

In [1]:

```

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.svm import SVC

```

In [2]:

```

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn import metrics

```

In [3]:

```

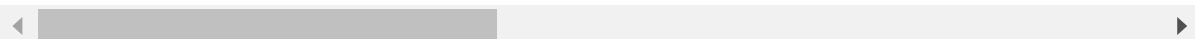
#import the heart disease dataset
df = pd.read_csv("heart_statlog.csv")
df

```

Out[3]:

	age	sex	chest	resting_blood_pressure	serum_cholesterol	fasting_blood_sugar	resting_heart_rate
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	
...
265	52	1	3	172	199	1	
266	44	1	2	120	263	0	
267	56	0	2	140	294	0	
268	57	1	4	140	192	0	
269	67	1	4	160	286	0	

270 rows × 14 columns



In [4]:

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

Out[4]:

	age	sex	chest	resting_blood_pressure	serum_cholesterol	fasting_blood_sugar	resting_heart_rate
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	
...
265	52	1	3	172	199	1	
266	44	1	2	120	263	0	
267	56	0	2	140	294	0	
268	57	1	4	140	192	0	
269	67	1	4	160	286	0	

270 rows × 14 columns

In [5]:

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

In [6]:

```
#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=True)
```

In [7]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train[['age', 'resting_blood_pressure', 'serum_cholesterol', 'maximum_heart_rate_achieved', 'fasting_blood_sugar', 'resting_blood_sugar']] = scaler.fit_transform(X_train[['age', 'resting_blood_pressure', 'serum_cholesterol', 'maximum_heart_rate_achieved', 'fasting_blood_sugar', 'resting_blood_sugar']])
X_train.head()
```

Out[7]:

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

In [8]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_test[['age', 'resting_blood_pressure', 'serum_cholesterol', 'maximum_heart_rate_achieved', 'fasting_blood_sugar', 'resting_blood_sugar']] = scaler.fit_transform(X_test[['age', 'resting_blood_pressure', 'serum_cholesterol', 'maximum_heart_rate_achieved', 'fasting_blood_sugar', 'resting_blood_sugar']])
X_test.head()
```

Out[8]:

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

In [9]:

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

# function initializing baseline machine learning models
def GetBasedModel():
    basedModels = []
    basedModels.append(('SVM Linear' , SVC(kernel='linear',gamma='auto',probability=True)))
    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, scoring=scoring)
        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]:

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

SVM Linear: 0.813203 (0.128437)

In [11]:

```
svc = SVC(kernel='linear',gamma='auto',probability=True)
svc.fit(X_train,y_train)
y_pred_svc = svc.predict(X_test)
```

In [12]:

```
CM=confusion_matrix(y_test,y_pred_svc)

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_svc)
prec = precision_score(y_test, y_pred_svc)
rec = recall_score(y_test, y_pred_svc)
f1 = f1_score(y_test, y_pred_svc)

model_results =pd.DataFrame([[ 'SVM',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	SVM	0.851852	0.807692	0.875	0.84	0.875	0.833333