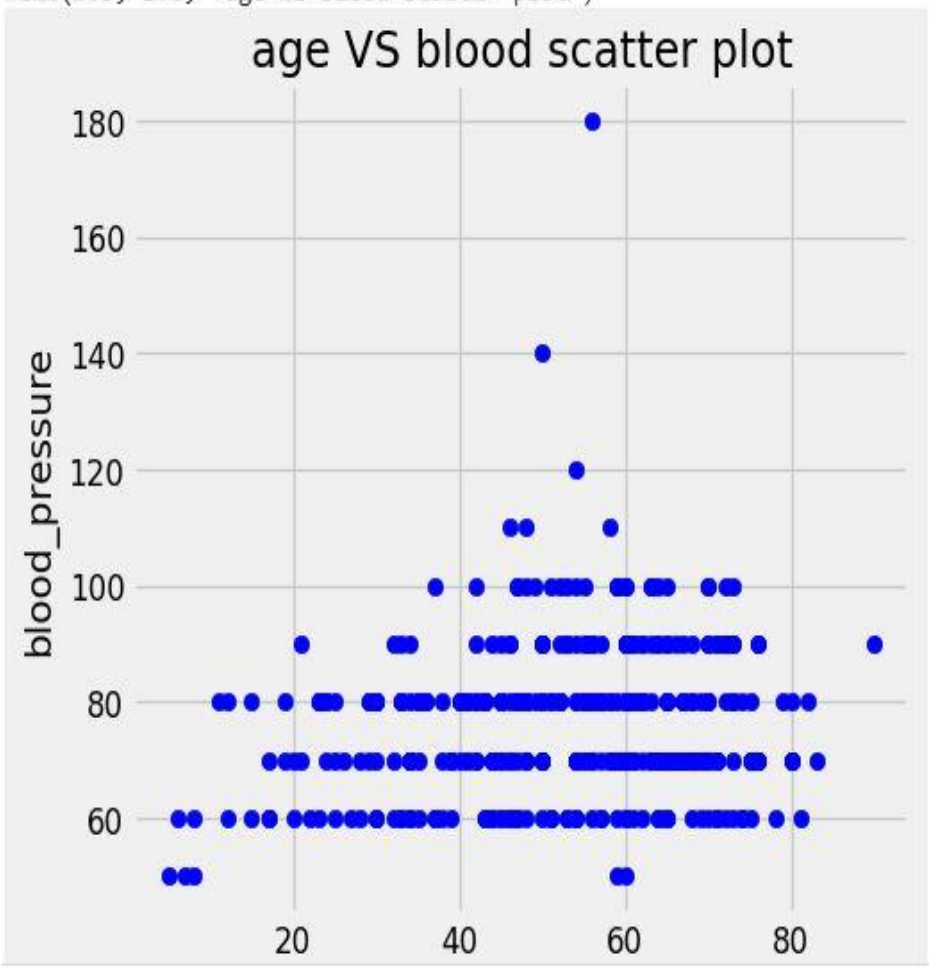
```
sns.distplot(dataset.age)
```



Bivariate Analysis

```
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(5,5))
plt.scatter(dataset['age'],dataset['blood_pressure'],color='blue')
plt.xlabel('age')
plt.ylabel('blood_pressure')
plt.title("age VS blood scatter plot")
```

```
Text(0.5, 1.0, 'age VS blood scatter plot')
```



MODEL BUILDING

Training the model in multiple algorithms

Navie Bayes Model

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

```
GaussianNB()
```

```
y_pred = gnb.predict(X_test)

y_pred
```

```
array([0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 1, 0, 1, 1, 1, 0, 0, 1])
```

```
[58] from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.99	0.94	0.96	72
1	0.92	0.98	0.95	48
accuracy			0.96	120
macro avg	0.95	0.96	0.96	120
weighted avg	0.96	0.96	0.96	120

K-Nearest Neighbor Model

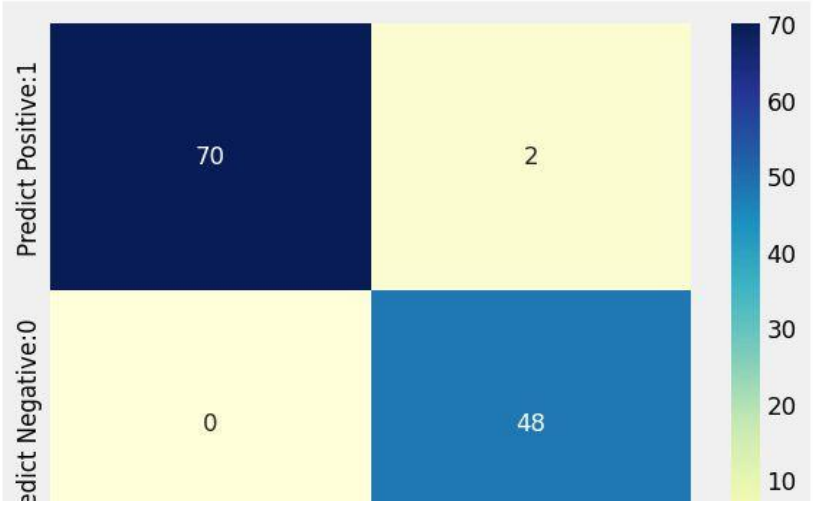
```
classification_accuracy = (TP + TN) / float(TP + TN + FP + FN)
print('Classification accuracy : {0:0.4f}'.format(classification_accuracy))
classification_error = (FP + FN) / float(TP + TN + FP + FN)
print('Classification error : {0:0.4f}'.format(classification_error))
precision = TP / float(TP + FP)
print('Precision : {0:0.4f}'.format(precision))
recall = TP / float(TP + FN)
print('Recall or Sensitivity : {0:0.4f}'.format(recall))
F1_score = 2*precision*recall / float(precision + recall)
print('F-1 Score : {0:0.4f}'.format(F1_score))
true_positive_rate = TP / float(TP + FN)
print('True Positive Rate : {0:0.4f}'.format(true_positive_rate))
false_positive_rate = FP / float(FP + TN)
print('False Positive Rate : {0:0.4f}'.format(false_positive_rate))
specificity = TN / (TN + FP)
print('Specificity : {0:0.4f}'.format(specificity))
```

```
Classification accuracy : 0.9833
Classification error : 0.0167
Precision : 0.9722
Recall or Sensitivity : 1.0000
F-1 Score : 0.9859
True Positive Rate : 1.0000
False Positive Rate : 0.0400
Specificity : 0.9600
```

```
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0'],
                                  index=['Predict Positive:1', 'Predict Negative:0'])

sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

```
<Axes: >
```



Decision Tree Model

```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.figure(figsize=(5,4))
plt.plot(fpr, tpr, linewidth=2)
plt.plot([0,1], [0,1], 'k--' )
plt.rcParams['font.size'] = 12
plt.title('ROC curve for Decision Tree Classifier with criterion entropy for Chronic Kidney Disease Prediction')

plt.xlabel('False Positive Rate (1 - Specificity)')

plt.ylabel('True Positive Rate (Sensitivity)')

plt.show()
```

```
[95] from sklearn.metrics import accuracy_score

     print('Decision Tree Model accuracy score with criterion entropy: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Decision Tree Model accuracy score with criterion entropy: 0.9917
```

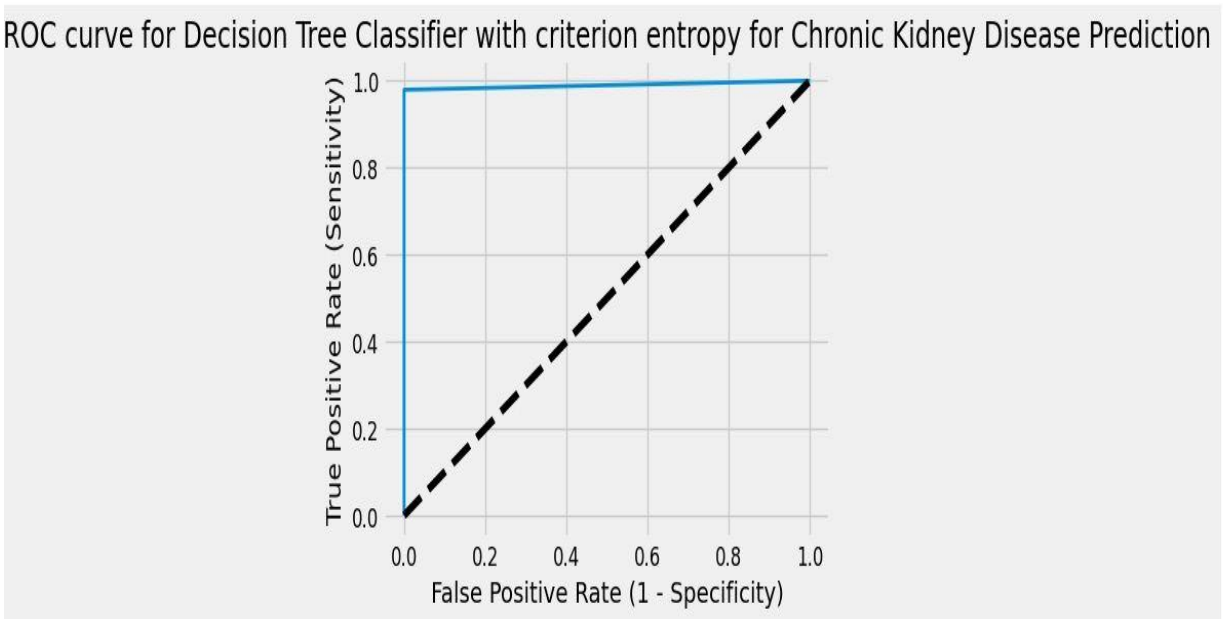
```
[96] data_acc['Decision Tree'] =accuracy_score(y_test, y_pred)*100
```

```
[97] y_pred_proba = clf_en.predict_proba(X_test)
```

```
print('Training set score: {:.4f}'.format(clf_en.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(clf_en.score(X_test, y_test)))
```

```
Training set score: 0.9821
Test set score: 0.9917
```



Support Vector Machine Model

```
[114] from sklearn.svm import SVC
      from sklearn.metrics import accuracy_score
      linear_classifier=SVC(kernel='linear',probability=True).fit(X_train,y_train)
      y_pred = linear_classifier.predict(X_test)
      print('SVM Model accuracy with linear kernel : {0:0.3f}'. format(accuracy_score(y_test, y_pred)))

      SVM Model accuracy with linear kernel : 0.983
```

```
      data_acc['SVM'] =accuracy_score(y_test, y_pred)*100
```

```
[116] y_pred

      array([0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
             0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
             1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1,
             0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1,
             1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1,
             0, 1, 1, 0, 1, 0, 1, 0, 0, 1])
```

ADVANTAGES & DISADVANTAGES OF PROPOSED SOLUTION

ADVANTAGE:

- Early detection of CKD allows proper management that could slow down CKD progression, prevent cardiovascular and other comorbidities and enable timely initiation of dialysis. Screening for CKD could be best managed by partnership between primary care physicians and nephrologists.
- Kidney function tests **check how well your kidneys are working**. Healthy kidneys assist with removing waste from your body.

❖ Conditions such as diabetes or high blood pressure can affect your kidney function. You may also need a kidney function test to diagnose or rule out an infection.

DISADVANTAGE:

Heart Disease and Stroke

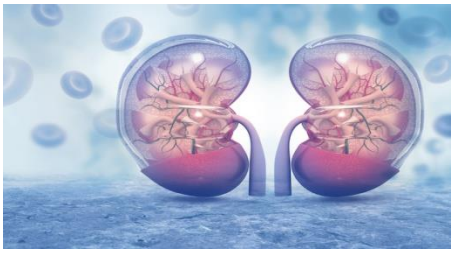
- Having CKD increases the chances of having heart disease and stroke.
- Managing high blood pressure, blood sugar, and cholesterol levels—all factors that increase the risk for heart disease and stroke—is very important for people with CKD.

Early Death

Adults with CKD are at a higher risk of dying earlier than adults of similar age without CKD.

Health Problems Due to Low Kidney Function

- A weakened immune system, which make it easier to develop infections.
- Loss of appetite or nausea.
- Decreased sexual response.



APPLICATIONS:

In kNN, the decision is made by calculating the Euclidian distances between a query and each example in the data, hoosing the value of the example (k) that is closest to the query, and either choosing the most common label for classification or the average of the labels for regression.The value of k is automatically chosen to increase the accuracy of the kNN algorithm.

Support vector machine (SVM) is a promising classical learning method for classification and regression problems and also solves various linear, non-linear, and practical difficulties.

MEDSCAN is an android based application in which Machine Learning and Deep Learning Models are integrated. MEDSCAN is an advanced application that predict the disease on the basis of the XRAY and MRI Scan images. This application uses the advanced models for the prediction of disease.

CONCLUSION:

Severe forms of kidney disease which requires dialysis are curable in some instances. Even if it is not curable, the patient can still lead a meaningful life while on dialysis. Kidney is the only vital organ which can be replaced long term by a machine with reasonable success.

FUTURE SCOPE:

Novel therapeutic alternatives for ESRD include **wearable artificial kidneys, xenotransplantation, stem cell–based therapy, and bioengineered and bio-artificial kidneys**. Of note, one of the main objectives of these novel therapeutic approaches should be to maintain patients at home and to avoid dialysis centers

APPENDIX

Source code:

HTML Files:

```
<div class="row"  style="margin-bottom: 125px;">
<div class="col-md-2"></div>
<div class="col-md-8">
<center><h1>Kidney Disease Predictor</h1></center>
<div class="card card-body" style="border: 1px solid black;">
<form class="form-horizontal" action="{{ url_for('predictPage') }}"
method="POST">
<div class="row">
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="age" placeholder="age">
</div>
</div>
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="bp" placeholder="bp">
</div>
</div>
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
```

```
<div class="row">
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="su" placeholder="su">
</div>
</div>
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="rbc" placeholder="rbc">
</div>
</div>
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="pc" placeholder="pc">
</div>
</div>
</div>
<div class="row">
<div class="col-md-4">
<div class="form-group">
```

```
<input style="border: 1px solid black;" class="form-control"
type="text" name="pcc" placeholder="pcc">
</div>
</div>
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="ba" placeholder="ba">
</div>
</div>
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="bgr" placeholder="bgr">
</div>
</div>
</div>
<div class="row">
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="bu" placeholder="bu">
</div>
```

```
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="sc" placeholder="sc">
</div>
</div>
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="pot" placeholder="pot">
</div>
</div>
</div>
<div class="row">
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="wc" placeholder="wc">
</div>
</div>
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
```

```
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="dm" placeholder="dm">
</div>
</div>
</div>
<div class="row">
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="cad" placeholder="cad">
</div>
</div>
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"
type="text" name="pe" placeholder="pe">
</div>
</div>
<div class="col-md-4">
<div class="form-group">
<input style="border: 1px solid black;" class="form-control"


<input type="submit" class="btn btn-info btn-block" value="Predict">
</form>
</div>
</div>
<div class="col-md-2"></div>
</div> <br>
```

```
<div class="row"  style="margin-bottom: 477px;">
<div class="col-md-3"></div>
<div class="col-md-6">
<div class="jumbotron ">
<h1 class="display-4">You have a Kidney Disease !</h1>
<p class="lead">Please Consult the Doctor Immideately.
It was too risky without consultation. Make sure of health in your diet.</p>
<hr class="my-4">
</div>
<div class="jumbotron">
<h1 class="display-4">Great! You are Healthy</h1>
<p class="lead">You are Absolutely Alright !
There is no Marks for Kidney Disease. Enjoy your life with full of Happiness.</p>
<hr class="my-4">
<div class="col-md-3"></div>
</div>
```

FLASK Files:

```
[ ] import pickle
    pickle.dump(knn,open('CKD.pkl','wb'))
```

```
[ ] !pip install flask
    from flask import Flask,render_template,request
    import pickle
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: flask in /usr/local/lib/python3.9/dist-packages (2.2.3)
Requirement already satisfied: Werkzeug>=2.2.2 in /usr/local/lib/python3.9/dist-packages (from flask) (2.2.3)
Requirement already satisfied: importlib-metadata>=3.6.0 in /usr/local/lib/python3.9/dist-packages (from flask) (6.3.0)
Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.9/dist-packages (from flask) (2.1.2)
Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.9/dist-packages (from flask) (8.1.3)
Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.9/dist-packages (from flask) (3.1.2)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.9/dist-packages (from importlib-metadata>=3.6.0->flask) (3.15.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from Jinja2>=3.0->flask) (2.1.2)
```

```
[ ] app=Flask(__name__)
    model=pickle.load(open('CKD.pkl','rb'))
```

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
import pandas as pd

app = Flask(__name__)
model = pickle.load(open('CKD.pkl','rb'))
@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    one = ['yes', 'present', 'good', 'normal', 'Yes', 'Present', 'Good', 'Normal', 'YES', 'PRESENT', 'GOOD', 'NORMAL']
    zero = ['no', 'notpresent', 'not present', 'poor', 'abnormal', 'No', 'Notpresent', 'NotPresent', 'Not Present', 'Poor', 'Abnormal', 'AbNormal', 'NO',
    int_features = []
    for i in request.form.values():
        if i in one:
            int_features.append(1.0)
        elif i in zero:
            int_features.append(0.0)
        else:
            int_features.append(float(i))
```

```
    final_features = [np.array(int_features)]

    final_features = scaler.transform(final_features)
    prediction = model.predict(final_features)

    output = prediction

    if output == [0]:
        output = "Kidney Disease Not Detected"
    elif output == [1]:
        output = "Kidney Disease Detected"

    return render_template('predict.html', prediction_text='Diagnosis Result: {}'.format(output))

if __name__ == "__main__":
    app.run(debug=True)
```

```
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
```