In [1]:

```python
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot
%matplotlib inline

#model validation
from sklearn.metrics import classification_report,confusion_matrix,precision_score,f1_score
from sklearn import metrics

# cross validation
from sklearn.model_selection import StratifiedKFold

from sklearn.neighbors import KNeighborsClassifier
```

In [2]:

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

In [3]:

```python
#import the heart disease dataset
df = pd.read_csv("heart_statlog.csv")
df.head()
```

Out[3]:

	age	sex	chest	resting_blood_pressure	serum_cholestoral	fasting_blood_sugar	resting_ele
0	70	1	4	130	322	0	
1	67	0	3	115	564	0	
2	57	1	2	124	261	0	
3	64	1	4	128	263	0	
4	74	0	2	120	269	0	

In [4]:

```python
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df['class'] = le.fit_transform(df["class"])
```

In [5]:

```python
# segregating dataset into features i.e., X and target variables i.e., y
X = df.drop(['class'],axis=1)
y = df['class']
```

In [6]:

```python
#splitting the model into training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2,shuffle
```

In [7]:

```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train[['age','resting_blood_pressure','serum_cholestoral','maximum_heart_rate_achieved','
X_train.head()
```

Out[7]:

	age	sex	chest	resting_blood_pressure	serum_cholestoral	fasting_blood_sugar	rest
197	0.465116	0	3	0.10	0.302405	0	
174	0.000000	1	1	0.18	0.192440	0	
173	0.790698	0	3	0.20	0.292096	0	
249	0.604651	1	4	0.45	0.536082	0	
207	0.558140	1	3	0.05	0.391753	0	

In [8]:

```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_test[['age','resting_blood_pressure','serum_cholestoral','maximum_heart_rate_achieved','o
X_test.head()
```

Out[8]:

	age	sex	chest	resting_blood_pressure	serum_cholestoral	fasting_blood_sugar	rest
260	0.617021	0	3	0.309524	0.460241	0	
194	0.404255	1	3	0.357143	0.255422	1	
112	0.659574	0	4	0.761905	0.375904	0	
14	0.595745	0	4	0.404762	0.371084	0	
206	0.659574	0	3	0.095238	0.407229	0	

In [9]:

```python
from sklearn import model_selection
from sklearn.model_selection import cross_val_score

# function initializing baseline machine learning models
def GetBasedModel():
    basedModels = []
    basedModels.append(('KNN'  , KNeighborsClassifier(9)))
    return basedModels

# function for performing 10-fold cross validation of all the baseline models
def BasedLine2(X_train, y_train,models):
    # Test options and evaluation metric
    num_folds = 10
    scoring = 'accuracy'
    seed = 7
    results = []
    names = []
    for name, model in models:
        kfold = model_selection.KFold(n_splits=10, random_state=None)
        cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, sco
        results.append(cv_results)
        names.append(name)
        msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
        print(msg)


    return results,msg
```

In [10]:

```python
models = GetBasedModel()
names,results = BasedLine2(X_train, y_train,models)
```

KNN: 0.804113 (0.100220)

In [11]:

```python
knn = KNeighborsClassifier(9)
knn.fit(X_train,y_train)
y_pred_knn = knn.predict(X_test)
```

In [12]:

```python
CM=confusion_matrix(y_test,y_pred_knn)

TN = CM[0][0]
FN = CM[1][0]
TP = CM[1][1]
FP = CM[0][1]
specificity = TN/(TN+FP)
sensitivity = TP/(TP+FN)
acc= accuracy_score(y_test, y_pred_knn)
prec = precision_score(y_test, y_pred_knn)
rec = recall_score(y_test, y_pred_knn)
f1 = f1_score(y_test, y_pred_knn)

model_results =pd.DataFrame([['KNN',acc, prec,rec,f1,sensitivity,specificity]],
             columns = ['Model', 'Accuracy','Precision','Recall','f1 score','Sensitivity'

model_results
```

Out[12]:

	Model	Accuracy	Precision	Recall	f1 score	Sensitivity	Specificity
0	KNN	0.833333	0.8	0.833333	0.816327	0.833333	0.833333

In [17]:

```python
pip install sklearn-genetic
```

Note: you may need to restart the kernel to use updated packages.

'C:\Users\Nupur' is not recognized as an internal or external command,
operable program or batch file.

In [14]:

```python
from genetic_selection import GeneticSelectionCV

from sklearn.neighbors import KNeighborsClassifier
#build the model with your preferred hyperparameters.
knn = KNeighborsClassifier(9)

# create the GeneticSelection search with the different parameters available.
knn = GeneticSelectionCV(knn,
                          cv=10,
                          scoring="accuracy",
                          max_features=13,
                          n_population=270,
                          crossover_proba=0.5,
                          mutation_proba=0.2,
                          n_generations=50,
                          crossover_independent_proba=0.5,
                          mutation_independent_proba=0.05,
                          n_gen_no_change=10,
                          n_jobs=-1)

# fit the GA search to our data.
knn = knn.fit(X_train, y_train)

# print the results.
print(knn.support_)
```

```
[False False  True False False  True False  True False False False  True
  True]
```

In [15]:

```python
y_pred_knn = knn.predict(X_test)
print("Accuracy score after genetic algorithm is= "+str(accuracy_score(y_test,y_pred_knn)))
```

```
Accuracy score after genetic algorithm is= 0.8703703703703703
```

In [16]:

```python
CM=confusion_matrix(y_test,y_pred_knn)

TN = CM[0][0]
FN = CM[1][0]
TP = CM[1][1]
FP = CM[0][1]
specificity = TN/(TN+FP)
sensitivity = TP/(TP+FN)
acc= accuracy_score(y_test, y_pred_knn)
prec = precision_score(y_test, y_pred_knn)
rec = recall_score(y_test, y_pred_knn)
f1 = f1_score(y_test, y_pred_knn)

model_results =pd.DataFrame([['KNN',acc, prec,rec,f1,sensitivity,specificity]],
              columns = ['Model', 'Accuracy','Precision','Recall','f1 score','Sensitivity'

model_results
```

Out[16]:

	Model	Accuracy	Precision	Recall	f1 score	Sensitivity	Specificity
0	KNN	0.87037	0.84	0.875	0.857143	0.875	0.866667