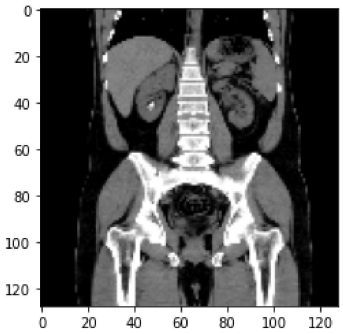
In [360…

```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
import random

DIR='C:/Users/Manohar Vemuri/Desktop/M/project/CT_SCAN'
CATEGORIES=["Normal","Kidney_stone"]
training_data=[]

def train_data():
    for categories in CATEGORIES:
        path=os.path.join(DIR,categories)
        class_num=CATEGORIES.index(categories)
        for img in os.listdir(path):
            try:
                img_array=cv2.imread(os.path.join(path,img))
                new_array=cv2.resize(img_array,(128,128))
                training_data.append([new_array,class_num])
            except Exception as e:
                pass
train_data()
random.shuffle(training_data)
```

In [361…

```
# KIDNEY STONE = 1
# NORMAL = 0
plt.imshow(training_data[3][0],cmap='gray')
print(training_data[3][1])
```

```
1
```



In [362…

```
X=[]
y=[]
for features,labels in training_data:
    X.append(features)
    y.append(labels)
X=np.array(X)
y=np.array(y)
```

In [363…

```
X.shape
```

Out[363…

```
(1609, 128, 128, 3)
```

In [364…

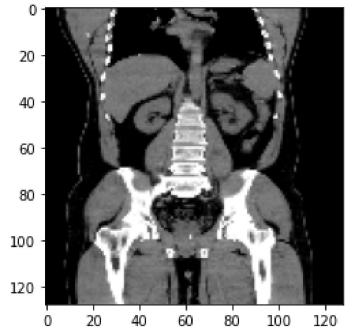
```
from sklearn.model_selection import train_test_split
Xtrain,Xtest,ytrain,ytest=train_test_split(X,y,test_size=0.1,random_state=1)
print(len(Xtrain),len(Xtest),len(ytrain),len(ytest))
Xtrain=np.array(Xtrain)
Xtest=np.array(Xtest)
```

```
1448 161 1448 161
```

In [365…

```
plt.imshow(Xtrain[6],cmap='gray')
print(ytrain[6])
```

```
0
```



In [366…

```
import numpy as np
import cv2 as cv
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.optimizers import SGD
from keras.layers import Dense, Conv2D, Flatten, Convolution2D, Activation
from keras.layers import Dropout, MaxPooling2D
```

In [367…

```
def median_blur(img):
    mb = cv2.medianBlur(img, 1)
    res=hist(mb)
    return res

def hist(img):
    res = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    dst = cv.equalizeHist(res)
    gamma_corrected = np.array(255*(dst / 255) ** 7.5, dtype = 'uint8')
    thres=thresholding_img(gamma_corrected)
    return thres

def thresholding_img(img):
    ret, thresh1 = cv2.threshold(img, 50, 255, cv2.THRESH_BINARY)
    return thresh1

processed_img=[]
for img in Xtrain:
    res=median_blur(img)
    processed_img.append(res)

processed_img=np.array(processed_img)
```

In [368…

```
def build_model(hp):
    model=keras.Sequential([
        keras.layers.Convolution2D(
            filters=hp.Int('conv_1_filter',min_value=32,max_value=128,step=16),
            kernel_size=hp.Choice('conv_1_kernel',values=[3,5]),
            activation='relu',
            input_shape=(128, 128, 1)
        ),
        keras.layers.MaxPooling2D(pool_size=(3, 3)),
        keras.layers.Conv2D(
            filters=hp.Int('conv_2_filter',min_value=32,max_value=128,step=16),
            kernel_size=hp.Choice('conv_2_kernel',values=[3,5]),
            activation='relu'
        ),
        keras.layers.MaxPooling2D(pool_size=(3, 3)),
        keras.layers.Flatten(),
        keras.layers.Dense(
            units=hp.Int('dense_1_units',min_value=32,max_value=128,step=16),
            activation='relu'
        ),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(1,activation='sigmoid')
    ]
    )


    model.compile(optimizer=keras.optimizers.RMSprop(hp.Choice('learning_rate',values=[1e-2,1e-3])),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model
```

In [370…

```
from kerastuner import RandomSearch
```

```
from kerastuner.engine.hyperparameters import HyperParameters
tuner_search=RandomSearch(build_model,objective='val_accuracy',max_trials=10,overwrite=True)
tuner_search.search(processed_img,ytrain,epochs=5,validation_split=0.1)
```

```
Trial 10 Complete [00h 01m 46s]
val_accuracy: 0.7172414064407349

Best val_accuracy So Far: 0.8137931227684021
Total elapsed time: 00h 21m 15s
INFO:tensorflow:Oracle triggered exit
```

In [375…

```
model=tuner_search.get_best_models(num_models=1)[0]
```

In [376…

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 124, 124, 48)      1248

 max_pooling2d (MaxPooling2D  (None, 41, 41, 48)       0
 )

 conv2d_1 (Conv2D)           (None, 39, 39, 112)       48496

 max_pooling2d_1 (MaxPooling  (None, 13, 13, 112)      0
 2D)

 flatten (Flatten)           (None, 18928)             0

 dense (Dense)               (None, 96)                1817184

 dropout (Dropout)           (None, 96)                0

 dense_1 (Dense)             (None, 1)                 97

=================================================================
Total params: 1,867,025
Trainable params: 1,867,025
Non-trainable params: 0
_________________________________________________________________
```

In [377…

```
model.fit(processed_img,ytrain,epochs=30,batch_size=100,verbose=1)
```

```
Epoch 1/30
15/15 [==============================] - 12s 750ms/step - loss: 0.2671 - accuracy: 0.8860
Epoch 2/30
15/15 [==============================] - 13s 872ms/step - loss: 0.1962 - accuracy: 0.9240
Epoch 3/30
15/15 [==============================] - 12s 825ms/step - loss: 0.1602 - accuracy: 0.9351
Epoch 4/30
15/15 [==============================] - 12s 815ms/step - loss: 0.1487 - accuracy: 0.9427
Epoch 5/30
15/15 [==============================] - 13s 849ms/step - loss: 0.1027 - accuracy: 0.9606
Epoch 6/30
15/15 [==============================] - 13s 883ms/step - loss: 0.0854 - accuracy: 0.9675
Epoch 7/30
15/15 [==============================] - 17s 1s/step - loss: 0.0743 - accuracy: 0.9710
Epoch 8/30
15/15 [==============================] - 19s 1s/step - loss: 0.0679 - accuracy: 0.9738
Epoch 9/30
15/15 [==============================] - 17s 1s/step - loss: 0.0701 - accuracy: 0.9696
Epoch 10/30
15/15 [==============================] - 17s 1s/step - loss: 0.0417 - accuracy: 0.9814
Epoch 11/30
15/15 [==============================] - 17s 1s/step - loss: 0.1900 - accuracy: 0.9510
Epoch 12/30
15/15 [==============================] - 16s 1s/step - loss: 0.0322 - accuracy: 0.9910
Epoch 13/30
15/15 [==============================] - 16s 1s/step - loss: 0.0339 - accuracy: 0.9876
Epoch 14/30
15/15 [==============================] - 16s 1s/step - loss: 0.0256 - accuracy: 0.9876
Epoch 15/30
15/15 [==============================] - 16s 1s/step - loss: 0.0371 - accuracy: 0.9841
Epoch 16/30
15/15 [==============================] - 16s 1s/step - loss: 0.0187 - accuracy: 0.9924
Epoch 17/30
15/15 [==============================] - 16s 1s/step - loss: 0.0390 - accuracy: 0.9869
Epoch 18/30
15/15 [==============================] - 16s 1s/step - loss: 0.0294 - accuracy: 0.9903
Epoch 19/30
15/15 [==============================] - 16s 1s/step - loss: 0.0327 - accuracy: 0.9855
Epoch 20/30
15/15 [==============================] - 17s 1s/step - loss: 0.0206 - accuracy: 0.9931
Epoch 21/30
15/15 [==============================] - 17s 1s/step - loss: 0.0122 - accuracy: 0.9924
```

```
Epoch 22/30
15/15 [==============================] - 16s 1s/step - loss: 0.0311 - accuracy: 0.9883
Epoch 23/30
15/15 [==============================] - 16s 1s/step - loss: 0.0117 - accuracy: 0.9965
Epoch 24/30
15/15 [==============================] - 16s 1s/step - loss: 0.0115 - accuracy: 0.9952
Epoch 25/30
15/15 [==============================] - 17s 1s/step - loss: 0.1359 - accuracy: 0.9669
Epoch 26/30
15/15 [==============================] - 17s 1s/step - loss: 0.0256 - accuracy: 0.9917
Epoch 27/30
15/15 [==============================] - 16s 1s/step - loss: 0.0180 - accuracy: 0.9924
Epoch 28/30
15/15 [==============================] - 16s 1s/step - loss: 0.0104 - accuracy: 0.9972
Epoch 29/30
15/15 [==============================] - 16s 1s/step - loss: 0.0168 - accuracy: 0.9959
Epoch 30/30
15/15 [==============================] - 16s 1s/step - loss: 0.0153 - accuracy: 0.9952
```

Out[377…
```
<keras.callbacks.History at 0x1760676fc10>
```

In [ ]:

In [ ]:

In [388…
```
# EVALUATING MODEL PERFORMANCE
```

In [389…
```
Xtest_processed_img=[]
for img in Xtest:
    res=median_blur(img)
    Xtest_processed_img.append(res)

Xtest_processed_img=np.array(Xtest_processed_img)
```

In [402…
```
ypred=model.predict(Xtest_processed_img)
```

In [403…
```
from sklearn.metrics import confusion_matrix
cc=confusion_matrix(ytest,(ypred>0.75)*1)
cc
```

Out[403…
```
array([[68,  9],
       [11, 73]], dtype=int64)
```

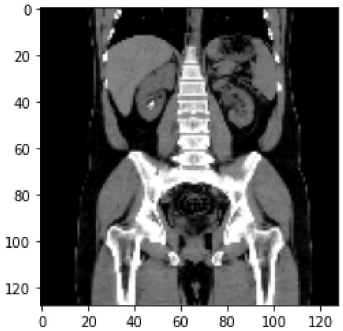
In [406…
```
from sklearn.metrics import accuracy_score
accuracy_score(ytest,(ypred>0.9)*1)
```

Out[406…
```
0.8881987577639752
```

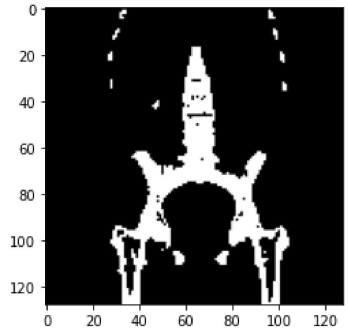
In [ ]:

In [393…
```
# TEST INDIVIDUAL IMAGE
```

In [394…
```
img_array=cv2.imread('test.png')
new_array=cv2.resize(img_array,(128,128))
plt.imshow(new_array,cmap='gray')
res=median_blur(new_array)
```



In [395…
```
plt.imshow(res,cmap='gray')
plt.show()
```



In [396…
```
res=res.reshape(-1,128,128,1)
ypred=model.predict(res)
```

In [397…
```
(ypred>0.5)*1
```

Out[397…
```
array([[1]])
```

In [350…

In [ ]:

In [ ]:

In [ ]: