SAMPLE CODING

GIRD SEARCH OPTIMIZATION

```
import numpy
import keras
import pandas as pd
from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import HalvingGridSearchCV
from keras.models import Sequential
from keras.layers import Dense
import warnings
warnings.filterwarnings('ignore')
from keras.wrappers.scikit_learn import KerasClassifier
import numpy as np
import matplotlib.pyplot as plt
import numpy as np

data = pd.read_csv('/content/drive/MyDrive/dataset-org.csv')

data.head()

X = data.iloc[:,0:178]
Y = data.iloc[:,178]

def create_my_model(optimizer='adam', activationL='relu', neurons=0.01):
  mymodel=Sequential()
  mymodel.add(Dense(16,input_dim=178,activation='relu'))
  mymodel.add(Dense(neurons, activation=activationL))
  mymodel.add(Dense(1,activation='sigmoid'))

  mymodel.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
  return mymodel

model=KerasClassifier(build_fn=create_my_model)

batchSize=[10,20,40,60,80,100]
epochs=[10,30,50]
optimizer=['SGD', 'Adadelta', 'RMSprop', 'Adagrad', 'Adam']
activationL=['relu', 'sigmoid', 'selu', 'exponential', 'LeakyReLU','elu','softplus', 'softsign', 'tanh']
neurons=(10,100)

parameter_grid=dict(batch_size=batchSize,epochs=epochs,optimizer=optimizer,activationL=
activationL, neurons=neurons)

mygrid =HalvingGridSearchCV(estimator=model,param_grid= parameter_grid, n_jobs=-1,
cv=3)
```

```
grid_result =mygrid.fit(X,Y)

print("Best:%f using %s"%(grid_result.best_score_,grid_result.best_params_))
```

GIRD SEARCH INTO ANN

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split

d1=pd.read_csv('/content/drive/MyDrive/dataset-org.csv',header=None)
print(d1)

X=pd.DataFrame(d1.iloc[:,:-1].values)
Y=d1.iloc[:,-1].values

X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state = 0 )

print(X_train)

print(Y_test)

print(Y_train)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

import tensorflow as tf

import keras  as ke

from keras.models import Sequential
from keras.layers import Dense

classifier = Sequential()

classifier.add(Dense(units= 100, kernel_initializer = 'uniform', activation = 'exponential', input_dim = 178))

classifier.add(Dense(units = 100,kernel_initializer = 'uniform', activation = 'exponential'))

classifier.add(Dense(units = 1,kernel_initializer   = 'uniform', activation = 'sigmoid'))

classifier.compile(optimizer = 'rmsprop', loss = 'binary_crossentropy', metrics = ['accuracy'])
classifier.fit(X_train, Y_train, batch_size = 60, epochs = 50)

y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)

y_pred

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(Y_test, y_pred)
print(cm)
accuracy_score(Y_test,y_pred)

sensitivity1=(1854/(1854+1))
print(sensitivity1)

precision1=(1854/(1854+446))
print(precision1)

recall1=(1854/(1854+0))
print(recall1)

F1Score=2*(precision1*recall1/(precision1+recall1))
print(F1Score)
```