Chronic Kidney Disease Analysis using Decision Tree Classifier

DONE BY:

1. Mohammad Abdul Razzaq
2. G. Lahari

INDEX:

1. INTRODUCTION:

1.1 OVERVIEW:

Chronic kidney disease (CKD) is a condition characterized by a gradual loss of kidney function over time. Kidney disease increases your risk of having heart and blood vessel disease. These problems may happen slowly over a long period. Chronic kidney disease may be caused by diabetes, high blood pressure and other disorders. Early detection and treatment can often keep chronic kidney disease from getting worse. When kidney disease progresses, it may eventually lead to kidney failure, which requires dialysis or a kidney transplant to maintain life.

1.2 PURPOSE:

The purpose of our project is to find out whether the person is having a Chronic Kidney disease or not using Machine Learning algorithm.If the doctors have a good tool that can identify patients who are likely to have kidney disease in advance, they can heal the patients in time. So based on the data collected we built a model which can predict whether the person will have a chronic kidney disease or not considering the previous records of some patients. This model considers a patient's Age, Blood Pressure, Specific Gravity, Albumin, Sugar, Red Blood Cells, Pus Cell, Pus Cell clumps etc and gives the output.

2. LITERATURE SURVEY

2.1 Existing Problem:

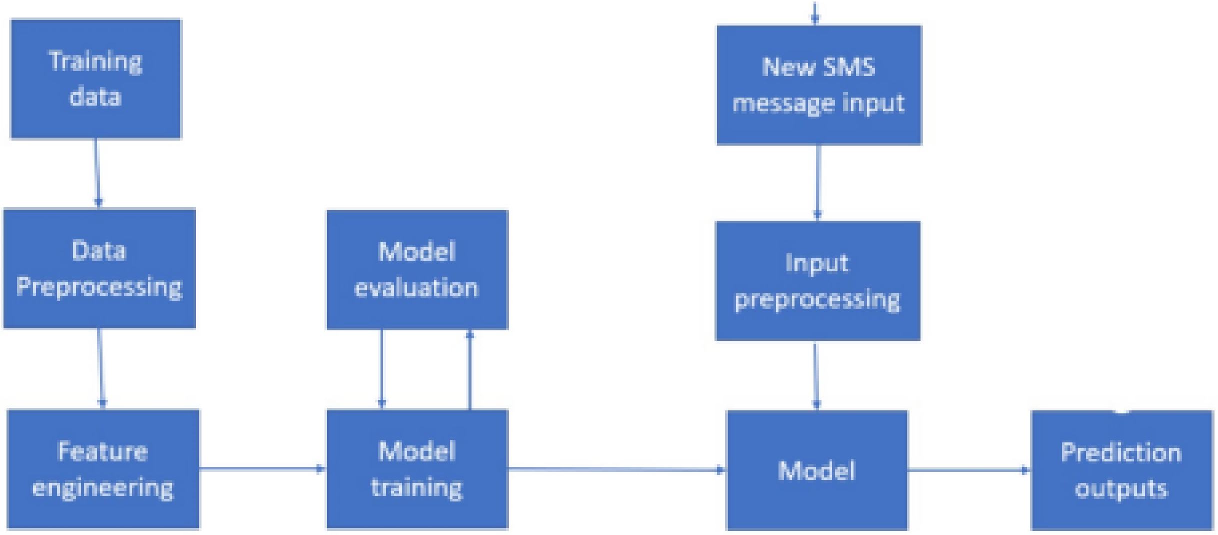
Chronic kidney disease (CKD) affects a sizable percentage of the world's population. If detected early, its adverse effects can be avoided, hence saving precious lives and reducing cost. We have been able to build a model based on labeled data that accurately predicts if a patient suffers from chronic kidney disease based on their characteristics.

2.2 Proposed Solution:

We have built a machine learning model using DecisionTree Classifier. We have split the data into training and testing sets. We trained the data and checked the accuracy of the model using test set. This model gave an accuracy of 97 percent. Using python and HTML we created a web application to display the output (i.e whether the person is having CKD or not) so we could use this model in hospitals to identify the people who may suffer with CKD at an early stage.

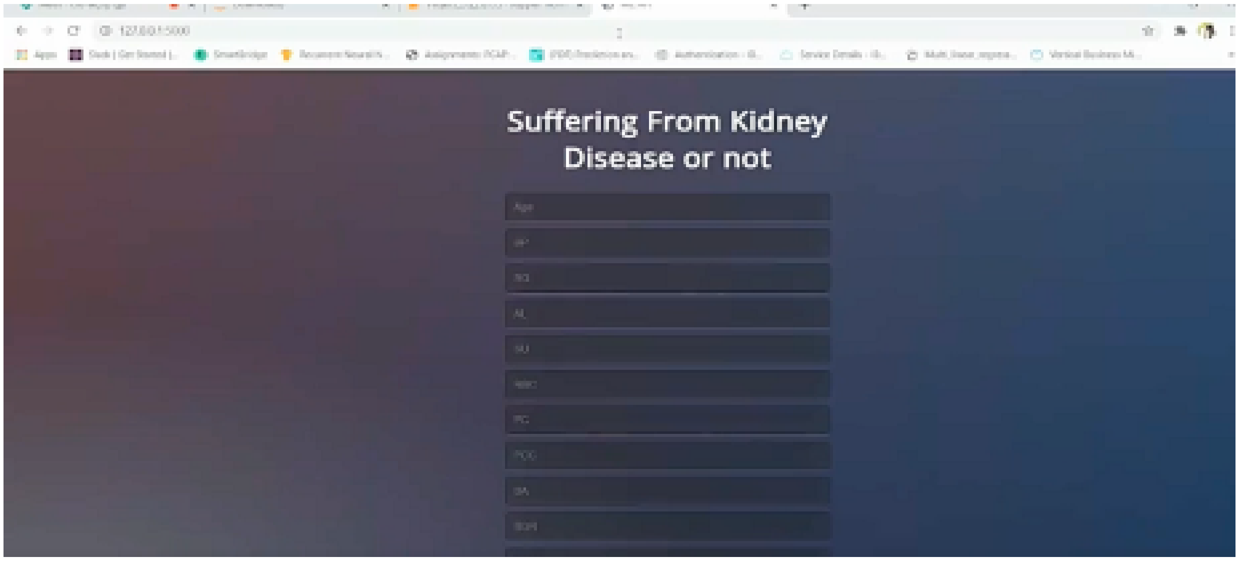
3. THEORITICAL ANALYSIS

3.1 Block diagram:



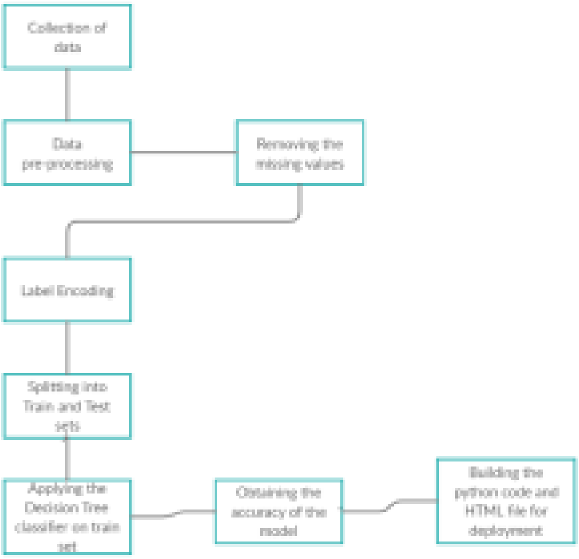
3.2 Software Designing:

The web page we created



Suffering From Kidney Disease or not

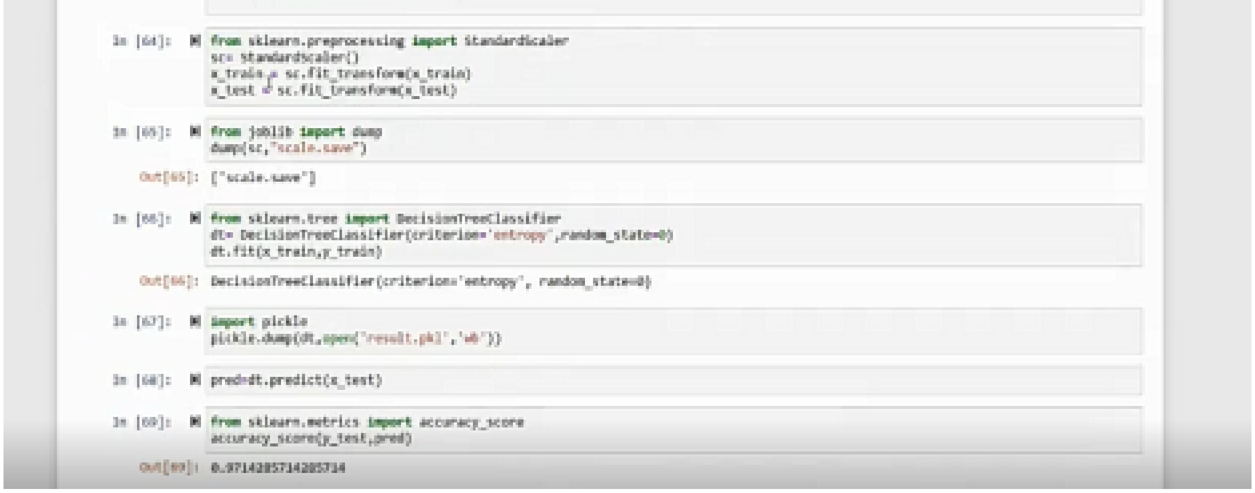
4. FLOWCHART



Collection of data
Data pre-processing
Removing the missing values
Label Encoding
Splitting into Train and Test sets
Applying the Decision Tree classifier on train set
Obtaining the accuracy of the model
Building the python code and HTML file for deployment

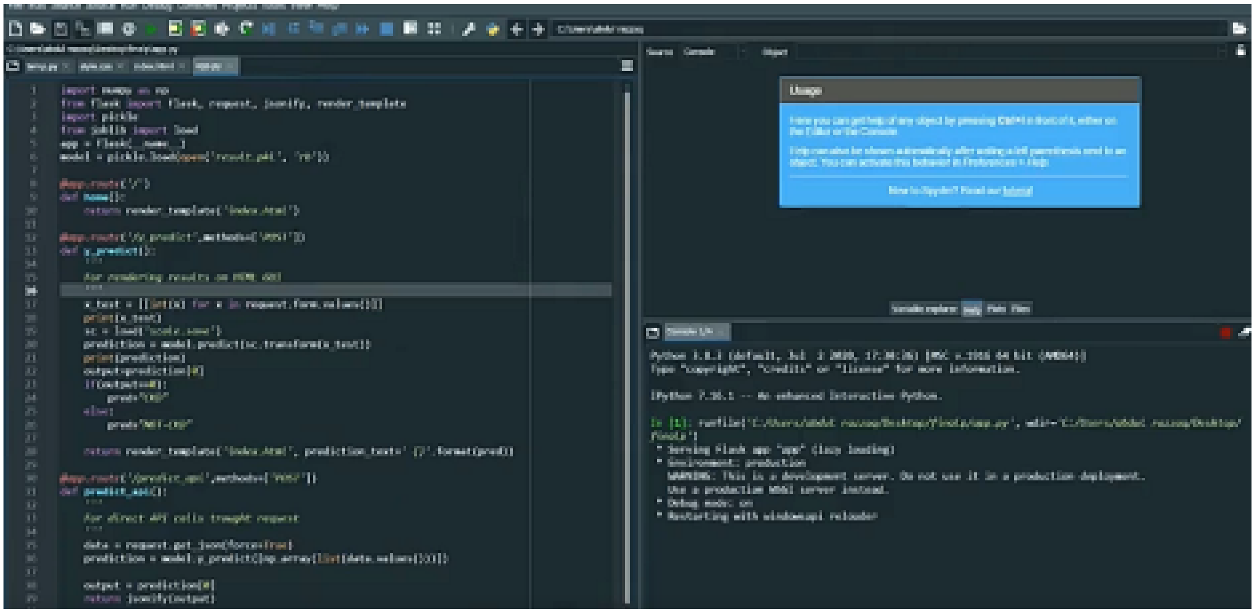
5. RESULT:

Accuracy:



```
In [64]:  from sklearn.preprocessing import StandardScaler
          sc= StandardScaler()
          x_train = sc.fit_transform(x_train)
          x_test = sc.fit_transform(x_test)

In [65]:  from joblib import dump
          dump(sc,"scale.save")

Out[65]:  ['scale.save']

In [66]:  from sklearn.tree import DecisionTreeClassifier
          dt= DecisionTreeClassifier(criterion='entropy',random_state=0)
          dt.fit(x_train,y_train)

Out[66]:  DecisionTreeClassifier(criterion='entropy', random_state=0)

In [67]:  import pickle
          pickle.dump(dt,open('result.pkl','wb'))

In [68]:  pred=dt.predict(x_test)

In [69]:  from sklearn.metrics import accuracy_score
          accuracy_score(y_test,pred)

Out[69]:  0.9714285714285714
```

app.py running:



```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
from joblib import load
app = Flask(__name__)
model = pickle.load(open('result.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/y_predict',methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML GUI
    '''
    x_test = [[int(x) for x in request.form.values()]]
    print(x_test)
    sc = load('scale.save')
    prediction = model.predict(sc.transform(x_test))
    print(prediction)
    output=prediction[0]
    if(output==0):
        pred="CKD"
    else:
        pred="NOT-CKD"

    return render_template('index.html', prediction_text=' {}'.format(pred))

@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    prediction = model.y_predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)
```

Usage

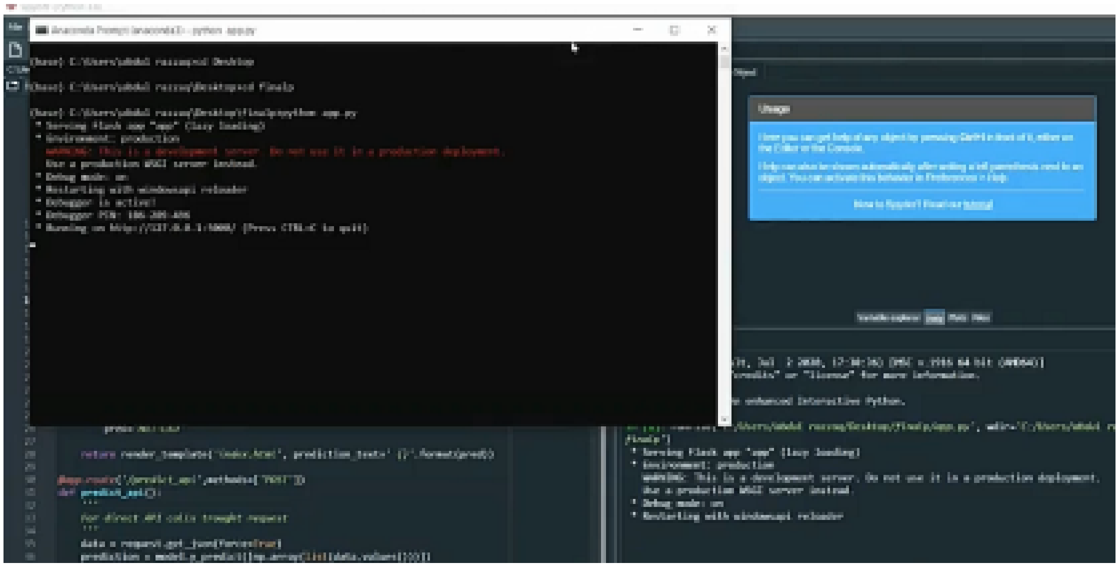
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.

New to Spyder? Read our tutorial

```
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/abdul razzaq/Desktop/flask/app.py', wdir='C:/Users/abdul razzaq/Desktop/
flask')
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
```

Anaconda prompt output:

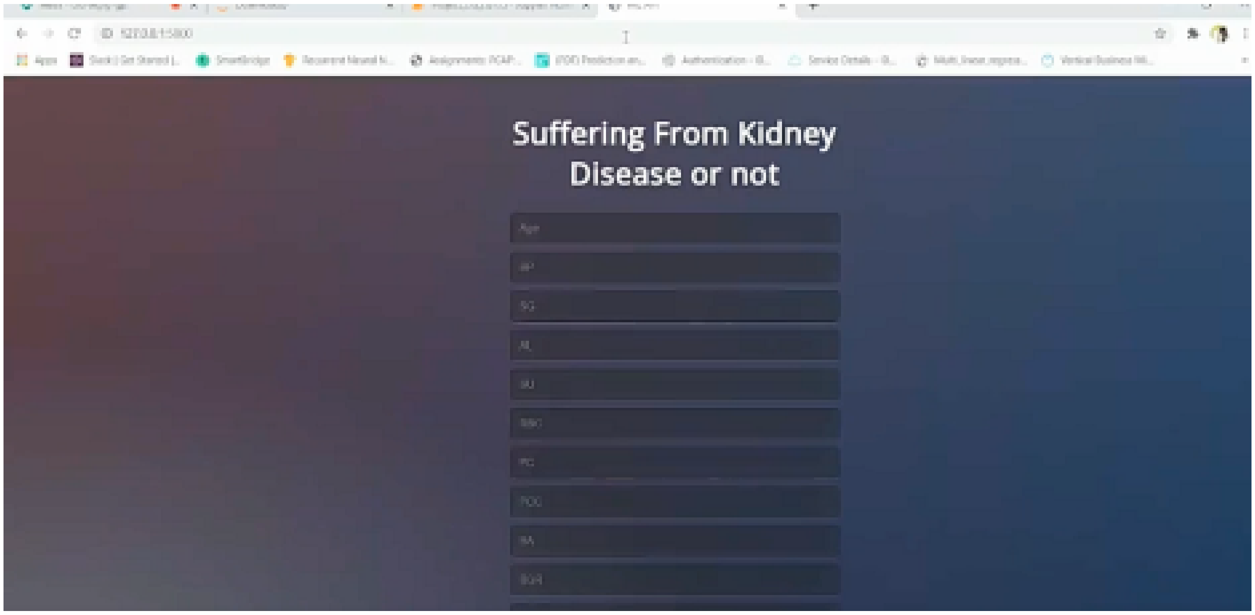


```
(base) C:\Users\abdul razzaq>cd Desktop

(base) C:\Users\abdul razzaq\Desktop>cd finalp

(base) C:\Users\abdul razzaq\Desktop\finalp>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 186-209-496
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

```
    return render_template('index.html', prediction_text=' {}'.format(pred))

@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    prediction = model.y_predict([np.array(list(data.values()))])
```

```
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
```

Web page (entering the input variables):



127.0.0.1:5000

Suffering From Kidney Disease or not

Age

BP

SG

AL

SU

RBC

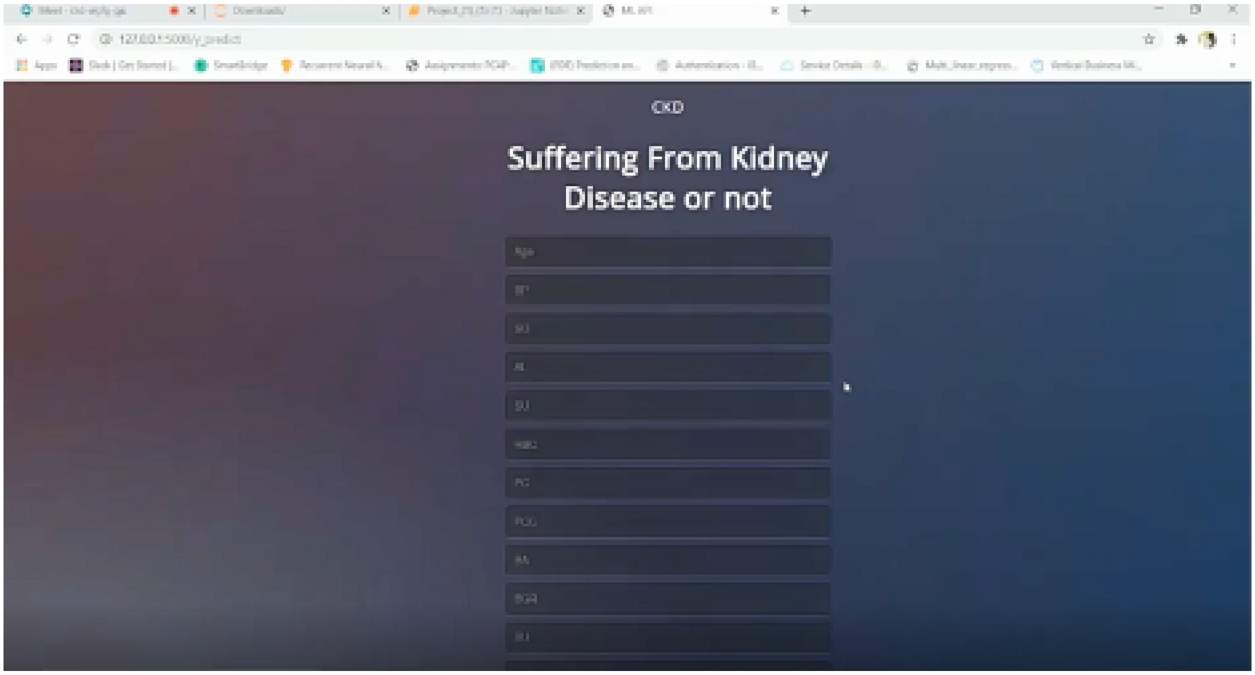
PC

PCC

BA

BGR

Final Output:



CKD

Suffering From Kidney Disease or not

6.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

● Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. ● With ML, you don’t need to takecare of your project every step. ● As algorithms gain experience, they keep improving in accuracy and efficiency.

DISADVANTAGES:

- Machine Learning requires massive data sets to train on, and these should be unbiased, and of good quality.
- ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy.

7. APPLICATIONS:

The successful implementation of ML methods can help the integration of computer-based systems in the healthcare environment providing opportunities to facilitate and enhance the work of medical experts and ultimately to improve the efficiency and quality of medical care.

8. CONCLUSION:

● As we can see we got the accuracy score as 0.97 we could use this model to obtain the desired results on giving the inputs to the model. ● This could help us to save people from CKD, as we can predict at the early stages.

9. FUTURE SCOPE:

● There are other possible evolutionary techniques that may be used to improve results of the proposed classifier.

● In this Decision tree classifier is applied to detect CKD. ● We can also evaluate and compare the performance of the used classifier with other existing classifiers.

● CKD early detection helps in timely treatment of the patients suffering from the disease and also to avoid the disease from getting worse. ● Early prediction of the disease and timely treatment are the need for medical sector.

● New classifiers can be used and their performance can be evaluated to find better solutions of the objective function in future work.

10. BIBLIOGRAPHY:

1. https://www.kidney.org/atoz/content/about-chronic-kidney-disease#ckd

2. https://www.ijert.org/research/comparative-study-of-chronic-kidney-disease-prediction-using-knn-and-svm-IJERTV4IS120622.pdf 3.

https://data-flair.training/blogs/advantages-and-disadvantages-of-machine-learning/#:~:text=Advantages%20of%20Machine%20learning.%201%201.%20Easily%20identifies,and%20multi-variety%20data.%205%205.%20Wide%20Applications.%20

APPENDIX:

Code:

In [1]:

```
#importing the libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
#importing the dataset
data=pd.read_csv("kidney_disease (1).csv")
```

In [65]:

```
data
```

Out[65]:

	id	age	bp	sg	al	su		c pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.020	1.0	0.0	normal	normal	notpresent	notpresent	...	44.0	7800.0	5.2	yes	yes	no	good	no	no	ckd

1	1	7.0	50.0	1.020	4.0	0.0	normal	normal	notpresent	notpresent	...	38.0	6000.0	5.2	no	no	no	good	no	no	ckd
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31.0	7500.0	5.2	no	yes	no	poor	no	yes	ckd
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32.0	6700.0	3.9	yes	no	no	poor	yes	yes	ckd
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35.0	7300.0	4.6	no	no	no	good	no	no	ckd
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	395	55.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	47.0	6700.0	4.9	no	no	no	good	no	no	notckd
396	396	42.0	70.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	54.0	7800.0	6.2	no	no	no	good	no	no	notckd

397	397	12.0	80.0	1.020	0.0	0.0	normal	normal	notpresent	notpresent	...	49.0	6600.0	5.4	no	no	no	good	no	no	notckd
398	398	17.0	60.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	51.0	7200.0	5.9	no	no	no	good	no	no	notckd
399	399	58.0	80.0	1.025	0.0	0.0	normal	normal	notpresent	notpresent	...	53.0	6800.0	6.1	no	no	no	good	no	no	notckd

400 rows × 26 columns

In [4]:

```
data.isnull().any() #checking for missing values
```

Out[4]:

```
id          False
age           True
bp           True
sg           True
al          True
su           True
rbc           True
pc           True
pcc           True
ba           True
bgr           True
bu           True
sc           True
sod           True
pot           True
hemo           True
pcv           True
wc           True
```

```
rc                True
htn               True
dm                True
cad               True
appet             True
pe                True
ane               True
classification    False
dtype: bool
```

In [5]:

```
data['pcv']=data['pcv'].replace('?','')
data['wc']=data['wc'].replace('?','')
data['rc']=data['rc'].replace('?','')
```

In [6]:

```
data['pcv']=pd.to_numeric(data['pcv'])
```

In [7]:

```
data['wc']=pd.to_numeric(data['wc'])
```

In [8]:

```
data['rc']=pd.to_numeric(data['rc'])
```

In [9]:

```
data['age']=data['age'].fillna(value=data['age'].mean())
```

In [10]:

```
data['bp']=data['bp'].fillna(value=data['bp'].mean())
```

In [11]:

```
data['sg']=data['sg'].fillna(value=data['sg'].mean())
```

In [12]:

```
data['al']=data['al'].fillna(value=data['al'].mean())
```

In [13]:

```
data['su']=data['su'].fillna(value=data['su'].mean())
```

In [14]:

```
data['rbc']=data['rbc'].fillna(value=data['rbc'].mode().iloc[0])
```

In [15]:

```
data['pc']=data['pc'].fillna(value=data['pc'].mode().iloc[0])
```

In [16]:

```
data['pcc']=data['pcc'].fillna(value=data['pcc'].mode().iloc[0])
```

In [17]:

```
data['ba']=data['ba'].fillna(value=data['ba'].mode().iloc[0])
```

In [18]:

```
data['bgr']=data['bgr'].fillna(value=data['bgr'].mean())
```

In [19]:

```
data['bu']=data['bu'].fillna(value=data['bu'].mean())
```

In [20]:

```
data['sc']=data['sc'].fillna(value=data['sc'].mean())
```

In [21]:

```
data['sod']=data['sod'].fillna(value=data['sod'].mean())
```

In [22]:

```
data['pot']=data['pot'].fillna(value=data['pot'].mean())
```

In [23]:

```
data['hemo']=data['hemo'].fillna(value=data['hemo'].mean())
```

In [24]:

```
data['pcv']=data['pcv'].fillna(value=data['pcv'].mode().iloc[0])
```

In [25]:

```
data['wc']=data['wc'].fillna(value=data['wc'].mode().iloc[0])
```

In [26]:

```
data['rc']=data['rc'].fillna(value=data['rc'].mode().iloc[0])
```

In [27]:

```
data['htn']=data['htn'].fillna(value=data['htn'].mode().iloc[0])
```

In [28]:

```
data['dm']=data['dm'].fillna(value=data['dm'].mode().iloc[0])
```

In [29]:

```
data['cad']=data['cad'].fillna(value=data['cad'].mode().iloc[0])
```

In [30]:

```
data['appet']=data['appet'].fillna(value=data['appet'].mode().iloc[0])
```

In [31]:

```
data['pe']=data['pe'].fillna(value=data['pe'].mode().iloc[0])
```

In [32]:

```
data['ane']=data['ane'].fillna(value=data['ane'].mode().iloc[0])
```

In [33]:

```
data['pcv']=pd.to_numeric(data['pcv'])
```

In [34]:

```
data.isnull().any()
```

Out[34]:

```
id               False
age              False
bp               False
sg               False
al               False
su               False
rbc              False
pc               False
pcc              False
ba               False
bgr              False
bu               False
sc               False
```

```
sod              False
pot              False
hemo             False
pcv              False
wc               False
rc               False
htn              False
dm               False
cad              False
appet            False
pe               False
ane              False
classification   False
dtype: bool
```

In [35]:

```
x=data.iloc[:,1:25]
```

In [36]:

```
type(x)
```

Out[36]:

```
pandas.core.frame.DataFrame
```

In [37]:

```
x=data.iloc[:,1:25].values
```

In [38]:

```
x
```

Out[38]:

```
array([[48.0, 80.0, 1.02, ..., 'good', 'no', 'no'],
       [7.0, 50.0, 1.02, ..., 'good', 'no', 'no'],
       [62.0, 80.0, 1.01, ..., 'poor', 'no', 'yes'],
       ...,
       [12.0, 80.0, 1.02, ..., 'good', 'no', 'no'],
       [17.0, 60.0, 1.025, ..., 'good', 'no', 'no'],
       [58.0, 80.0, 1.025, ..., 'good', 'no', 'no']], dtype=object)
```

In [39]:

```
y=data.iloc[:,25:].values
```

In [40]:

```
from sklearn.preprocessing import LabelEncoder
```

In [41]:

```
lb=LabelEncoder()
```

In [42]:

```
x[:,5]=lb.fit_transform(x[:,5])
```

In [43]:

```
x
```

Out[43]:

```
array([[48.0, 80.0, 1.02, ..., 'good', 'no', 'no'],
       [7.0, 50.0, 1.02, ..., 'good', 'no', 'no'],
       [62.0, 80.0, 1.01, ..., 'poor', 'no', 'yes'],
       ...,
       [12.0, 80.0, 1.02, ..., 'good', 'no', 'no'],
       [17.0, 60.0, 1.025, ..., 'good', 'no', 'no'],
       [58.0, 80.0, 1.025, ..., 'good', 'no', 'no']], dtype=object)
```

In [44]:

```
x[:,6]=lb.fit_transform(x[:,6])
```

In [45]:

```
x[:,7]=lb.fit_transform(x[:,7])
```

In [46]:

```
x[:,8]=lb.fit_transform(x[:,8])
```

In [47]:

```
x[:,18]=lb.fit_transform(x[:,18])
```

In [48]:

```
x[:,19]=lb.fit_transform(x[:,19])
```

In [49]:

```
x[:,20]=lb.fit_transform(x[:,20])
```

In [50]:

```
x[:,21]=lb.fit_transform(x[:,21])
```

In [51]:

```
x[:,22]=lb.fit_transform(x[:,22])
```

In [52]:

```
x[:,23]=lb.fit_transform(x[:,23])
```

In [53]:

```
y[:,0]=lb.fit_transform(y[:,0])
```

In [54]:

```
x[0:5,:]
```

Out[54]:

```
array([[48.0, 80.0, 1.02, 1.0, 0.0, 1, 1, 0, 0, 121.0, 36.0, 1.2,
        137.52875399361022, 4.627243589743592, 15.4, 44.0, 7800.0, 5.2,
        1, 1, 0, 0, 0, 0],
       [7.0, 50.0, 1.02, 4.0, 0.0, 1, 1, 0, 0, 148.0365168539326, 18.0,
        0.8, 137.52875399361022, 4.627243589743592, 11.3, 38.0, 6000.0,
        5.2, 0, 0, 0, 0, 0, 0],
       [62.0, 80.0, 1.01, 2.0, 3.0, 1, 1, 0, 0, 423.0, 53.0, 1.8,
        137.52875399361022, 4.627243589743592, 9.6, 31.0, 7500.0, 5.2, 0,
        1, 0, 1, 0, 1],
       [48.0, 70.0, 1.005, 4.0, 0.0, 1, 0, 1, 0, 117.0, 56.0, 3.8, 111.0,
        2.5, 11.2, 32.0, 6700.0, 3.9, 1, 0, 0, 1, 1, 1],
       [51.0, 80.0, 1.01, 2.0, 0.0, 1, 1, 0, 0, 106.0, 26.0, 1.4,
        137.52875399361022, 4.627243589743592, 11.6, 35.0, 7300.0, 4.6,
        0, 0, 0, 0, 0, 0]], dtype=object)
```

In [55]:

```
from sklearn.model_selection import train_test_split
```

In [56]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.35,random_state=1)
```

In [57]:

```
y_train =y_train.flatten()
y_test = y_test.flatten()
```

In [58]:

```
y_train = y_train.astype(np.float)
y_test = y_test.astype(np.float)
```

In [59]:

```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

In [60]:

```
from joblib import dump
dump(sc,"scale.save")
```

Out[60]:

```
['scale.save']
```

In [61]:

```
from sklearn.tree import DecisionTreeClassifier
dt= DecisionTreeClassifier(criterion='entropy',random_state=0)
dt.fit(x_train,y_train)
```

Out[61]:

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [62]:

```
import pickle
pickle.dump(dt,open('result.pkl','wb'))
```

In [63]:

```
pred=dt.predict(x_test)
```

In [64]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,pred)
```

Out[64]:

```
0.9714285714285714
```

app.py:

```
import numpy as np
from flask import Flask, request, jsonify, render_template import pickle
from joblib import load
app = Flask(__name__)
model = pickle.load(open('result.pkl', 'rb'))


@app.route('/')
def home():
    return render_template('index.html')

@app.route('/y_predict',methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML GUI
    '''
    x_test = [[int(x) for x in request.form.values()]] print(x_test)
    sc = load('scale.save')
    prediction = model.predict(sc.transform(x_test)) print(prediction)
    output=prediction[0]
    if(output==0):
        pred="CKD"
    else:
        pred="NOT-CKD"

    return render_template('index.html', prediction_text=' {}'.format(pred))

@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
                      prediction = model.y_predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

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

HTML file:

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet
' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300
' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
<style>
.login{
top: 20%;
}
</style>
</head>

<body>
 <div class="login">
```

```
  <br>
   {{ prediction_text }}
<h1>Suffering From Kidney Disease or not </h1>

     <!-- Main Input For Receiving Query to our ML -->
     <form action="{{ url_for('y_predict')}}"method="post">
      <input type ="number"placeholder="Age" name="age"/>
      <input type ="number"  placeholder="BP" name="bp"/>
<input type ="number" placeholder="SG" name="sg"/>

      <input type ="number" placeholder="AL" name="al"/>
<input type ="number"  placeholder="SU" name="su"/>
<input type ="number" placeholder="RBC" name="rbc"/>
<input type ="number" placeholder="PC" name="pc"/>
<input type ="number" placeholder="PCC" name="pcc"/>
<input type ="number" placeholder="BA" name="ba"/>
<input type ="number" placeholder="BGR" name="bgr"/>
<input type ="number" placeholder="BU" name="bu"/>
<input type ="number" placeholder="SC" name="sc"/>
<input type ="number" placeholder="SOD" name="sod"/>
 <input type ="number" placeholder="POT" name="pot"/>
 <input type ="number" placeholder="HEMO" name="hemo"/>
 <input type ="number" placeholder="PVC" name="pcv"/>
<input type ="number" placeholder="WC" name="wc"/>
 <input type ="number" placeholder="RC" name="rc"/>
 <input type ="number" placeholder="HTN" name="htn"/>
<input type ="number" placeholder="DM"  name="dm"/>
<input type ="number" placeholder="CAD" name="cad"/>
<input type ="number" placeholder="APPET" name="appet"/>
<input type ="number" placeholder="PE" name="pe"/>
 <input type ="number" placeholder="ANE" name="ane"/>
 <p><input type="submit"value="submit"/></p>  </form>
<br>
</div>
</body>
</html>
```