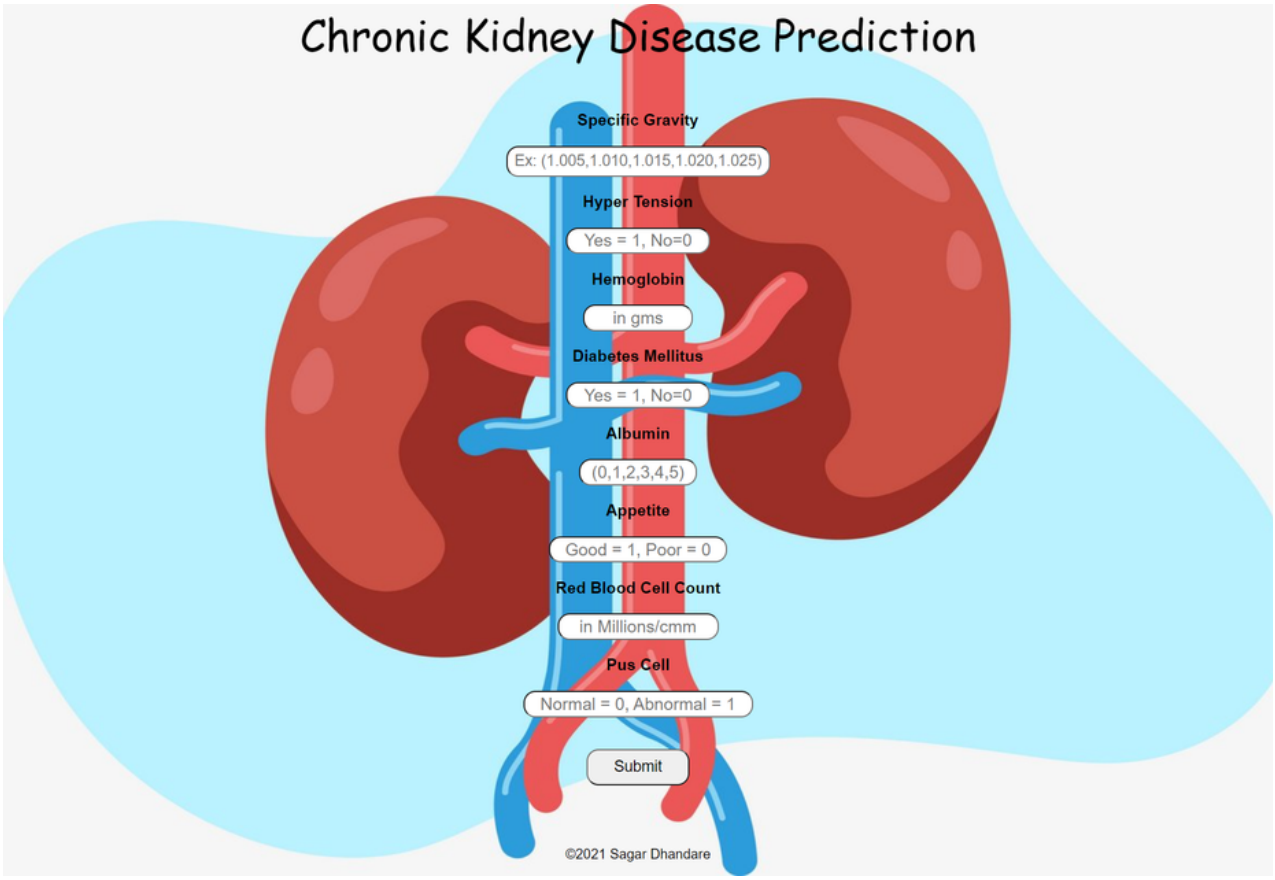
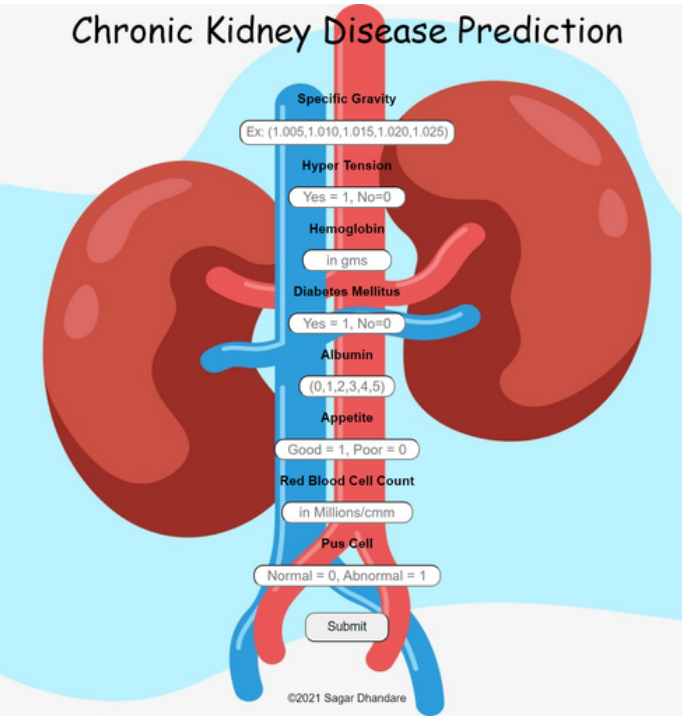
Chronic Kidney Disease Prediction

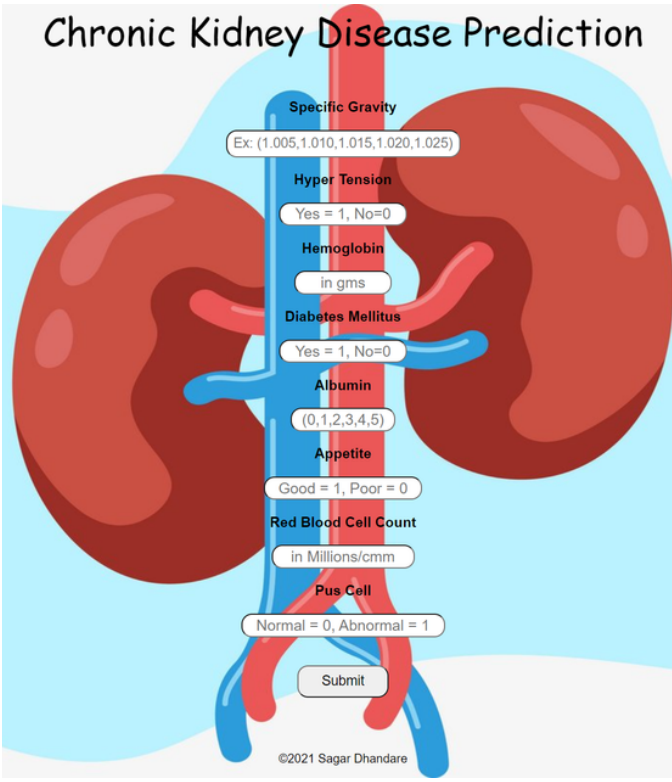


Are you in Healthcare department .
This Project is for you .
Follow me →

Prediction of this Project

The Person is affected by





Approaching of this Problem

Output

- The Output column is clear and Classification type (Yes or no).
- So its come under Machine Learning -> Supervised Learning -> Classificatiion .

Let's move to coding part →

RandomForestClassificationAssignment

May 22, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as matlab
```

[3]:
```
dataset=pd.read_csv("CKD.csv")
dataset
```

[3]:
```
           age         bp sg   al   su     rbc        pc         pcc
0     2.000000  76.459948  c  3.0  0.0  normal  abnormal  notpresent  \
1     3.000000  76.459948  c  2.0  0.0  normal    normal  notpresent
2     4.000000  76.459948  a  1.0  0.0  normal    normal  notpresent
3     5.000000  76.459948  d  1.0  0.0  normal    normal  notpresent
4     5.000000  50.000000  c  0.0  0.0  normal    normal  notpresent
..         …        … ..  …  …     …       …         …
394  51.492308  70.000000  a  0.0  0.0  normal    normal  notpresent
395  51.492308  70.000000  c  0.0  2.0  normal    normal  notpresent
396  51.492308  70.000000  c  3.0  0.0  normal    normal  notpresent
397  51.492308  90.000000  a  0.0  0.0  normal    normal  notpresent
398  51.492308  80.000000  a  0.0  0.0  normal    normal  notpresent

              ba         bgr  …        pcv            wc        rc  htn   dm
0    notpresent  148.112676  …  38.868902   8408.191126  4.705597   no   no  \
1    notpresent  148.112676  …  34.000000  12300.000000  4.705597   no   no
2    notpresent   99.000000  …  34.000000   8408.191126  4.705597   no   no
3    notpresent  148.112676  …  38.868902   8408.191126  4.705597   no   no
4    notpresent  148.112676  …  36.000000  12400.000000  4.705597   no   no
..          …         …  …        …            …        …  …  …
394  notpresent  219.000000  …  37.000000   9800.000000  4.400000   no   no
395  notpresent  220.000000  …  27.000000   8408.191126  4.705597  yes  yes
396  notpresent  110.000000  …  26.000000   9200.000000  3.400000  yes  yes
397  notpresent  207.000000  …  38.868902   8408.191126  4.705597  yes  yes
398  notpresent  100.000000  …  53.000000   8500.000000  4.900000   no   no

     cad  appet    pe  ane classification
0     no    yes   yes   no            yes
1     no    yes  poor   no            yes
2     no    yes  poor   no            yes
3     no    yes  poor  yes            yes
```

```
4     no    yes  poor   no            yes
..   …     …    …   …             …
394   no    yes  poor   no            yes
395   no    yes  poor  yes            yes
396   no   poor  poor   no            yes
397   no    yes  poor  yes            yes
398   no    yes  poor   no             no

[399 rows x 25 columns]
```

[4]:
```
dataset=pd.get_dummies(dataset,drop_first=True)
dataset
```

[4]:
```
           age         bp   al   su         bgr          bu        sc
0     2.000000  76.459948  3.0  0.0  148.112676   57.482105  3.077356  \
1     3.000000  76.459948  2.0  0.0  148.112676   22.000000  0.700000
2     4.000000  76.459948  1.0  0.0   99.000000   23.000000  0.600000
3     5.000000  76.459948  1.0  0.0  148.112676   16.000000  0.700000
4     5.000000  50.000000  0.0  0.0  148.112676   25.000000  0.600000
..         …         …  …  …         …          …       …
394  51.492308  70.000000  0.0  0.0  219.000000   36.000000  1.300000
395  51.492308  70.000000  0.0  2.0  220.000000   68.000000  2.800000
396  51.492308  70.000000  3.0  0.0  110.000000  115.000000  6.000000
397  51.492308  90.000000  0.0  0.0  207.000000   80.000000  6.800000
398  51.492308  80.000000  0.0  0.0  100.000000   49.000000  1.000000

            sod       pot       hrmo  …  pc_normal  pcc_present  ba_present
0    137.528754  4.627244  12.518156  …      False        False       False  \
1    137.528754  4.627244  10.700000  …       True        False       False
2    138.000000  4.400000  12.000000  …       True        False       False
3    138.000000  3.200000   8.100000  …       True        False       False
4    137.528754  4.627244  11.800000  …       True        False       False
..          …       …         …  …        …           …           …
394  139.000000  3.700000  12.500000  …       True        False       False
395  137.528754  4.627244   8.700000  …       True        False       False
396  134.000000  2.700000   9.100000  …       True        False       False
397  142.000000  5.500000   8.500000  …       True        False       False
398  140.000000  5.000000  16.300000  …       True        False       False

     htn_yes  dm_yes  cad_yes  appet_yes  pe_yes  ane_yes  classification_yes
0      False   False    False       True    True    False                True
1      False   False    False       True   False    False                True
2      False   False    False       True   False    False                True
3      False   False    False       True   False     True                True
4      False   False    False       True   False    False                True
..       …       …        …          …       …        …                   …
394    False   False    False       True   False    False                True
```

```
395     True    True    False       True   False     True                 True
396     True    True    False      False   False    False                 True
397     True    True    False       True   False     True                 True
398    False   False    False       True   False    False                False

[399 rows x 28 columns]
```

[5]:
```
dataset.columns
```

[5]:
```
Index(['age', 'bp', 'al', 'su', 'bgr', 'bu', 'sc', 'sod', 'pot', 'hrmo', 'pcv',
       'wc', 'rc', 'sg_b', 'sg_c', 'sg_d', 'sg_e', 'rbc_normal', 'pc_normal',
       'pcc_present', 'ba_present', 'htn_yes', 'dm_yes', 'cad_yes',
       'appet_yes', 'pe_yes', 'ane_yes', 'classification_yes'],
      dtype='object')
```

[6]:
```
dataset["classification_yes"].value_counts()
```

[6]:
```
classification_yes
True     249
False    150
Name: count, dtype: int64
```

[7]:
```
independent=dataset[['age', 'bp', 'al', 'su', 'bgr', 'bu', 'sc', 'sod', 'pot',␣
 ↪'hrmo', 'pcv',
       'wc', 'rc', 'sg_b', 'sg_c', 'sg_d', 'sg_e', 'rbc_normal', 'pc_normal',
       'pcc_present', 'ba_present', 'htn_yes', 'dm_yes', 'cad_yes',
       'appet_yes', 'pe_yes', 'ane_yes']]
dependent=dataset[['classification_yes']]
```

[8]:
```
independent.shape
```

[8]:
```
(399, 27)
```

[9]:
```
dependent
```

[9]:
```
     classification_yes
0                  True
1                  True
2                  True
3                  True
4                  True
..                  …
394                True
395                True
396                True
397                True
398               False
```

```
[399 rows x 1 columns]
```

[10]:
```
#split into training set and test
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(independent,dependent,test_size=1/
↪3,random_state=0)
```

[11]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

[12]:
```
from sklearn.ensemble import RandomForestClassifier
```

[13]:
```
from sklearn.model_selection import GridSearchCV
param_grid={'criterion':['gini','entropy'],'max_features':
↪['auto','sqrt','log2']}
grid=GridSearchCV(RandomForestClassifier(),param_grid,refit=True,verbose=3,n_jobs=-1,scoring='
#fitting the model for grid search
grid.fit(X_train,Y_train)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits

D:\anaconda3\envs\aiml2\lib\site-
packages\sklearn\model_selection\_search.py:909: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
  self.best_estimator_.fit(X, y, **fit_params)
```

[13]:
```
GridSearchCV(estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_features': ['auto', 'sqrt', 'log2']},
             scoring='f1_weighted', verbose=3)
```

[14]:
```
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=0)
classifier.fit(X_train,Y_train)
```

```
C:\Users\spavi\AppData\Local\Temp\ipykernel_13256\1498940516.py:3:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train,Y_train)
```

[14]:
```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

[15]:
```
y_pred=classifier.predict(X_test)
```

[16]:
```
y_pred
```

[16]:
```
array([False,  True, False,  True,  True,  True,  True,  True, False,
       False,  True,  True, False,  True, False, False,  True, False,
        True,  True, False,  True,  True, False,  True, False, False,
       False,  True,  True,  True,  True, False, False, False, False,
        True, False,  True, False, False,  True,  True,  True,  True,
        True,  True, False,  True,  True,  True,  True,  True, False,
        True,  True, False,  True, False, False, False,  True,  True,
        True,  True,  True,  True,  True, False,  True,  True,  True,
       False, False, False,  True,  True,  True,  True, False,  True,
        True, False, False,  True,  True, False,  True,  True,  True,
        True,  True, False,  True,  True,  True,  True,  True,  True,
        True,  True,  True, False,  True, False,  True, False,  True,
       False,  True,  True, False, False, False,  True,  True,  True,
       False, False, False,  True, False,  True,  True,  True, False,
       False,  True, False, False,  True,  True, False])
```

[17]:
```python
re=grid.cv_results_
grid_predictions=grid.predict(X_test)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,y_pred)
```

[18]:
```python
print(cm)
```

```
[[50  1]
 [ 1 81]]
```

[19]:
```python
from sklearn.metrics import classification_report
clf_report=classification_report(Y_test,y_pred)
```

[20]:
```python
print(clf_report)
```

```
              precision    recall  f1-score   support

       False       0.98      0.98      0.98        51
        True       0.99      0.99      0.99        82

    accuracy                           0.98       133
   macro avg       0.98      0.98      0.98       133
weighted avg       0.98      0.98      0.98       133
```

[21]:
```python
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test)[:,1])
```

[21]: 0.9997608799617408

[22]:
```python
y_pred=classifier.predict(X_test)
y_pred
```

[22]:
```
array([False,  True, False,  True,  True,  True,  True,  True, False,
       False,  True,  True, False,  True, False, False,  True, False,
        True,  True, False,  True,  True, False,  True, False, False,
       False,  True,  True,  True,  True, False, False, False, False,
        True, False,  True, False, False,  True,  True,  True,  True,
        True,  True, False,  True,  True,  True,  True,  True, False,
        True,  True, False,  True, False, False, False,  True,  True,
        True,  True,  True,  True,  True, False,  True,  True,  True,
       False, False, False,  True,  True,  True,  True, False,  True,
        True, False, False,  True,  True, False,  True,  True,  True,
        True,  True, False,  True,  True,  True,  True,  True,  True,
        True,  True,  True, False,  True, False,  True, False,  True,
       False,  True,  True, False, False, False,  True,  True,  True,
       False, False, False,  True, False,  True,  True,  True, False,
       False,  True, False, False,  True,  True, False])
```

[23]:
```
table=pd.DataFrame.from_dict(re)
table
```

[23]:
```
   mean_fit_time  std_fit_time  mean_score_time  std_score_time
0       0.671718      0.145541         0.903983        0.428615  \
1       0.387407      0.006247         0.043740        0.006247
2       0.381162      0.007654         0.043739        0.006249
3       0.412403      0.007652         0.043741        0.006249
4       0.393657      0.006249         0.040616        0.007653
5       0.387409      0.006248         0.046864        0.000001

  param_criterion param_max_features
0            gini               auto  \
1            gini               sqrt
2            gini               log2
3         entropy               auto
4         entropy               sqrt
5         entropy               log2

                                            params  split0_test_score
0     {'criterion': 'gini', 'max_features': 'auto'}           0.981569  \
1     {'criterion': 'gini', 'max_features': 'sqrt'}           0.981569
2     {'criterion': 'gini', 'max_features': 'log2'}           1.000000
3  {'criterion': 'entropy', 'max_features': 'auto'}           0.981569
4  {'criterion': 'entropy', 'max_features': 'sqrt'}           1.000000
5  {'criterion': 'entropy', 'max_features': 'log2'}           1.000000

   split1_test_score  split2_test_score  split3_test_score  split4_test_score
0           0.961755           0.962573           0.981031                1.0  \
1           0.981014           0.962573           0.962264                1.0
2           0.961755           0.962573           0.981031                1.0
```

```
3           0.961755           0.962573           0.981031                1.0
4           0.961755           0.944023           0.981031                1.0
5           0.961755           0.962573           0.981031                1.0

   mean_test_score  std_test_score  rank_test_score
0         0.977386        0.014184                4
1         0.977484        0.014072                3
2         0.981072        0.016923                1
3         0.977386        0.014184                4
4         0.977362        0.021879                6
5         0.981072        0.016923                1
```

[24]:
```
age_input=float(input("Age:"))
bp_input=float(input("BP:"))
al_input=float(input("AL:"))
su_input=float(input("SU:"))
bgr_input=float(input("BGR:"))
bu_input=float(input("BU:"))
sc_input=float(input("SC:"))
sod_input=float(input("SOD:"))
pot_input=float(input("POT:"))
hrmo_input=float(input("HRMO:"))
pcv_input=float(input("PCV:"))
wc_input=float(input("WC:"))
rc_input=float(input("RC:"))
sg_b_input=float(input("SG_B:"))
sg_c_input=float(input("SG_C:"))
sg_d_input=float(input("SG_D:"))
sg_e_input=float(input("SG_E:"))
rbc_normal_input=float(input("RBC Normal 0 or 1:"))
pc_normal_input=float(input("PC Normal 0 or 1:"))
pcc_present_input=float(input("PC Present 0 or 1:"))
ba_present_input=float(input("BA_Present 0 or 1:"))
htn_yes_input=float(input("HTN 0 or 1:"))
dm_yes_input=float(input("DM 0 or 1:"))
cad_yes_input=float(input("CAD 0 or 1:"))
appet_yes_input=float(input("Appet 0 or 1:"))
pe_yes_input=float(input("PE 0 or 1:"))
ane_yes_input=float(input("ANE 0 or 1:"))
```

```
Age:50
BP:80.2
AL:3.0
SU:1.0
BGR:150.35436
BU:47.89
SC:4.67654
SOD:174.857
```

```
POT:4.74256
HRMO:15.9698
PCV:35.746
WC:9200.857
RC:8557.354
SG_B:45398
SG_C:42378
SG_D:79535
SG_E:96639
RBC Normal 0 or 1:1
PC Normal 0 or 1:0
PC Present 0 or 1:1
BA_Present 0 or 1:0
HTN 0 or 1:1
DM 0 or 1:1
CAD 0 or 1:0
Appet 0 or 1:1
PE 0 or 1:0
ANE 0 or 1:1
```

[25]:
```
Future_Prediction=grid.predict([[age_input,bp_input, al_input, su_input,␣
↪bgr_input, bu_input, sc_input, sod_input, pot_input, hrmo_input, pcv_input,
      wc_input, rc_input, sg_b_input, sg_c_input, sg_d_input, sg_e_input,␣
↪rbc_normal_input, pc_normal_input,
      pcc_present_input, ba_present_input, htn_yes_input, dm_yes_input,␣
↪cad_yes_input,
      appet_yes_input, pe_yes_input, ane_yes_input]])
print("Future_Prediction={}".format(Future_Prediction))
```

```
Future_Prediction=[ True]
```

[26]:
```
#pickle is used to save model creation
import pickle
#Create filename,it is pickle extension so we save .sav
filename="finalized_model_randomforestclassifier.sav"
```

[29]:
```
pickle.dump(classifier, open(filename,"wb"))
```

[31]:
```
#load the model and rb is used for just read
loaded_model=pickle.load(open("finalized_model_randomforestclassifier.
↪sav","rb"))
#we check it and for prediction we can do it
result=loaded_model.predict([[50,80.2,3.0,1.0,150.35436,47.89,4.67654,174.857,4.
↪74256,15.9698,35.746,9200.857,8557.
↪354,45398,42378,79535,96639,1,0,1,0,1,1,0,1,0,1]])
```

[32]:
```
result
```

[32]: array([True])

[ ]: