Heart Disease

December 8, 2020

HEART DISEASE PREDICTION USING LOGISTIC REGRESSION.

Table of Contents

1 Research Question

- Using Logistic Regression can we pinpoint the most relevant/risk factors of heart disease as well as predict the overall risk?

1.1 Data Dictionary:

Column Name	Description
Male	1 = Male 0 = Female
Age	Age of patient
Current Smoker	Whether patient is a current smoker
cigsPerDay	Number of cigarettes average in one day
BPmeds	Patient on BP meds or not.
PrevalentStroke	Whether patient has history of stroke
PrevelantHyp	Whether patient is hypersentive or not
diabetes	whether patient is diabetic
totChol	total cholesterol level
sysBP	Systolic blood pressure
diaBP	Diastolic Blood pressure
BMI	Body mass Index
heartRate	Beats per minute
glucose	sugar level in blood
TenYearCHD	10 year risk of CHD. 1 = Yes 0 = No

2 Import Libraries

[89]:
```
import pandas as pd # for manipulating data
import numpy as np # Linear Algebra operations
import matplotlib.pyplot as plt # for visualizations
import matplotlib.mlab as mlab # for Visualization
import statsmodels.api as sm #API for stat model eval
from sklearn.linear_model import LogisticRegression #Model
from sklearn.model_selection import train_test_split
import seaborn as sns # for visualizations
import scipy.stats as st #Stats
from sklearn.metrics import confusion_matrix #True Positive False Negative
```

3 Load Data

[5]:
```
heart_df = pd.read_csv('framingham.csv')
```

4 Exploratory Data Analysis (EDA)

[7]:
```
heart_df.head()
```

[7]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	0
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	0
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	1
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	0

4.0.1 Check Null Values

Exploring the possibility of missing values. There are no missing values in this dataset.

[6]:
```
heart_df.isnull().sum()
```

[6]:
```
male                0
age                 0
education         105
currentSmoker       0
cigsPerDay         29
BPMeds             53
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol            50
sysBP               0
```

```
diaBP               0
BMI                19
heartRate           1
glucose           388
TenYearCHD          0
dtype: int64
```

4.0.2 Determine total Null values

Total number of rows missing value is 582

[10]:
```
count = 0
for number in heart_df.isnull().sum(axis = 1):
    if number > 0:
        count = count+1
print('Total number of rows missing value is ',count)
```

```
Total number of rows missing value is  582
```

4.0.3 Calculate percentage of entire dataset is Null values

This equals to 14 percent of the entire dataset with missing values.

[15]:
```
print('This equals to',round((count/len(heart_df.index))*100), 'percent of the␣
 ↪entire dataset with missing values.')
```

```
This equals to 14 percent of the entire dataset with missing values.
```

[16]:
```
heart_df.dropna(axis= 0, inplace= True)
```

4.0.4 Data Types

Ensuring our data types are correct before doing further exploration. Our classes are numeric so this makes sense to see integers and floats.

[17]:
```
heart_df.dtypes
```

[17]:
```
male                int64
age                 int64
education         float64
currentSmoker       int64
cigsPerDay        float64
BPMeds            float64
prevalentStroke     int64
prevalentHyp        int64
diabetes            int64
totChol           float64
sysBP             float64
```

```
diaBP              float64
BMI                float64
heartRate          float64
glucose            float64
TenYearCHD           int64
dtype: object
```

4.0.5 Summery Statistics

[18]:
```
heart_df.describe().style.set_caption('There are 768 rows, with potential␣
↪outliers.')
```

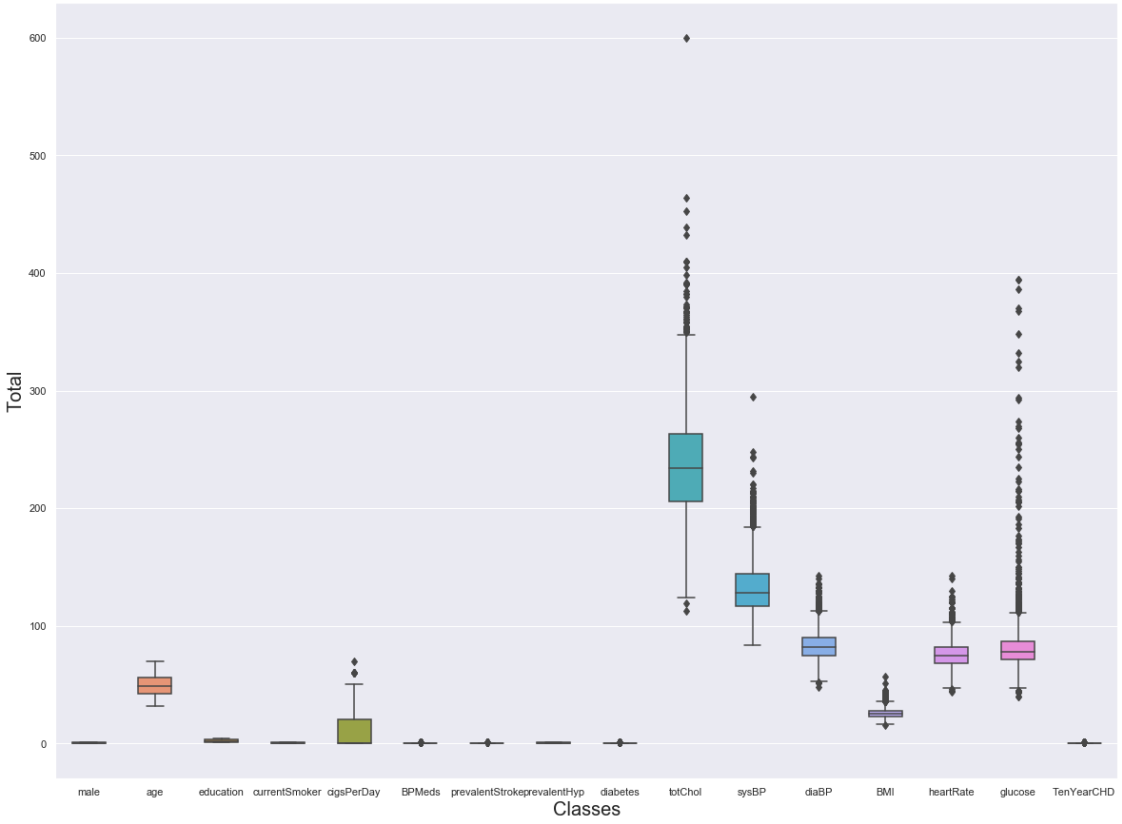
[18]:
```
<pandas.io.formats.style.Styler at 0x7faf8acd1650>
```

[20]:
```
heart_df.shape
```

[20]:
```
(3658, 16)
```

4.1 Visualizing Outliers:

[39]:
```
# let's see how data is distributed for every column
fig, ax = plt.subplots(figsize=(20,15))
sns.boxplot(data=heart_df, width= 0.5,ax=ax,  fliersize=6)
sns.set()
ax.set_xlabel('Classes',fontsize=20);
ax.set_ylabel('Total',fontsize=20);
```



5 Feature Engineering

5.0.1 Handeling Outliers

We are utilizing feature engineering to clean our independent variables to ensure model accuracy. - The following were removed: - we are removing the top 2% data from the cigsPerday column - we are removing the top 1% data from the BMI column - we are removing the top 1% data from the sysBP column - we are removing the top 5% data from the Insulin column

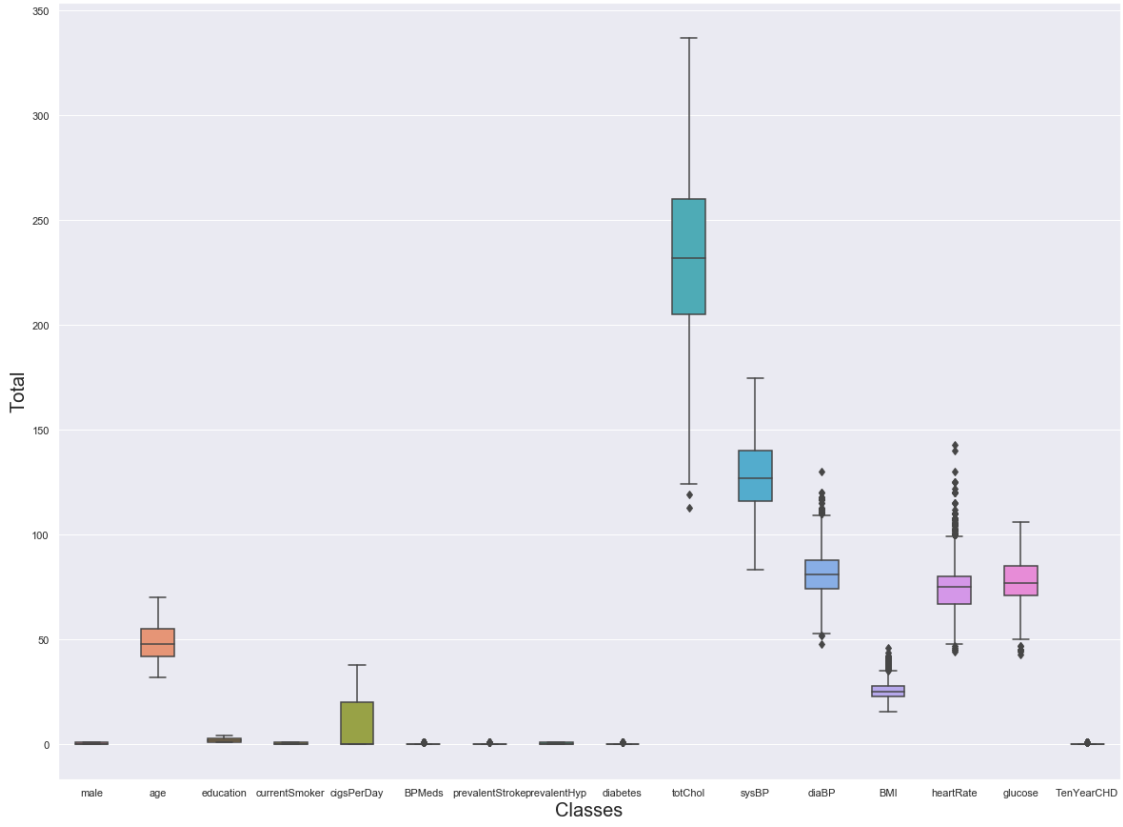
[90]:
```
q = heart_df['cigsPerDay'].quantile(0.98)
# we are removing the top 2% data from the cigsPerday column
data_cleaned = heart_df[heart_df['cigsPerDay']<q]

q = data_cleaned['totChol'].quantile(0.98)
# we are removing the top 1% data from the BMI column
data_cleaned  = data_cleaned[data_cleaned['totChol']<q]

q = data_cleaned['sysBP'].quantile(0.95)
# we are removing the top 1% data from the sysBP column
data_cleaned  = data_cleaned[data_cleaned['sysBP']<q]
```

```
q = data_cleaned['glucose'].quantile(0.95)
# we are removing the top 5% data from the Insulin column
data_cleaned  = data_cleaned[data_cleaned['glucose']<q]
```

5.0.2 Revisiting Skewness after Outlier handeling

The data looks much better now than before. We will start our analysis with this data now as we don't want to lose important information. If our model doesn't work with accuracy, we will come back for more preprocessing.

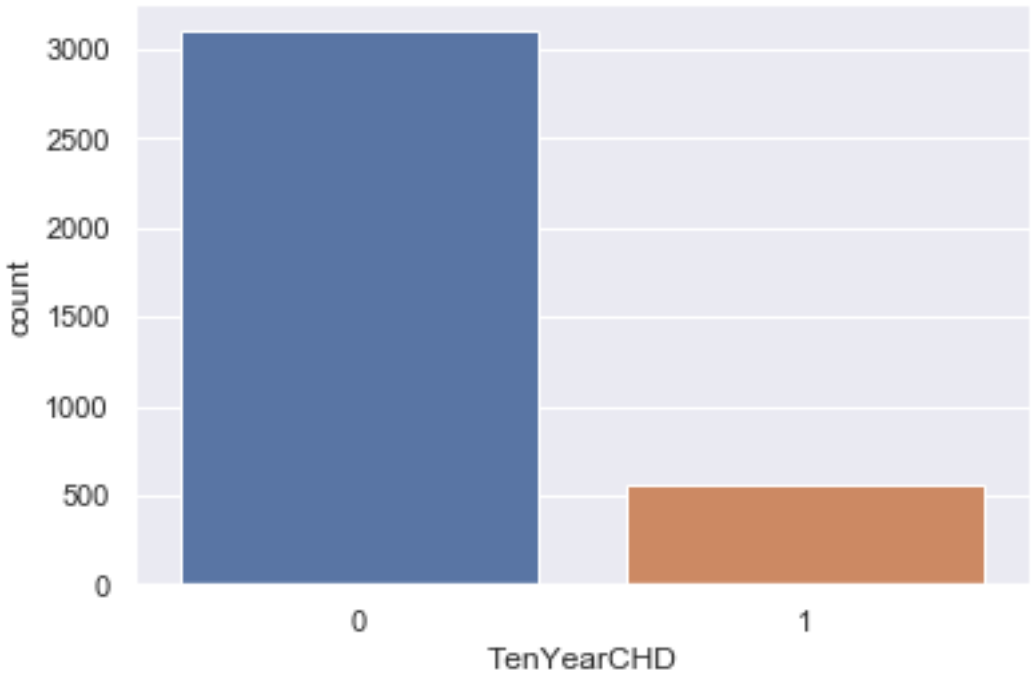
[47]:
```
fig, ax = plt.subplots(figsize=(20,15))
sns.boxplot(data=data_cleaned, width= 0.5,ax=ax,  fliersize=6)
sns.set()
ax.set_xlabel('Classes',fontsize=20);
ax.set_ylabel('Total',fontsize=20);
```



There are 3179 patents with no heart disease and 572 patients with risk of heart disease

[49]:
```
sns.countplot(x='TenYearCHD',data=heart_df)
```

[49]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7faf8081e0d0>
```



5.1 Logistic Regression

Logistic regression is a type of regression analysis in statistics used for prediction of outcome of a categorical dependent variable from a set of predictor or independent variables. In logistic regression the dependent variable is always binary. Logistic regression is mainly used to for prediction and also calculating the probability of success.

[51]:
```
from statsmodels.tools import add_constant as add_constant
heart_df_constant = add_constant(heart_df)
heart_df_constant.head()
```

[51]:

	const	male	age	education	currentSmoker	cigsPerDay	BPMeds	\
0	1.0	1	39	4.0	0	0.0	0.0	
1	1.0	0	46	2.0	0	0.0	0.0	
2	1.0	1	48	1.0	1	20.0	0.0	
3	1.0	0	61	3.0	1	30.0	0.0	
4	1.0	0	46	3.0	1	23.0	0.0	

	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	\
0	0	0	0	195.0	106.0	70.0	26.97	
1	0	0	0	250.0	121.0	81.0	28.73	
2	0	0	0	245.0	127.5	80.0	25.34	
3	0	1	0	225.0	150.0	95.0	28.58	
4	0	0	0	285.0	130.0	84.0	23.10	

```
   heartRate  glucose  TenYearCHD
0       80.0     77.0           0
1       95.0     76.0           0
2       75.0     70.0           0
3       65.0    103.0           1
4       85.0     85.0           0
```

The results below show some of the attributes with P value higher than the preferred alpha(5%) and thereby showing low statistically significant relationship with the probability of heart disease.

- Backward elemination approach is used here to remove those attributes with highest Pvalue one at a time follwed by running the regression repeatedly until all attributes have P Values less than 0.05.

[54]:
```
st.chisqprob = lambda chisq, df: st.chi2.sf(chisq, df)
cols=heart_df_constant.columns[:-1]
model=sm.Logit(heart_df.TenYearCHD,heart_df_constant[cols])
result=model.fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.376500
         Iterations 7
```

[54]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:             TenYearCHD   No. Observations:                 3658
Model:                          Logit   Df Residuals:                     3642
Method:                           MLE   Df Model:                           15
Date:                Tue, 08 Dec 2020   Pseudo R-squ.:                 0.1175
Time:                        15:22:54   Log-Likelihood:                -1377.2
converged:                       True   LL-Null:                       -1560.6
Covariance Type:            nonrobust   LLR p-value:                6.676e-69
==============================================================================
===
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---
const          -8.3282      0.715    -11.640      0.000      -9.730
-6.926
male            0.5553      0.109      5.093      0.000       0.342
0.769
age             0.0635      0.007      9.509      0.000       0.050
0.077
education      -0.0478      0.049     -0.967      0.334      -0.145
```

```
0.049
currentSmoker        0.0716      0.157      0.457      0.648      -0.236
0.379
cigsPerDay           0.0179      0.006      2.872      0.004       0.006
0.030
BPMeds               0.1625      0.234      0.693      0.488      -0.297
0.622
prevalentStroke      0.6937      0.490      1.417      0.157      -0.266
1.653
prevalentHyp         0.2342      0.138      1.697      0.090      -0.036
0.505
diabetes             0.0392      0.316      0.124      0.901      -0.579
0.658
totChol              0.0023      0.001      2.070      0.038       0.000
0.005
sysBP                0.0154      0.004      4.044      0.000       0.008
0.023
diaBP               -0.0042      0.006     -0.646      0.518      -0.017
0.008
BMI                  0.0067      0.013      0.523      0.601      -0.018
0.032
heartRate           -0.0032      0.004     -0.771      0.441      -0.012
0.005
glucose              0.0071      0.002      3.190      0.001       0.003
0.012
==============================================================================
===
"""
```

Backward elemination approach is used here to remove those attributes with highest Pvalue one at a time follwed by running the regression repeatedly until all attributes have P Values less than 0.05.

[55]:
```
def back_feature_elem (data_frame,dep_var,col_list):
    while len(col_list)>0 :
        model=sm.Logit(dep_var,data_frame[col_list])
        result=model.fit(disp=0)
        largest_pvalue=round(result.pvalues,3).nlargest(1)
        if largest_pvalue[0]<(0.05):
            return result
            break
        else:
            col_list=col_list.drop(largest_pvalue.index)

result=back_feature_elem(heart_df_constant,heart_df.TenYearCHD,cols)
```

Takes in the dataframe, the dependent variable and a list of column names, runs the regression repeatedly eleminating feature with the highest P-value above alpha one at a time and returns the

regression summary with all p-values below alpha

[56]:
```
result.summary()
```

[56]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:             TenYearCHD   No. Observations:                 3658
Model:                          Logit   Df Residuals:                     3651
Method:                           MLE   Df Model:                            6
Date:                Tue, 08 Dec 2020   Pseudo R-squ.:                  0.1148
Time:                        15:24:34   Log-Likelihood:                -1381.4
converged:                       True   LL-Null:                       -1560.6
Covariance Type:            nonrobust   LLR p-value:                 2.408e-74
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -9.1353      0.475    -19.213      0.000     -10.067      -8.203
male           0.5617      0.107      5.258      0.000       0.352       0.771
age            0.0660      0.006     10.267      0.000       0.053       0.079
cigsPerDay     0.0192      0.004      4.606      0.000       0.011       0.027
totChol        0.0023      0.001      2.031      0.042    8.01e-05       0.004
sysBP          0.0175      0.002      8.155      0.000       0.013       0.022
glucose        0.0073      0.002      4.343      0.000       0.004       0.011
==============================================================================
"""
```

5.2 Interpreting Results

This fitted model shows that, holding all other features constant, the odds of getting diagnosed with heart disease for males (sex_male = 1)over that of females (sex_male = 0) is 1.753586.

In terms of percent change, we can say that the odds for males are 75.3% higher than the odds for females.

[63]:
```
params = np.exp(result.params)
conf = np.exp(result.conf_int())
conf['OR'] = params
pvalue=round(result.pvalues,3)
conf['pvalue']=pvalue
conf.columns = ['CI 95%(2.5%)', 'CI 95%(97.5%)', 'Odds Ratio','pvalue']
print ((conf))
```

```
            CI 95%(2.5%)  CI 95%(97.5%)  Odds Ratio  pvalue
const           0.000042       0.000274    0.000108   0.000
male            1.422325       2.161998    1.753586   0.000
age             1.054823       1.081727    1.068190   0.000
cigsPerDay      1.011106       1.027788    1.019413   0.000
```

```
totChol         1.000080       1.004491    1.002283    0.042
sysBP           1.013404       1.021977    1.017682    0.000
glucose         1.004004       1.010626    1.007310    0.000
```

5.3 Train Test Split

[91]:
```
import sklearn
new_features=heart_df[['age','male','cigsPerDay','totChol','sysBP','glucose','TenYearCHD']]
x=new_features.iloc[:,:-1]
y=new_features.iloc[:,-1]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=5)
```

Accuracy model is 0.85

[92]:
```
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression()
logreg.fit(x_train,y_train)
y_pred=logreg.predict(x_test)
```

[68]:
```
sklearn.metrics.accuracy_score(y_test,y_pred)
```

[68]:
```
0.8592896174863388
```

5.4 Confusion Matrix

True Positives: 8

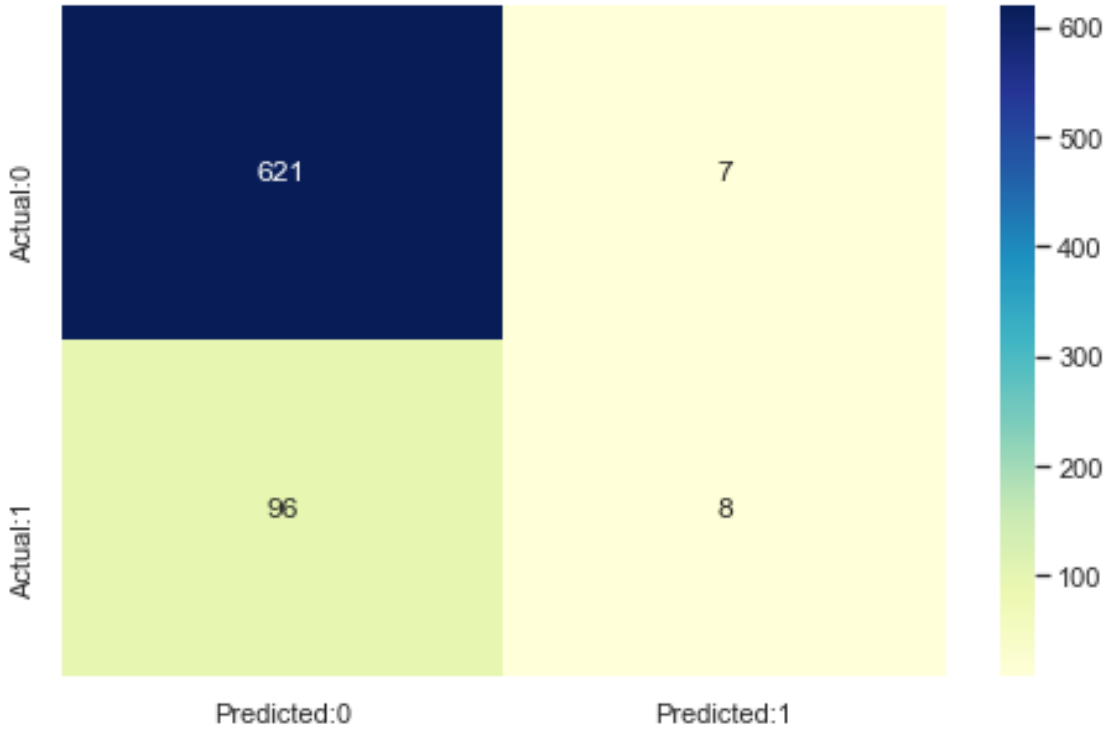
True Negatives: 621

False Positives: 7 (Type I error)

False Negatives: 96 (Type II error)

[70]:
```
cm=confusion_matrix(y_test,y_pred)
conf_matrix=pd.DataFrame(data=cm,columns=['Predicted:0','Predicted:
↪1'],index=['Actual:0','Actual:1'])
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu")
```

[70]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7faf6ef58350>
```



5.5 Model Evaluation

From the below statistics it is clear that the model is highly specific than sensitive. The negative values are predicted more accurately than the positives.

[73]:
```
TN=cm[0,0]
TP=cm[1,1]
FN=cm[1,0]
FP=cm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
```

[74]:
```
print('The acuuracy of the model = TP+TN/(TP+TN+FP+FN) = ',(TP+TN)/
 ↪float(TP+TN+FP+FN),'\n',

'The Missclassification = 1-Accuracy = ',1-((TP+TN)/float(TP+TN+FP+FN)),'\n',

'Sensitivity or True Positive Rate = TP/(TP+FN) = ',TP/float(TP+FN),'\n',

'Specificity or True Negative Rate = TN/(TN+FP) = ',TN/float(TN+FP),'\n',

'Positive Predictive value = TP/(TP+FP) = ',TP/float(TP+FP),'\n',
```

```
'Negative predictive Value = TN/(TN+FN) = ',TN/float(TN+FN),'\n',

'Positive Likelihood Ratio = Sensitivity/(1-Specificity) = ',sensitivity/
↪(1-specificity),'\n',

'Negative likelihood Ratio = (1-Sensitivity)/Specificity = ',(1-sensitivity)/
↪specificity)
```

```
The acuuracy of the model = TP+TN/(TP+TN+FP+FN) =  0.8592896174863388
 The Missclassification = 1-Accuracy =  0.1407103825136612
 Sensitivity or True Positive Rate = TP/(TP+FN) =  0.07692307692307693
 Specificity or True Negative Rate = TN/(TN+FP) =  0.9888535031847133
 Positive Predictive value = TP/(TP+FP) =  0.5333333333333333
 Negative predictive Value = TN/(TN+FN) =  0.8661087866108786
 Positive Likelihood Ratio = Sensitivity/(1-Specificity) =  6.901098901098869
 Negative likelihood Ratio = (1-Sensitivity)/Specificity =  0.9334819769602379
```

5.6 Precision

What are the relevant(what you have predicted and actual are matching.) positive CHD patients out of entire positive CHD predictions.

Calling some people + symptoms some people -. Some are really + and some are really -. Precison is out of positive predictions how many times I was correct. Precision focusing on user prediction. Recall focuses on Actual situation.

[82]:
```
# Precison
Precision = TP/(TP+FP)
Precision
```

[82]:
```
0.5333333333333333
```

5.7 Recall

Fraction of actual CHD patients to the number of CHD patients retrieved from population.

100 patients, some have symptoms, others do not. We predicted some have disease. Recall: Out of all patients that have symptoms, how many were correctly predicted/detect symptoms correctly. Best way to have perfect recall is to say everyone has symptoms. because you won't miss any sick patients.

[83]:
```
# Recall
Recall = TP/(TP+FN)
Recall
```

[83]:
```
0.07692307692307693
```

5.8 F1_Score

Harmonic mean of precision and recall. We consider F1_score because it balances between precision and recall.

[84]:
```
# F1 Score
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
F1_Score
```

[84]:
```
0.13445378151260504
```

[85]:
```
y_pred_prob=logreg.predict_proba(x_test)[:,:]
y_pred_prob_df=pd.DataFrame(data=y_pred_prob, columns=['Prob of no heart␣
 ↪disease (0)','Prob of Heart Disease (1)'])
y_pred_prob_df.head()
```

[85]:

	Prob of no heart disease (0)	Prob of Heart Disease (1)
0	0.960481	0.039519
1	0.965880	0.034120
2	0.609298	0.390702
3	0.819423	0.180577
4	0.807793	0.192207

5.9 Area Under Curve

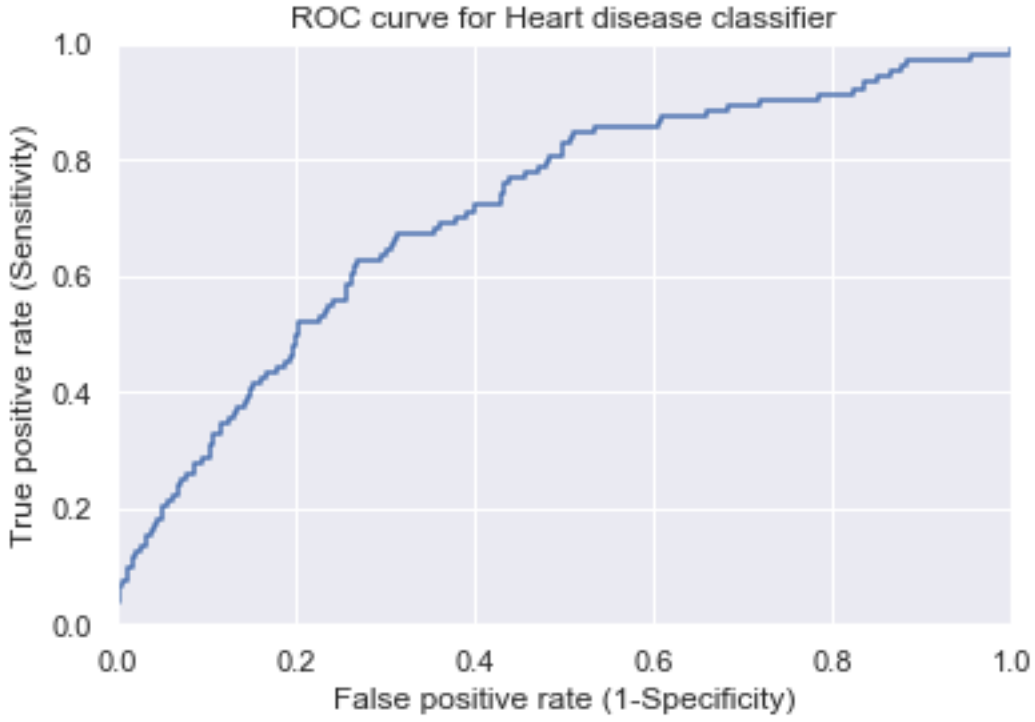
The area under the ROC curve quantifies model classification accuracy; the higher the area, the greater the disparity between true and false positives, and the stronger the model in classifying members of the training dataset. An area of 0.5 corresponds to a model that performs no better than random classification and a good classifier stays as far away from that as possible. An area of 1 is ideal. The closer the AUC to 1 the better.

[86]:
```
from sklearn.preprocessing import binarize

for i in range(1,5):
    cm2=0
    y_pred_prob_yes=logreg.predict_proba(x_test)
    y_pred2=binarize(y_pred_prob_yes,i/10)[:,1]
    cm2=confusion_matrix(y_test,y_pred2)
    print ('With',i/10,'threshold the Confusion Matrix is ','\n',cm2,'\n',
            'with',cm2[0,0]+cm2[1,1],'correct predictions and',cm2[1,0],'Type␣
 ↪II errors( False Negatives)','\n\n',
          'Sensitivity: ',cm2[1,1]/(float(cm2[1,1]+cm2[1,0])),'Specificity:␣
 ↪',cm2[0,0]/(float(cm2[0,0]+cm2[0,1])),'\n\n\n')
```

```
With 0.1 threshold the Confusion Matrix is
 [[299 329]
 [ 16  88]]
 with 387 correct predictions and 16 Type II errors( False Negatives)
```

```
 Sensitivity:  0.8461538461538461 Specificity:  0.47611464968152867



With 0.2 threshold the Confusion Matrix is
 [[486 142]
 [ 50  54]]
 with 540 correct predictions and 50 Type II errors( False Negatives)

 Sensitivity:  0.5192307692307693 Specificity:  0.7738853503184714



With 0.3 threshold the Confusion Matrix is
 [[561  67]
 [ 71  33]]
 with 594 correct predictions and 71 Type II errors( False Negatives)

 Sensitivity:  0.3173076923076923 Specificity:  0.893312101910828



With 0.4 threshold the Confusion Matrix is
 [[603  25]
 [ 86  18]]
 with 621 correct predictions and 86 Type II errors( False Negatives)

 Sensitivity:  0.17307692307692307 Specificity:  0.9601910828025477



/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:70:
FutureWarning: Pass threshold=0.1 as keyword args. From version 0.25 passing
these as positional arguments will result in an error
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:70:
FutureWarning: Pass threshold=0.2 as keyword args. From version 0.25 passing
these as positional arguments will result in an error
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:70:
FutureWarning: Pass threshold=0.3 as keyword args. From version 0.25 passing
these as positional arguments will result in an error
  FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:70:
FutureWarning: Pass threshold=0.4 as keyword args. From version 0.25 passing
these as positional arguments will result in an error
  FutureWarning)
```

[87]:
```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_yes[:,1])
plt.plot(fpr,tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC curve for Heart disease classifier')
plt.xlabel('False positive rate (1-Specificity)')
plt.ylabel('True positive rate (Sensitivity)')
plt.grid(True)
```



[88]:
```
sklearn.metrics.roc_auc_score(y_test,y_pred_prob_yes[:,1])
```

[88]: 0.7144322635962763

6 Conclusion Summary

- Men seem to be more susceptible to heart disease than women.Increase in Age,number of cigarettes smoked per day and systolic Blood Pressure also show increasing odds of having heart disease
- The model predicted with 0.85 accuracy. The model is more specific than sensitive
- The Area under the ROC curve is 71.4 which is somewhat satisfactory.

[ ]: