In [1]:

```
import sys
import pandas as pd
import numpy as np
import sklearn
import keras

print(sys.version)
print(pd.__version__)
print(np.__version__)
print(sklearn.__version__)
print(keras.__version__)
```

```
Using TensorFlow backend.

3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
0.24.2
1.18.1
0.22.1
2.3.1
```

In [2]:

```
# import the uci pima indians diabetes dataset
url = "diabetes.csv"
names = ['n_pregnant', 'glucose_concentration', 'blood_pressuer (mm Hg)', 'skin_thickne
ss (mm)', 'serum_insulin (mu U/ml)',
        'BMI', 'pedigree_function', 'age', 'class']
df = pd.read_csv(url, names = names)
```

In [3]:

```
df.describe()
```

Out[3]:

	n_pregnant	glucose_concentration	blood_pressuer (mm Hg)	skin_thickness (mm)	serum_insulin (mu U/ml)	
count	768.000000	768.000000	768.000000	768.000000	768.000000	76
mean	3.845052	120.894531	69.105469	20.536458	79.799479	3
std	3.369578	31.972618	19.355807	15.952218	115.244002	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	2
50%	3.000000	117.000000	72.000000	23.000000	30.500000	3
75%	6.000000	140.250000	80.000000	32.000000	127.250000	3
max	17.000000	199.000000	122.000000	99.000000	846.000000	6

In [4]:

```
df[df['glucose_concentration']==0]
```

Out[4]:

	n_pregnant	glucose_concentration	blood_pressuer (mm Hg)	skin_thickness (mm)	serum_insulin (mu U/ml)	BMI
75	1	0	48	20	0	24.7
182	1	0	74	20	23	27.7
342	1	0	68	35	0	32.0
349	5	0	80	32	0	41.0
502	6	0	68	41	0	39.0

In [5]:

```
columns = ['glucose_concentration', 'blood_pressuer (mm Hg)', 'skin_thickness (mm)', 's
erum_insulin (mu U/ml)', 'BMI']
for col in columns:
    df[col].replace(0,np.NaN,inplace=True)
```

In [6]:

```
# Drop the rows with missing values
df.dropna(inplace=True)

# summarize the number of rows and columns in df
df.describe()
```

Out[6]:

	n_pregnant	glucose_concentration	blood_pressuer (mm Hg)	skin_thickness (mm)	serum_insulin (mu U/ml)	
count	392.000000	392.000000	392.000000	392.000000	392.000000	39
mean	3.301020	122.627551	70.663265	29.145408	156.056122	3
std	3.211424	30.860781	12.496092	10.516424	118.841690	
min	0.000000	56.000000	24.000000	7.000000	14.000000	1
25%	1.000000	99.000000	62.000000	21.000000	76.750000	2
50%	2.000000	119.000000	70.000000	29.000000	125.500000	3
75%	5.000000	143.000000	78.000000	37.000000	190.000000	3
max	17.000000	198.000000	110.000000	63.000000	846.000000	6

In [7]:

```
dataset = df.values
print(dataset.shape)
```

```
(392, 9)
```

In [8]:

```
col = ['n_pregnant', 'glucose_concentration', 'blood_pressuer (mm Hg)', 'skin_thickness
(mm)', 'serum_insulin (mu U/ml)',
       'BMI', 'pedigree_function', 'age']
target='class'
X=df[col]
Y=df[target].astype(int)
```

In [9]:

```
print(X.shape)
print(Y.shape)
#print(X)
```

```
(392, 8)
(392,)
```

In [10]:

```
# Normalize the data using sklearn StandardScaler
from sklearn.preprocessing import StandardScaler

scaler=StandardScaler().fit(X)
print(scaler)
```

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [11]:

```
# Transform and display the training data
X_standardized = scaler.transform(X)
data = pd.DataFrame(X_standardized)
data.describe()
```

Out[11]:

	0	1	2	3	4	
count	3.920000e+02	3.920000e+02	3.920000e+02	3.920000e+02	3.920000e+02	3.920000e
mean	-4.021726e-17	3.129583e-17	-4.641624e-16	1.042250e-16	6.485742e-17	1.543550e
std	1.001278e+00	1.001278e+00	1.001278e+00	1.001278e+00	1.001278e+00	1.001278e
min	-1.029213e+00	-2.161731e+00	-3.739001e+00	-2.108484e+00	-1.196867e+00	-2.120941e
25%	-7.174265e-01	-7.665958e-01	-6.941640e-01	-7.755315e-01	-6.681786e-01	-6.676780e
50%	-4.056403e-01	-1.176959e-01	-5.314565e-02	-1.384444e-02	-2.574448e-01	1.621036e
75%	5.297185e-01	6.609841e-01	5.878727e-01	7.478426e-01	2.859877e-01	5.718696e
max	4.271153e+00	2.445459e+00	3.151946e+00	3.223325e+00	5.812990e+00	4.846172e

In [12]:

```
#import necessary packages
from sklearn.model_selection import GridSearchCV, KFold
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.optimizers import Adam
```

In [13]:

```
#Start defining the model
def create_model():
    model=Sequential()
    model.add(Dense(8,input_dim=8,kernel_initializer='normal',activation='relu'))
    model.add(Dense(4,input_dim=8,kernel_initializer='normal',activation='relu'))
    model.add(Dense(1,activation='sigmoid'))

    adam = Adam(lr = 0.01)
    model.compile(loss = 'binary_crossentropy', optimizer = adam, metrics = ['accuracy'
])
    return model
```

In [14]:

```
model = create_model()
print(model.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 8)                 72
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 36
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 5
=================================================================
Total params: 113
Trainable params: 113
Non-trainable params: 0
_________________________________________________________________
None
```

In [15]:

```
#define a random seed
seed=6
np.random.seed(seed)
#Start defining the model
def create_model():
    model=Sequential()
    model.add(Dense(8,input_dim=8,kernel_initializer='normal',activation='relu'))
    model.add(Dense(4,input_dim=8,kernel_initializer='normal',activation='relu'))
    model.add(Dense(1,activation='sigmoid'))

    adam = Adam(lr = 0.01)
    model.compile(loss = 'binary_crossentropy', optimizer = adam, metrics = ['accuracy'
])
    return model

#create model
model=KerasClassifier(build_fn=create_model, verbose=0)

# Define the grid search parameters
batch_size = [10,20,30,40]
epochs = [10, 50, 100]

# make a dictionary of the grid search parameters
param_grid = dict(batch_size=batch_size, epochs=epochs)

# build and fit the GridSearchCV
grid = GridSearchCV(estimator=model, param_grid=param_grid, cv = KFold(random_state=see
d), verbose=10)
grid_result = grid.fit(X_standardized, Y)

# summarize the results
print("Best: {0}, using {1}".format(grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('{0} ({1}) with: {2}'.format(mean, stdev, param))
```

```
C:\Users\balsh\anaconda3\lib\site-packages\sklearn\model_selection\_split.
py:296: FutureWarning: Setting a random_state has no effect since shuffle
is False. This will raise an error in 0.24. You should leave random_state
to its default (None), or set shuffle=True.
  FutureWarning
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wo
rkers.

Fitting 5 folds for each of 12 candidates, totalling 60 fits
[CV] batch_size=10, epochs=10 ........................................
[CV] ............ batch_size=10, epochs=10, score=0.722, total=   1.8s
[CV] batch_size=10, epochs=10 ........................................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.7s remaining:
0.0s

[CV] ............ batch_size=10, epochs=10, score=0.582, total=   1.5s
[CV] batch_size=10, epochs=10 ........................................

[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    3.2s remaining:
0.0s

[CV] ............ batch_size=10, epochs=10, score=0.808, total=   1.8s
[CV] batch_size=10, epochs=10 ........................................

[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    5.0s remaining:
0.0s

[CV] ............ batch_size=10, epochs=10, score=0.795, total=   1.7s
[CV] batch_size=10, epochs=10 ........................................

[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    6.8s remaining:
0.0s

[CV] ............ batch_size=10, epochs=10, score=0.846, total=   1.6s
[CV] batch_size=10, epochs=50 ........................................

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    8.4s remaining:
0.0s

[CV] ............ batch_size=10, epochs=50, score=0.722, total=   4.4s
[CV] batch_size=10, epochs=50 ........................................

[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:   12.8s remaining:
0.0s

[CV] ............ batch_size=10, epochs=50, score=0.709, total=   4.4s
[CV] batch_size=10, epochs=50 ........................................

[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:   17.2s remaining:
0.0s

[CV] ............ batch_size=10, epochs=50, score=0.821, total=   4.5s
[CV] batch_size=10, epochs=50 ........................................

[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:   21.7s remaining:
0.0s

[CV] ............ batch_size=10, epochs=50, score=0.846, total=   4.4s
[CV] batch_size=10, epochs=50 ........................................

[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:   26.1s remaining:
0.0s
```

```
[CV] ............ batch_size=10, epochs=50, score=0.859, total=   4.0s
[CV] batch_size=10, epochs=100 ....................................
[CV] ............ batch_size=10, epochs=100, score=0.772, total=   8.4s
[CV] batch_size=10, epochs=100 ....................................
[CV] ............ batch_size=10, epochs=100, score=0.671, total=   8.2s
[CV] batch_size=10, epochs=100 ....................................
[CV] ............ batch_size=10, epochs=100, score=0.795, total=   7.4s
[CV] batch_size=10, epochs=100 ....................................
[CV] ............ batch_size=10, epochs=100, score=0.833, total=   7.7s
[CV] batch_size=10, epochs=100 ....................................
[CV] ............ batch_size=10, epochs=100, score=0.756, total=   7.8s
[CV] batch_size=20, epochs=10 .....................................
[CV] ............ batch_size=20, epochs=10, score=0.772, total=   1.3s
[CV] batch_size=20, epochs=10 .....................................
[CV] ............ batch_size=20, epochs=10, score=0.582, total=   1.1s
[CV] batch_size=20, epochs=10 .....................................
[CV] ............ batch_size=20, epochs=10, score=0.808, total=   1.2s
[CV] batch_size=20, epochs=10 .....................................
[CV] ............ batch_size=20, epochs=10, score=0.808, total=   1.1s
[CV] batch_size=20, epochs=10 .....................................
[CV] ............ batch_size=20, epochs=10, score=0.821, total=   1.3s
[CV] batch_size=20, epochs=50 .....................................
[CV] ............ batch_size=20, epochs=50, score=0.759, total=   2.5s
[CV] batch_size=20, epochs=50 .....................................
[CV] ............ batch_size=20, epochs=50, score=0.620, total=   2.6s
[CV] batch_size=20, epochs=50 .....................................
[CV] ............ batch_size=20, epochs=50, score=0.833, total=   2.6s
[CV] batch_size=20, epochs=50 .....................................
[CV] ............ batch_size=20, epochs=50, score=0.872, total=   2.6s
[CV] batch_size=20, epochs=50 .....................................
[CV] ............ batch_size=20, epochs=50, score=0.859, total=   2.5s
[CV] batch_size=20, epochs=100 ....................................
[CV] ............ batch_size=20, epochs=100, score=0.759, total=   4.4s
[CV] batch_size=20, epochs=100 ....................................
[CV] ............ batch_size=20, epochs=100, score=0.646, total=   4.2s
[CV] batch_size=20, epochs=100 ....................................
[CV] ............ batch_size=20, epochs=100, score=0.795, total=   4.5s
[CV] batch_size=20, epochs=100 ....................................
[CV] ............ batch_size=20, epochs=100, score=0.782, total=   4.4s
[CV] batch_size=20, epochs=100 ....................................
[CV] ............ batch_size=20, epochs=100, score=0.833, total=   4.5s
[CV] batch_size=30, epochs=10 .....................................
[CV] ............ batch_size=30, epochs=10, score=0.734, total=   1.1s
[CV] batch_size=30, epochs=10 .....................................
[CV] ............ batch_size=30, epochs=10, score=0.608, total=   1.1s
[CV] batch_size=30, epochs=10 .....................................
[CV] ............ batch_size=30, epochs=10, score=0.808, total=   1.2s
[CV] batch_size=30, epochs=10 .....................................
[CV] ............ batch_size=30, epochs=10, score=0.808, total=   1.1s
[CV] batch_size=30, epochs=10 .....................................
[CV] ............ batch_size=30, epochs=10, score=0.821, total=   1.0s
[CV] batch_size=30, epochs=50 .....................................
[CV] ............ batch_size=30, epochs=50, score=0.709, total=   2.0s
[CV] batch_size=30, epochs=50 .....................................
[CV] ............ batch_size=30, epochs=50, score=0.658, total=   2.0s
[CV] batch_size=30, epochs=50 .....................................
[CV] ............ batch_size=30, epochs=50, score=0.859, total=   2.3s
[CV] batch_size=30, epochs=50 .....................................
[CV] ............ batch_size=30, epochs=50, score=0.846, total=   2.0s
[CV] batch_size=30, epochs=50 .....................................
[CV] ............ batch_size=30, epochs=50, score=0.846, total=   2.0s
```

```
[CV] batch_size=30, epochs=100 ......................................
[CV] ........... batch_size=30, epochs=100, score=0.722, total=   3.2s
[CV] batch_size=30, epochs=100 ......................................
[CV] ........... batch_size=30, epochs=100, score=0.671, total=   3.3s
[CV] batch_size=30, epochs=100 ......................................
[CV] ........... batch_size=30, epochs=100, score=0.821, total=   3.5s
[CV] batch_size=30, epochs=100 ......................................
[CV] ........... batch_size=30, epochs=100, score=0.859, total=   3.1s
[CV] batch_size=30, epochs=100 ......................................
[CV] ........... batch_size=30, epochs=100, score=0.833, total=   3.4s
[CV] batch_size=40, epochs=10 .......................................
[CV] ............ batch_size=40, epochs=10, score=0.785, total=   1.0s
[CV] batch_size=40, epochs=10 .......................................
[CV] ............ batch_size=40, epochs=10, score=0.620, total=   1.0s
[CV] batch_size=40, epochs=10 .......................................
[CV] ............ batch_size=40, epochs=10, score=0.795, total=   1.2s
[CV] batch_size=40, epochs=10 .......................................
[CV] ............ batch_size=40, epochs=10, score=0.846, total=   1.0s
[CV] batch_size=40, epochs=10 .......................................
[CV] ............ batch_size=40, epochs=10, score=0.859, total=   1.2s
[CV] batch_size=40, epochs=50 .......................................
[CV] ............ batch_size=40, epochs=50, score=0.785, total=   1.9s
[CV] batch_size=40, epochs=50 .......................................
[CV] ............ batch_size=40, epochs=50, score=0.633, total=   1.7s
[CV] batch_size=40, epochs=50 .......................................
[CV] ............ batch_size=40, epochs=50, score=0.833, total=   1.7s
[CV] batch_size=40, epochs=50 .......................................
[CV] ............ batch_size=40, epochs=50, score=0.821, total=   1.9s
[CV] batch_size=40, epochs=50 .......................................
[CV] ............ batch_size=40, epochs=50, score=0.846, total=   1.7s
[CV] batch_size=40, epochs=100 ......................................
[CV] ........... batch_size=40, epochs=100, score=0.722, total=   2.6s
[CV] batch_size=40, epochs=100 ......................................
[CV] ........... batch_size=40, epochs=100, score=0.658, total=   2.7s
[CV] batch_size=40, epochs=100 ......................................
[CV] ........... batch_size=40, epochs=100, score=0.769, total=   2.6s
[CV] batch_size=40, epochs=100 ......................................
[CV] ........... batch_size=40, epochs=100, score=0.821, total=   2.8s
[CV] batch_size=40, epochs=100 ......................................
[CV] ........... batch_size=40, epochs=100, score=0.859, total=   2.7s

[Parallel(n_jobs=1)]: Done  60 out of  60 | elapsed:  2.8min finished
```

```
Best: 0.7912041544914246, using {'batch_size': 10, 'epochs': 50}
0.7505030870437622 (0.09330364448009798) with: {'batch_size': 10, 'epoch
s': 10}
0.7912041544914246 (0.06341564947695214) with: {'batch_size': 10, 'epoch
s': 50}
0.7655306696891785 (0.053930915597060584) with: {'batch_size': 10, 'epoch
s': 100}
0.7580655574798584 (0.08935860875152117) with: {'batch_size': 20, 'epoch
s': 10}
0.7887698650360108 (0.09280377518135459) with: {'batch_size': 20, 'epoch
s': 50}
0.763063931465149 (0.06344715185877535) with: {'batch_size': 20, 'epochs':
100}
0.7555339097976684 (0.08000663660471047) with: {'batch_size': 30, 'epoch
s': 10}
0.783674132823944 (0.08344023121060123) with: {'batch_size': 30, 'epochs':
50}
0.781045114994049 (0.07217075299611107) with: {'batch_size': 30, 'epochs':
100}
0.7810126543045044 (0.08528952648905208) with: {'batch_size': 40, 'epoch
s': 10}
0.7835443019866943 (0.078048000929465) with: {'batch_size': 40, 'epochs':
50}
0.7656929612159729 (0.07102004888718165) with: {'batch_size': 40, 'epoch
s': 100}
```

In [16]:

```
# import necessary packages
from keras.layers import Dropout
# Define a random seed
seed = 6
np.random.seed(seed)
epochs=50
batch_size=10
# Start defining the model
def create_model(learn_rate, dropout_rate):
    # Create model
    model = Sequential()
    model.add(Dense(8, input_dim = 8, kernel_initializer = 'normal', activation = 'rel
u'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(4, input_dim = 8, kernel_initializer = 'normal', activation = 'rel
u'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(1, activation = 'sigmoid'))

    # Compile the model
    adam = Adam(lr = learn_rate)
    model.compile(loss = 'binary_crossentropy', optimizer = adam, metrics = ['accuracy'
])
    return model

# Create the model
model = KerasClassifier(build_fn = create_model, epochs=epochs, batch_size=batch_size,
verbose = 0) # This comes from the previous best

# define the grid search parameters
learn_rate = [0.0001, 0.001, 0.01, 0.1]
dropout_rate = [0.0, 0.1, 0.2, 0.3]

# make a dictionary of the grid search parameters
param_grid = dict(learn_rate=learn_rate, dropout_rate=dropout_rate)

# build and fit the GridSearchCV
grid = GridSearchCV(estimator=model, param_grid=param_grid, cv = KFold(random_state=see
d), verbose=10)
grid_result = grid.fit(X_standardized, Y)

# summarize the results
print("Best: {0}, using {1}".format(grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('{0} ({1}) with: {2}'.format(mean, stdev, param))
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] dropout_rate=0.0, learn_rate=0.0001 .............................

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wo
rkers.

[CV] . dropout_rate=0.0, learn_rate=0.0001, score=0.759, total=   4.6s
[CV] dropout_rate=0.0, learn_rate=0.0001 .............................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    4.5s remaining:
0.0s

[CV] . dropout_rate=0.0, learn_rate=0.0001, score=0.620, total=   4.2s
[CV] dropout_rate=0.0, learn_rate=0.0001 .............................

[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    8.7s remaining:
0.0s

[CV] . dropout_rate=0.0, learn_rate=0.0001, score=0.821, total=   4.2s
[CV] dropout_rate=0.0, learn_rate=0.0001 .............................

[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:   13.0s remaining:
0.0s

[CV] . dropout_rate=0.0, learn_rate=0.0001, score=0.769, total=   4.5s
[CV] dropout_rate=0.0, learn_rate=0.0001 .............................

[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:   17.5s remaining:
0.0s

[CV] . dropout_rate=0.0, learn_rate=0.0001, score=0.769, total=   4.2s
[CV] dropout_rate=0.0, learn_rate=0.001 ..............................

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:   21.7s remaining:
0.0s

[CV] .. dropout_rate=0.0, learn_rate=0.001, score=0.759, total=   4.3s
[CV] dropout_rate=0.0, learn_rate=0.001 ..............................

[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:   26.0s remaining:
0.0s

[CV] .. dropout_rate=0.0, learn_rate=0.001, score=0.633, total=   4.4s
[CV] dropout_rate=0.0, learn_rate=0.001 ..............................

[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:   30.4s remaining:
0.0s

[CV] .. dropout_rate=0.0, learn_rate=0.001, score=0.808, total=   4.3s
[CV] dropout_rate=0.0, learn_rate=0.001 ..............................


[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:   34.7s remaining:
0.0s

[CV] .. dropout_rate=0.0, learn_rate=0.001, score=0.808, total=   4.4s
[CV] dropout_rate=0.0, learn_rate=0.001 ..............................

[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:   39.1s remaining:
0.0s
```

```
[CV] .. dropout_rate=0.0, learn_rate=0.001, score=0.833, total=   4.6s
[CV] dropout_rate=0.0, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.0, learn_rate=0.01, score=0.747, total=   4.3s
[CV] dropout_rate=0.0, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.0, learn_rate=0.01, score=0.671, total=   4.3s
[CV] dropout_rate=0.0, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.0, learn_rate=0.01, score=0.769, total=   4.3s
[CV] dropout_rate=0.0, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.0, learn_rate=0.01, score=0.833, total=   4.3s
[CV] dropout_rate=0.0, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.0, learn_rate=0.01, score=0.859, total=   4.3s
[CV] dropout_rate=0.0, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.0, learn_rate=0.1, score=0.734, total=   4.6s
[CV] dropout_rate=0.0, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.0, learn_rate=0.1, score=0.658, total=   4.2s
[CV] dropout_rate=0.0, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.0, learn_rate=0.1, score=0.705, total=   4.2s
[CV] dropout_rate=0.0, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.0, learn_rate=0.1, score=0.821, total=   4.4s
[CV] dropout_rate=0.0, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.0, learn_rate=0.1, score=0.795, total=   4.4s
[CV] dropout_rate=0.1, learn_rate=0.0001 ............................
[CV] . dropout_rate=0.1, learn_rate=0.0001, score=0.684, total=   5.0s
[CV] dropout_rate=0.1, learn_rate=0.0001 ............................
[CV] . dropout_rate=0.1, learn_rate=0.0001, score=0.570, total=   5.3s
[CV] dropout_rate=0.1, learn_rate=0.0001 ............................
[CV] . dropout_rate=0.1, learn_rate=0.0001, score=0.718, total=   5.1s
[CV] dropout_rate=0.1, learn_rate=0.0001 ............................
[CV] . dropout_rate=0.1, learn_rate=0.0001, score=0.744, total=   5.0s
[CV] dropout_rate=0.1, learn_rate=0.0001 ............................
[CV] . dropout_rate=0.1, learn_rate=0.0001, score=0.679, total=   5.1s
[CV] dropout_rate=0.1, learn_rate=0.001 .............................
[CV] .. dropout_rate=0.1, learn_rate=0.001, score=0.759, total=   5.1s
[CV] dropout_rate=0.1, learn_rate=0.001 .............................
[CV] .. dropout_rate=0.1, learn_rate=0.001, score=0.633, total=   5.4s
[CV] dropout_rate=0.1, learn_rate=0.001 .............................
[CV] .. dropout_rate=0.1, learn_rate=0.001, score=0.821, total=   5.3s
[CV] dropout_rate=0.1, learn_rate=0.001 .............................
[CV] .. dropout_rate=0.1, learn_rate=0.001, score=0.833, total=   5.2s
[CV] dropout_rate=0.1, learn_rate=0.001 .............................
[CV] .. dropout_rate=0.1, learn_rate=0.001, score=0.833, total=   5.1s
[CV] dropout_rate=0.1, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.1, learn_rate=0.01, score=0.772, total=   5.3s
[CV] dropout_rate=0.1, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.1, learn_rate=0.01, score=0.658, total=   5.4s
[CV] dropout_rate=0.1, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.1, learn_rate=0.01, score=0.821, total=   5.1s
[CV] dropout_rate=0.1, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.1, learn_rate=0.01, score=0.885, total=   5.1s
[CV] dropout_rate=0.1, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.1, learn_rate=0.01, score=0.808, total=   5.2s
[CV] dropout_rate=0.1, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.1, learn_rate=0.1, score=0.658, total=   5.2s
[CV] dropout_rate=0.1, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.1, learn_rate=0.1, score=0.570, total=   5.0s
[CV] dropout_rate=0.1, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.1, learn_rate=0.1, score=0.782, total=   5.3s
[CV] dropout_rate=0.1, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.1, learn_rate=0.1, score=0.769, total=   5.0s
[CV] dropout_rate=0.1, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.1, learn_rate=0.1, score=0.731, total=   4.9s
```

```
[CV] dropout_rate=0.2, learn_rate=0.0001 ..........................
[CV] . dropout_rate=0.2, learn_rate=0.0001, score=0.759, total=   4.9s
[CV] dropout_rate=0.2, learn_rate=0.0001 ..........................
[CV] . dropout_rate=0.2, learn_rate=0.0001, score=0.633, total=   5.1s
[CV] dropout_rate=0.2, learn_rate=0.0001 ..........................
[CV] . dropout_rate=0.2, learn_rate=0.0001, score=0.705, total=   5.2s
[CV] dropout_rate=0.2, learn_rate=0.0001 ..........................
[CV] . dropout_rate=0.2, learn_rate=0.0001, score=0.833, total=   5.3s
[CV] dropout_rate=0.2, learn_rate=0.0001 ..........................
[CV] . dropout_rate=0.2, learn_rate=0.0001, score=0.705, total=   4.9s
[CV] dropout_rate=0.2, learn_rate=0.001 ...........................
[CV] .. dropout_rate=0.2, learn_rate=0.001, score=0.759, total=   5.1s
[CV] dropout_rate=0.2, learn_rate=0.001 ...........................
[CV] .. dropout_rate=0.2, learn_rate=0.001, score=0.620, total=   5.2s
[CV] dropout_rate=0.2, learn_rate=0.001 ...........................
[CV] .. dropout_rate=0.2, learn_rate=0.001, score=0.821, total=   5.0s
[CV] dropout_rate=0.2, learn_rate=0.001 ...........................
[CV] .. dropout_rate=0.2, learn_rate=0.001, score=0.859, total=   5.0s
[CV] dropout_rate=0.2, learn_rate=0.001 ...........................
[CV] .. dropout_rate=0.2, learn_rate=0.001, score=0.846, total=   5.2s
[CV] dropout_rate=0.2, learn_rate=0.01 ............................
[CV] ... dropout_rate=0.2, learn_rate=0.01, score=0.747, total=   5.5s
[CV] dropout_rate=0.2, learn_rate=0.01 ............................
[CV] ... dropout_rate=0.2, learn_rate=0.01, score=0.658, total=   4.9s
[CV] dropout_rate=0.2, learn_rate=0.01 ............................
[CV] ... dropout_rate=0.2, learn_rate=0.01, score=0.833, total=   5.1s
[CV] dropout_rate=0.2, learn_rate=0.01 ............................
[CV] ... dropout_rate=0.2, learn_rate=0.01, score=0.833, total=   5.1s
[CV] dropout_rate=0.2, learn_rate=0.01 ............................
[CV] ... dropout_rate=0.2, learn_rate=0.01, score=0.795, total=   4.9s
[CV] dropout_rate=0.2, learn_rate=0.1 .............................
[CV] .... dropout_rate=0.2, learn_rate=0.1, score=0.646, total=   5.0s
[CV] dropout_rate=0.2, learn_rate=0.1 .............................
[CV] .... dropout_rate=0.2, learn_rate=0.1, score=0.646, total=   5.1s
[CV] dropout_rate=0.2, learn_rate=0.1 .............................
[CV] .... dropout_rate=0.2, learn_rate=0.1, score=0.769, total=   5.4s
[CV] dropout_rate=0.2, learn_rate=0.1 .............................
[CV] .... dropout_rate=0.2, learn_rate=0.1, score=0.808, total=   4.8s
[CV] dropout_rate=0.2, learn_rate=0.1 .............................
[CV] .... dropout_rate=0.2, learn_rate=0.1, score=0.808, total=   5.0s
[CV] dropout_rate=0.3, learn_rate=0.0001 ..........................
[CV] . dropout_rate=0.3, learn_rate=0.0001, score=0.722, total=   5.0s
[CV] dropout_rate=0.3, learn_rate=0.0001 ..........................
[CV] . dropout_rate=0.3, learn_rate=0.0001, score=0.595, total=   4.9s
[CV] dropout_rate=0.3, learn_rate=0.0001 ..........................
[CV] . dropout_rate=0.3, learn_rate=0.0001, score=0.705, total=   5.0s
[CV] dropout_rate=0.3, learn_rate=0.0001 ..........................
[CV] . dropout_rate=0.3, learn_rate=0.0001, score=0.808, total=   5.1s
[CV] dropout_rate=0.3, learn_rate=0.0001 ..........................
[CV] . dropout_rate=0.3, learn_rate=0.0001, score=0.731, total=   5.2s
[CV] dropout_rate=0.3, learn_rate=0.001 ...........................
[CV] .. dropout_rate=0.3, learn_rate=0.001, score=0.734, total=   4.9s
[CV] dropout_rate=0.3, learn_rate=0.001 ...........................
[CV] .. dropout_rate=0.3, learn_rate=0.001, score=0.582, total=   5.1s
[CV] dropout_rate=0.3, learn_rate=0.001 ...........................
[CV] .. dropout_rate=0.3, learn_rate=0.001, score=0.821, total=   5.1s
[CV] dropout_rate=0.3, learn_rate=0.001 ...........................
[CV] .. dropout_rate=0.3, learn_rate=0.001, score=0.808, total=   5.0s
[CV] dropout_rate=0.3, learn_rate=0.001 ...........................
[CV] .. dropout_rate=0.3, learn_rate=0.001, score=0.859, total=   6.2s
[CV] dropout_rate=0.3, learn_rate=0.01 ............................
```

```
[CV] ... dropout_rate=0.3, learn_rate=0.01, score=0.747, total=   5.6s
[CV] dropout_rate=0.3, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.3, learn_rate=0.01, score=0.646, total=   5.8s
[CV] dropout_rate=0.3, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.3, learn_rate=0.01, score=0.782, total=   6.3s
[CV] dropout_rate=0.3, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.3, learn_rate=0.01, score=0.795, total=   5.3s
[CV] dropout_rate=0.3, learn_rate=0.01 ..............................
[CV] ... dropout_rate=0.3, learn_rate=0.01, score=0.821, total=   5.5s
[CV] dropout_rate=0.3, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.3, learn_rate=0.1, score=0.646, total=   5.0s
[CV] dropout_rate=0.3, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.3, learn_rate=0.1, score=0.595, total=   5.1s
[CV] dropout_rate=0.3, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.3, learn_rate=0.1, score=0.795, total=   4.9s
[CV] dropout_rate=0.3, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.3, learn_rate=0.1, score=0.744, total=   4.9s
[CV] dropout_rate=0.3, learn_rate=0.1 ...............................
[CV] .... dropout_rate=0.3, learn_rate=0.1, score=0.679, total=   4.9s

[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:  6.6min finished

Best: 0.78864004611969, using {'dropout_rate': 0.1, 'learn_rate': 0.01}
0.7477442383766174 (0.06724676745381974) with: {'dropout_rate': 0.0, 'lear
n_rate': 0.0001}
0.768224585056305 (0.07175017418186987) with: {'dropout_rate': 0.0, 'learn
_rate': 0.001}
0.7758520007133484 (0.06650874261352188) with: {'dropout_rate': 0.0, 'lear
n_rate': 0.01}
0.7425835847854614 (0.05898556932094381) with: {'dropout_rate': 0.0, 'lear
n_rate': 0.1}
0.6788380384445191 (0.05945338826029287) with: {'dropout_rate': 0.1, 'lear
n_rate': 0.0001}
0.7759168982505799 (0.07655375483984386) with: {'dropout_rate': 0.1, 'lear
n_rate': 0.001}
0.78864004611969 (0.07465831638149004) with: {'dropout_rate': 0.1, 'learn_
rate': 0.01}
0.7019798755645752 (0.07898891947698417) with: {'dropout_rate': 0.1, 'lear
n_rate': 0.1}
0.7271989464759827 (0.06659103905495732) with: {'dropout_rate': 0.2, 'lear
n_rate': 0.0001}
0.7810775637626648 (0.08739374146471103) with: {'dropout_rate': 0.2, 'lear
n_rate': 0.001}
0.7733203530311584 (0.0657499912847478) with: {'dropout_rate': 0.2, 'learn
_rate': 0.01}
0.7351509213447571 (0.0744789303276814) with: {'dropout_rate': 0.2, 'learn
_rate': 0.1}
0.7120090842247009 (0.06833877353815908) with: {'dropout_rate': 0.3, 'lear
n_rate': 0.0001}
0.7607270359992981 (0.09796058824673763) with: {'dropout_rate': 0.3, 'lear
n_rate': 0.001}
0.7579681992530822 (0.06100996481663896) with: {'dropout_rate': 0.3, 'lear
n_rate': 0.01}
0.6916910171508789 (0.07067851744494327) with: {'dropout_rate': 0.3, 'lear
n_rate': 0.1}
```

In [17]:

```
# Define a random seed
seed = 6
np.random.seed(seed)
learn_rate = 0.01
# Start defining the model
def create_model(activation, init):
    # Create model
    model = Sequential()
    model.add(Dense(8, input_dim = 8, kernel_initializer = init, activation = activatio
n))
    #model.add(Dropout(dropout_rate))
    model.add(Dense(4, input_dim = 8, kernel_initializer = init, activation = activatio
n))
    #model.add(Dropout(dropout_rate))
    model.add(Dense(1, activation = 'sigmoid'))

    # Compile the model
    adam = Adam(lr = learn_rate)
    model.compile(loss = 'binary_crossentropy', optimizer = adam, metrics = ['accuracy'
])
    return model

# Create the model
model = KerasClassifier(build_fn = create_model, epochs=epochs, batch_size=batch_size,
verbose = 0) # This comes from the previous best

# define the grid search parameters
activations = ['softmax', 'relu', 'tanh', 'linear']
initializers = ['uniform', 'normal', 'zero']

# make a dictionary of the grid search parameters
param_grid = dict(activation = activations, init = initializers)

# build and fit the GridSearchCV
grid = GridSearchCV(estimator=model, param_grid=param_grid, cv = KFold(random_state=see
d), verbose=10)
grid_result = grid.fit(X_standardized, Y)

# summarize the results
print("Best: {0}, using {1}".format(grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('{0} ({1}) with: {2}'.format(mean, stdev, param))
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
[CV] activation=softmax, init=uniform .............................

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wo
rkers.

[CV] .... activation=softmax, init=uniform, score=0.722, total=   4.8s
[CV] activation=softmax, init=uniform .............................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    4.7s remaining:
0.0s

[CV] .... activation=softmax, init=uniform, score=0.658, total=   5.2s
[CV] activation=softmax, init=uniform .............................

[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    9.9s remaining:
0.0s

[CV] .... activation=softmax, init=uniform, score=0.821, total=   4.4s
[CV] activation=softmax, init=uniform .............................

[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:   14.3s remaining:
0.0s

[CV] .... activation=softmax, init=uniform, score=0.833, total=   4.5s
[CV] activation=softmax, init=uniform .............................

[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:   18.8s remaining:
0.0s

[CV] .... activation=softmax, init=uniform, score=0.846, total=   4.5s
[CV] activation=softmax, init=normal ..............................

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:   23.3s remaining:
0.0s

[CV] ..... activation=softmax, init=normal, score=0.734, total=   4.6s
[CV] activation=softmax, init=normal ..............................

[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:   28.0s remaining:
0.0s

[CV] ..... activation=softmax, init=normal, score=0.620, total=   4.5s
[CV] activation=softmax, init=normal ..............................

[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:   32.5s remaining:
0.0s

[CV] ..... activation=softmax, init=normal, score=0.795, total=   4.7s
[CV] activation=softmax, init=normal ..............................

[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:   37.2s remaining:
0.0s

[CV] ..... activation=softmax, init=normal, score=0.846, total=   4.8s
[CV] activation=softmax, init=normal ..............................

[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:   41.9s remaining:
0.0s
```

```
[CV] ..... activation=softmax, init=normal, score=0.859, total=   4.5s
[CV] activation=softmax, init=zero ...................................
[CV] ....... activation=softmax, init=zero, score=0.646, total=   4.4s
[CV] activation=softmax, init=zero ...................................
[CV] ....... activation=softmax, init=zero, score=0.570, total=   4.3s
[CV] activation=softmax, init=zero ...................................
[CV] ....... activation=softmax, init=zero, score=0.705, total=   5.0s
[CV] activation=softmax, init=zero ...................................
[CV] ....... activation=softmax, init=zero, score=0.744, total=   4.7s
[CV] activation=softmax, init=zero ...................................
[CV] ....... activation=softmax, init=zero, score=0.679, total=   4.6s
[CV] activation=relu, init=uniform ...................................
[CV] ....... activation=relu, init=uniform, score=0.747, total=   4.7s
[CV] activation=relu, init=uniform ...................................
[CV] ....... activation=relu, init=uniform, score=0.633, total=   4.7s
[CV] activation=relu, init=uniform ...................................
[CV] ....... activation=relu, init=uniform, score=0.782, total=   4.3s
[CV] activation=relu, init=uniform ...................................
[CV] ....... activation=relu, init=uniform, score=0.808, total=   4.5s
[CV] activation=relu, init=uniform ...................................
[CV] ....... activation=relu, init=uniform, score=0.833, total=   4.4s
[CV] activation=relu, init=normal ....................................
[CV] ........ activation=relu, init=normal, score=0.772, total=   4.4s
[CV] activation=relu, init=normal ....................................
[CV] ........ activation=relu, init=normal, score=0.633, total=   4.5s
[CV] activation=relu, init=normal ....................................
[CV] ........ activation=relu, init=normal, score=0.731, total=   4.4s
[CV] activation=relu, init=normal ....................................
[CV] ........ activation=relu, init=normal, score=0.808, total=   4.3s
[CV] activation=relu, init=normal ....................................
[CV] ........ activation=relu, init=normal, score=0.795, total=   4.5s
[CV] activation=relu, init=zero ......................................
[CV] .......... activation=relu, init=zero, score=0.646, total=   4.8s
[CV] activation=relu, init=zero ......................................
[CV] .......... activation=relu, init=zero, score=0.570, total=   4.4s
[CV] activation=relu, init=zero ......................................
[CV] .......... activation=relu, init=zero, score=0.705, total=   4.4s
[CV] activation=relu, init=zero ......................................
[CV] .......... activation=relu, init=zero, score=0.744, total=   4.5s
[CV] activation=relu, init=zero ......................................
[CV] .......... activation=relu, init=zero, score=0.679, total=   4.5s
[CV] activation=tanh, init=uniform ...................................
[CV] ....... activation=tanh, init=uniform, score=0.772, total=   4.5s
[CV] activation=tanh, init=uniform ...................................
[CV] ....... activation=tanh, init=uniform, score=0.570, total=   4.4s
[CV] activation=tanh, init=uniform ...................................
[CV] ....... activation=tanh, init=uniform, score=0.833, total=   4.5s
[CV] activation=tanh, init=uniform ...................................
[CV] ....... activation=tanh, init=uniform, score=0.808, total=   4.5s
[CV] activation=tanh, init=uniform ...................................
[CV] ....... activation=tanh, init=uniform, score=0.833, total=   4.6s
[CV] activation=tanh, init=normal ....................................
[CV] ........ activation=tanh, init=normal, score=0.747, total=   4.4s
[CV] activation=tanh, init=normal ....................................
[CV] ........ activation=tanh, init=normal, score=0.620, total=   4.4s
[CV] activation=tanh, init=normal ....................................
[CV] ........ activation=tanh, init=normal, score=0.808, total=   5.0s
[CV] activation=tanh, init=normal ....................................
[CV] ........ activation=tanh, init=normal, score=0.795, total=   4.5s
[CV] activation=tanh, init=normal ....................................
[CV] ........ activation=tanh, init=normal, score=0.833, total=   4.5s
```

```
[CV] activation=tanh, init=zero .....................................
[CV] .......... activation=tanh, init=zero, score=0.646, total=   4.4s
[CV] activation=tanh, init=zero .....................................
[CV] .......... activation=tanh, init=zero, score=0.570, total=   4.5s
[CV] activation=tanh, init=zero .....................................
[CV] .......... activation=tanh, init=zero, score=0.705, total=   4.5s
[CV] activation=tanh, init=zero .....................................
[CV] .......... activation=tanh, init=zero, score=0.744, total=   4.6s
[CV] activation=tanh, init=zero .....................................
[CV] .......... activation=tanh, init=zero, score=0.679, total=   4.4s
[CV] activation=linear, init=uniform ................................
[CV] ..... activation=linear, init=uniform, score=0.835, total=   4.4s
[CV] activation=linear, init=uniform ................................
[CV] ..... activation=linear, init=uniform, score=0.633, total=   4.5s
[CV] activation=linear, init=uniform ................................
[CV] ..... activation=linear, init=uniform, score=0.821, total=   4.5s
[CV] activation=linear, init=uniform ................................
[CV] ..... activation=linear, init=uniform, score=0.846, total=   4.5s
[CV] activation=linear, init=uniform ................................
[CV] ..... activation=linear, init=uniform, score=0.821, total=   4.3s
[CV] activation=linear, init=normal .................................
[CV] ...... activation=linear, init=normal, score=0.835, total=   5.1s
[CV] activation=linear, init=normal .................................
[CV] ...... activation=linear, init=normal, score=0.620, total=   4.4s
[CV] activation=linear, init=normal .................................
[CV] ...... activation=linear, init=normal, score=0.821, total=   4.3s
[CV] activation=linear, init=normal .................................
[CV] ...... activation=linear, init=normal, score=0.859, total=   4.4s
[CV] activation=linear, init=normal .................................
[CV] ...... activation=linear, init=normal, score=0.808, total=   4.4s
[CV] activation=linear, init=zero ...................................
[CV] ........ activation=linear, init=zero, score=0.646, total=   4.2s
[CV] activation=linear, init=zero ...................................
[CV] ........ activation=linear, init=zero, score=0.570, total=   4.4s
[CV] activation=linear, init=zero ...................................
[CV] ........ activation=linear, init=zero, score=0.705, total=   4.5s
[CV] activation=linear, init=zero ...................................
[CV] ........ activation=linear, init=zero, score=0.744, total=   4.4s
[CV] activation=linear, init=zero ...................................
[CV] ........ activation=linear, init=zero, score=0.679, total=   4.4s

[Parallel(n_jobs=1)]: Done  60 out of  60 | elapsed:  4.5min finished
```

```
Best: 0.7911067843437195, using {'activation': 'linear', 'init': 'unifor
m'}
0.7759493708610534 (0.07352347706662563) with: {'activation': 'softmax',
'init': 'uniform'}
0.7708860754966735 (0.08722145913036237) with: {'activation': 'softmax',
'init': 'normal'}
0.6686789989471436 (0.05899773033245815) with: {'activation': 'softmax',
'init': 'zero'}
0.7605647444725037 (0.06994542741212405) with: {'activation': 'relu', 'ini
t': 'uniform'}
0.7476793169975281 (0.06306381621409568) with: {'activation': 'relu', 'ini
t': 'normal'}
0.6686789989471436 (0.05899773033245815) with: {'activation': 'relu', 'ini
t': 'zero'}
0.7632262110710144 (0.09936818298562612) with: {'activation': 'tanh', 'ini
t': 'uniform'}
0.7605972051620483 (0.07558978386106281) with: {'activation': 'tanh', 'ini
t': 'normal'}
0.6686789989471436 (0.05899773033245815) with: {'activation': 'tanh', 'ini
t': 'zero'}
0.7911067843437195 (0.07968826468817176) with: {'activation': 'linear', 'i
nit': 'uniform'}
0.7885751247406005 (0.08587248740892742) with: {'activation': 'linear', 'i
nit': 'normal'}
0.6686789989471436 (0.05899773033245815) with: {'activation': 'linear', 'i
nit': 'zero'}
```

In [18]:

```
# Define a random seed
seed = 6
np.random.seed(seed)
init = 'uniform'   # taken from previous results
activation = 'linear'
# Start defining the model
def create_model(neuron1, neuron2):
    # Create model
    model = Sequential()
    model.add(Dense(neuron1, input_dim = 8, kernel_initializer = init, activation = act
ivation))
#    model.add(Dropout(dropout_rate))
    model.add(Dense(neuron2, input_dim = neuron1, kernel_initializer = init, activation
= activation))
#    model.add(Dropout(dropout_rate))
    model.add(Dense(1, activation = 'sigmoid'))

    # Compile the model
    adam = Adam(lr = learn_rate)
    model.compile(loss = 'binary_crossentropy', optimizer = adam, metrics = ['accuracy'
])
    return model

# Create the model
model = KerasClassifier(build_fn = create_model, epochs=50, batch_size=10, verbose = 0)

# define the grid search parameters
neuron1 = [4, 8, 16]
neuron2 = [2, 4, 8]

# make a dictionary of the grid search parameters
param_grid = dict(neuron1 = neuron1, neuron2 = neuron2)

# build and fit the GridSearchCV
grid = GridSearchCV(estimator=model, param_grid=param_grid, cv = KFold(random_state=see
d), refit=True, verbose=10) # To retrain with the best parameters found so far
grid_result = grid.fit(X_standardized, Y)

# summarize the results
print("Best: {0}, using {1}".format(grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print('{0} ({1}) with: {2}'.format(mean, stdev, param))
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV] neuron1=4, neuron2=2 ............................................

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wo
rkers.

[CV] ................ neuron1=4, neuron2=2, score=0.835, total=   4.6s
[CV] neuron1=4, neuron2=2 ............................................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    4.5s remaining:
0.0s

[CV] ................ neuron1=4, neuron2=2, score=0.633, total=   4.1s
[CV] neuron1=4, neuron2=2 ............................................

[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    8.6s remaining:
0.0s

[CV] ................ neuron1=4, neuron2=2, score=0.821, total=   4.5s
[CV] neuron1=4, neuron2=2 ............................................

[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:   13.1s remaining:
0.0s

[CV] ................ neuron1=4, neuron2=2, score=0.846, total=   5.1s
[CV] neuron1=4, neuron2=2 ............................................

[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:   18.2s remaining:
0.0s

[CV] ................ neuron1=4, neuron2=2, score=0.833, total=   4.5s
[CV] neuron1=4, neuron2=4 ............................................

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:   22.7s remaining:
0.0s

[CV] ................ neuron1=4, neuron2=4, score=0.835, total=   4.5s
[CV] neuron1=4, neuron2=4 ............................................

[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:   27.1s remaining:
0.0s

[CV] ................ neuron1=4, neuron2=4, score=0.608, total=   4.4s
[CV] neuron1=4, neuron2=4 ............................................

[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:   31.6s remaining:
0.0s

[CV] ................ neuron1=4, neuron2=4, score=0.821, total=   4.2s
[CV] neuron1=4, neuron2=4 ............................................

[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:   35.8s remaining:
0.0s

[CV] ................ neuron1=4, neuron2=4, score=0.846, total=   4.3s
[CV] neuron1=4, neuron2=4 ............................................

[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:   40.1s remaining:
0.0s
```

```
[CV] ............... neuron1=4, neuron2=4, score=0.808, total=   4.2s
[CV] neuron1=4, neuron2=8 .....................................
[CV] ............... neuron1=4, neuron2=8, score=0.835, total=   4.3s
[CV] neuron1=4, neuron2=8 .....................................
[CV] ............... neuron1=4, neuron2=8, score=0.633, total=   4.5s
[CV] neuron1=4, neuron2=8 .....................................
[CV] ............... neuron1=4, neuron2=8, score=0.821, total=   4.5s
[CV] neuron1=4, neuron2=8 .....................................
[CV] ............... neuron1=4, neuron2=8, score=0.846, total=   4.5s
[CV] neuron1=4, neuron2=8 .....................................
[CV] ............... neuron1=4, neuron2=8, score=0.821, total=   4.4s
[CV] neuron1=8, neuron2=2 .....................................
[CV] ............... neuron1=8, neuron2=2, score=0.835, total=   4.5s
[CV] neuron1=8, neuron2=2 .....................................
[CV] ............... neuron1=8, neuron2=2, score=0.633, total=   5.0s
[CV] neuron1=8, neuron2=2 .....................................
[CV] ............... neuron1=8, neuron2=2, score=0.833, total=   4.4s
[CV] neuron1=8, neuron2=2 .....................................
[CV] ............... neuron1=8, neuron2=2, score=0.846, total=   4.4s
[CV] neuron1=8, neuron2=2 .....................................
[CV] ............... neuron1=8, neuron2=2, score=0.808, total=   4.2s
[CV] neuron1=8, neuron2=4 .....................................
[CV] ............... neuron1=8, neuron2=4, score=0.835, total=   4.3s
[CV] neuron1=8, neuron2=4 .....................................
[CV] ............... neuron1=8, neuron2=4, score=0.608, total=   4.3s
[CV] neuron1=8, neuron2=4 .....................................
[CV] ............... neuron1=8, neuron2=4, score=0.821, total=   4.5s
[CV] neuron1=8, neuron2=4 .....................................
[CV] ............... neuron1=8, neuron2=4, score=0.846, total=   4.5s
[CV] neuron1=8, neuron2=4 .....................................
[CV] ............... neuron1=8, neuron2=4, score=0.833, total=   4.4s
[CV] neuron1=8, neuron2=8 .....................................
[CV] ............... neuron1=8, neuron2=8, score=0.835, total=   4.4s
[CV] neuron1=8, neuron2=8 .....................................
[CV] ............... neuron1=8, neuron2=8, score=0.620, total=   4.9s
[CV] neuron1=8, neuron2=8 .....................................
[CV] ............... neuron1=8, neuron2=8, score=0.821, total=   4.6s
[CV] neuron1=8, neuron2=8 .....................................
[CV] ............... neuron1=8, neuron2=8, score=0.833, total=   4.8s
[CV] neuron1=8, neuron2=8 .....................................
[CV] ............... neuron1=8, neuron2=8, score=0.833, total=   4.5s
[CV] neuron1=16, neuron2=2 ....................................
[CV] ............... neuron1=16, neuron2=2, score=0.835, total=   5.2s
[CV] neuron1=16, neuron2=2 ....................................
[CV] ............... neuron1=16, neuron2=2, score=0.620, total=   4.5s
[CV] neuron1=16, neuron2=2 ....................................
[CV] ............... neuron1=16, neuron2=2, score=0.821, total=   4.5s
[CV] neuron1=16, neuron2=2 ....................................
[CV] ............... neuron1=16, neuron2=2, score=0.833, total=   4.4s
[CV] neuron1=16, neuron2=2 ....................................
[CV] ............... neuron1=16, neuron2=2, score=0.833, total=   4.5s
[CV] neuron1=16, neuron2=4 ....................................
[CV] ............... neuron1=16, neuron2=4, score=0.835, total=   4.5s
[CV] neuron1=16, neuron2=4 ....................................
[CV] ............... neuron1=16, neuron2=4, score=0.633, total=   4.2s
[CV] neuron1=16, neuron2=4 ....................................
[CV] ............... neuron1=16, neuron2=4, score=0.821, total=   5.0s
[CV] neuron1=16, neuron2=4 ....................................
[CV] ............... neuron1=16, neuron2=4, score=0.846, total=   4.6s
[CV] neuron1=16, neuron2=4 ....................................
[CV] ............... neuron1=16, neuron2=4, score=0.833, total=   4.9s
```

```
[CV] neuron1=16, neuron2=8 ...........................................
[CV] .............. neuron1=16, neuron2=8, score=0.835, total=   4.3s
[CV] neuron1=16, neuron2=8 ...........................................
[CV] .............. neuron1=16, neuron2=8, score=0.633, total=   4.6s
[CV] neuron1=16, neuron2=8 ...........................................
[CV] .............. neuron1=16, neuron2=8, score=0.833, total=   4.8s
[CV] neuron1=16, neuron2=8 ...........................................
[CV] .............. neuron1=16, neuron2=8, score=0.833, total=   4.6s
[CV] neuron1=16, neuron2=8 ...........................................
[CV] .............. neuron1=16, neuron2=8, score=0.821, total=   5.2s


[Parallel(n_jobs=1)]: Done  45 out of  45 | elapsed:  3.4min finished

Best: 0.7936708807945252, using {'neuron1': 4, 'neuron2': 2}
0.7936708807945252 (0.08079181748331918) with: {'neuron1': 4, 'neuron2':
2}
0.783479380607605 (0.08890601661969585) with: {'neuron1': 4, 'neuron2': 4}
0.7911067843437195 (0.07968826468817176) with: {'neuron1': 4, 'neuron2':
8}
0.7911067724227905 (0.08009971934827462) with: {'neuron1': 8, 'neuron2':
2}
0.7886075854301453 (0.09087249991485369) with: {'neuron1': 8, 'neuron2':
4}
0.7885751247406005 (0.08432733392428071) with: {'neuron1': 8, 'neuron2':
8}
0.7885751247406005 (0.08432733392428071) with: {'neuron1': 16, 'neuron2':
2}
0.7936708807945252 (0.08079181748331918) with: {'neuron1': 16, 'neuron2':
4}
0.7911067724227905 (0.0792746637362651) with: {'neuron1': 16, 'neuron2':
8}
```

In [19]:

```
# generate predictions with optimal hyperparameters
y_pred = grid.predict(X_standardized)
```

In [20]:

```
from sklearn.metrics import classification_report, accuracy_score

print(accuracy_score(Y, y_pred))
print(classification_report(Y, y_pred))
```

```
0.7857142857142857
              precision    recall  f1-score   support

           0       0.81      0.89      0.85       262
           1       0.72      0.58      0.64       130

    accuracy                           0.79       392
   macro avg       0.76      0.74      0.75       392
weighted avg       0.78      0.79      0.78       392
```

In [21]:

```
# example datapoint
example = df.iloc[1]
print(example)
```

```
n_pregnant                   0.000
glucose_concentration      137.000
blood_pressuer (mm Hg)      40.000
skin_thickness (mm)         35.000
serum_insulin (mu U/ml)    168.000
BMI                         43.100
pedigree_function            2.288
age                         33.000
class                        1.000
Name: 4, dtype: float64
```

In [22]:

```
# make a prediction using our optimized deep neural network
prediction = grid.predict(X_standardized[1].reshape(1, -1))
print(prediction)
```

```
[[1]]
```