In [1]:
```
## ANALYSIS OF BREAST CANCER DATA USING LOGISTIC REGRESSION
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pylab
import math
```

In [2]:
```
from scipy.stats import norm
from scipy import stats
import statsmodels.api as sm
from sklearn import datasets, linear_model
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
%matplotlib inline
```

In [3]:
```
BC = pd.read_excel('cancer.xlsx')
```

In [4]:
```
BC.head()
```

Out[4]:

	PAT_ID	Race	MarST	Gender	PatStatus	AgeDiag	Grade	Stability	No.Visits	Lstay	...	LyN
0	32400010	3	1	0	1	52	3	0	5	1	...	
1	32400023	3	1	0	1	48	3	0	4	3	...	
2	32400073	3	0	1	1	61	3	0	9	1	...	
3	32400073	3	1	1	1	63	2	0	3	5	...	
4	32400396	3	0	0	0	69	2	0	7	9	...	

5 rows × 26 columns

In [5]:
```
## We Consider Both Sex

#Import 'train_test_split' from 'sklearn.model_selection'
from sklearn.model_selection import train_test_split

#Import numpy#
import numpy as np
```

In [7]:
```
y = BC.PatStatus
x = BC.drop(['PatStatus','PAT_ID'], axis = 1)
```

In [8]:
```
#Split the data into train and test sets #
x_train, x_test, y_train, y_test=train_test_split(x,y, test_size=0.2, random_stat

## Scaling the data
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler = preprocessing.MinMaxScaler()
x_train_minmax = min_max_scaler.fit_transform(x_train)
x_test_minmax = min_max_scaler.fit_transform(x_test)
```

In [9]:
```
log_reg = sm.Logit(y_train, x_train)
log_reg = log_reg.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.451584
         Iterations 7
```

In [10]:
```
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              PatStatus   No. Observations:                80001
Model:                          Logit   Df Residuals:                    79977
Method:                           MLE   Df Model:                           23
Date:                Thu, 16 Jul 2020   Pseudo R-squ.:                 0.3466
Time:                        00:02:16   Log-Likelihood:                -36127.
converged:                       True   LL-Null:                       -55289.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Race           0.0056      0.001      8.874      0.000       0.004       0.007
MarST          0.0538      0.019      2.895      0.004       0.017       0.090
Gender        -0.0592      0.026     -2.255      0.024      -0.111      -0.008
AgeDiag        0.0066      0.001     10.397      0.000       0.005       0.008
Grade         -0.0159      0.012     -1.286      0.198      -0.040       0.008
Stability      0.2178      0.035      6.280      0.000       0.150       0.286
No.Visits      0.0335      0.003     12.467      0.000       0.028       0.039
Lstay         -0.1740      0.003    -51.229      0.000      -0.181      -0.167
Laterality    -0.0810      0.004    -22.634      0.000      -0.088      -0.074
FamHist       -3.3407      0.063    -53.148      0.000      -3.464      -3.217
PrioBSurgy    -0.3006      0.025    -12.034      0.000      -0.350      -0.252
Suture         0.0427      0.022      1.932      0.053      -0.001       0.086
 Density       0.2365      0.015     16.194      0.000       0.208       0.265
NipRet         2.1587      0.033     64.839      0.000       2.093       2.224
LyNode         0.3620      0.029     12.287      0.000       0.304       0.420
Amorph         0.1636      0.030      5.422      0.000       0.104       0.223
 Size         -0.1563      0.015    -10.446      0.000      -0.186      -0.127
Eggshell      -0.4962      0.024    -20.468      0.000      -0.544      -0.449
Milk           0.8963      0.025     35.627      0.000       0.847       0.946
AxiAden        1.0621      0.021     50.061      0.000       1.021       1.104
Distroph      -0.7367      0.038    -19.439      0.000      -0.811      -0.662
Lucent         1.9973      0.037     53.432      0.000       1.924       2.071
Dermal         0.0292      0.023      1.254      0.210      -0.016       0.075
SkinnLesson   -1.6329      0.020    -80.003      0.000      -1.673      -1.593
==============================================================================
```

In [11]:
```
## exclude insignificant variables in the model
y = BC.PatStatus
x = BC.drop(['PatStatus', 'PAT_ID','Grade','Suture','Dermal'], axis = 1)
```

In [12]:
```
#Split the data into train and test sets #
x_train, x_test, y_train, y_test=train_test_split(x,y, test_size=0.2, random_stat


## Scaling the data
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler = preprocessing.MinMaxScaler()
x_train_minmax = min_max_scaler.fit_transform(x_train)
x_test_minmax = min_max_scaler.fit_transform(x_test)
```

In [13]:
```
## The new fitted logistic regression model with selected variables
import statsmodels.api as sm
log_reg = sm.Logit(y_train, x_train)
log_reg = log_reg.fit()

log_reg1 = LogisticRegression()
log_reg1.fit(x_train_minmax, y_train)
```

```
Optimization terminated successfully.
         Current function value: 0.451626
         Iterations 7
```

Out[13]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [14]:
```
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              PatStatus   No. Observations:                80001
Model:                          Logit   Df Residuals:                    79980
Method:                           MLE   Df Model:                           20
Date:                Thu, 16 Jul 2020   Pseudo R-squ.:                 0.3465
Time:                        00:38:12   Log-Likelihood:                -36131.
converged:                       True   LL-Null:                       -55289.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Race           0.0056      0.001      8.858      0.000       0.004       0.007
MarST          0.0535      0.019      2.882      0.004       0.017       0.090
Gender        -0.0567      0.026     -2.166      0.030      -0.108      -0.005
AgeDiag        0.0065      0.001     10.360      0.000       0.005       0.008
Stability      0.2168      0.035      6.256      0.000       0.149       0.285
No.Visits      0.0336      0.003     12.592      0.000       0.028       0.039
Lstay         -0.1741      0.003    -51.358      0.000      -0.181      -0.167
Laterality    -0.0812      0.004    -22.710      0.000      -0.088      -0.074
FamHist       -3.3429      0.062    -53.584      0.000      -3.465      -3.221
PrioBSurgy    -0.3001      0.025    -12.021      0.000      -0.349      -0.251
 Density       0.2380      0.015     16.366      0.000       0.209       0.266
NipRet         2.1586      0.033     65.045      0.000       2.094       2.224
LyNode         0.3696      0.029     12.677      0.000       0.312       0.427
Amorph         0.1559      0.030      5.216      0.000       0.097       0.214
 Size         -0.1575      0.015    -10.587      0.000      -0.187      -0.128
Eggshell      -0.4944      0.024    -20.452      0.000      -0.542      -0.447
Milk           0.8986      0.025     35.828      0.000       0.849       0.948
AxiAden        1.0612      0.021     50.045      0.000       1.020       1.103
Distroph      -0.7165      0.036    -19.853      0.000      -0.787      -0.646
Lucent         1.9938      0.037     54.102      0.000       1.922       2.066
SkinnLesson   -1.6347      0.020    -80.147      0.000      -1.675      -1.595
==============================================================================
```

In [15]:
```
## Score of the model giving the accuracy of the model
print("Accuracy", (log_reg1.score(x_test_minmax, y_test)))
```

```
Accuracy 0.8024098795060247
```

In [16]:
```
### PREDICTION ON THE TEST DATASET
```

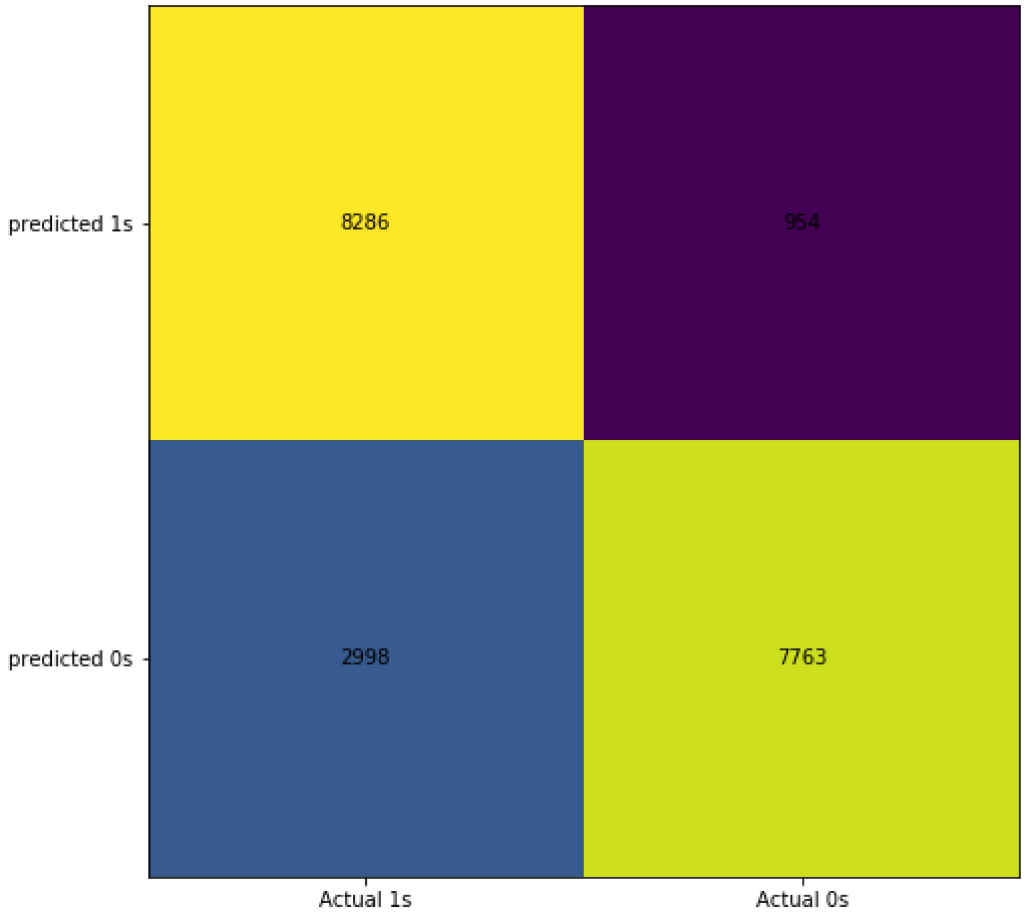
In [17]:
```
### Getting the prediction for the Testing dataset
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_
from sklearn.model_selection import cross_val_score, cross_validate

x_train = x_train_minmax
x_test = x_test_minmax
```

In [18]:
```
tns_probs=[0 for _ in range(len(y_test))]
y_predict = log_reg1.predict(x_test)
```

In [19]:
```
## Keeping the probabilities for Testing outcomes
y_pred = log_reg1.predict_proba(x_test)
y_pred = y_pred[:,1]
```

In [20]:
```
## confusion matrix
test_cm = confusion_matrix(y_test, np.round(y_predict))
fig, ax = plt.subplots(figsize = (8, 8))
ax.imshow(test_cm)
ax.grid(False)
ax.xaxis.set(ticks=(0,1), ticklabels=('Actual 1s', 'Actual 0s'))
ax.yaxis.set(ticks=(0,1), ticklabels=('predicted 1s', 'predicted 0s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, test_cm[i, j], ha= 'center', va= 'center', color= 'black')
plt.show()
```



In [21]:
```
## Error for the prediction for test dataset outcomes
test_error = (test_cm[0,1] + test_cm[1,0])/np.sum(test_cm)
print(test_error)
```

```
0.1975901204939753
```

In [22]:
```
## Accuracy of prediction
1-test_error
```

Out[22]:
```
0.8024098795060247
```

In [23]:
```
## Sensitivity Analysis
test_sens = test_cm[1, 1]/(test_cm[1, 1] + test_cm[0, 1])
print(test_sens)
```

```
0.8905586784444189
```

In [24]:
```
 ## Specificity Analysis
test_spec = test_cm[0, 0]/(test_cm[0, 0]+test_cm[1, 0])
print(test_spec)
```

```
0.7343140730237504
```

In [25]:
```
 ## PPV Analysis
test_npv = test_cm[1, 1]/(test_cm[1, 1] + test_cm[1, 0])
print(test_npv)
```

```
0.7214013567512313
```

In [26]:
```
## NPV Analysis
test_npv = test_cm[0, 0]/(test_cm[0, 0]+test_cm[0, 1])
print(test_npv)
```

```
0.8967532467532467
```

In [27]:
```
## The AUC Score
test_auc = roc_auc_score(y_test, tns_probs)
y_pred_auc = np.round(roc_auc_score(y_test, y_pred), decimals = 2)
```

In [28]:
```
 print(test_auc)
```

```
0.5
```

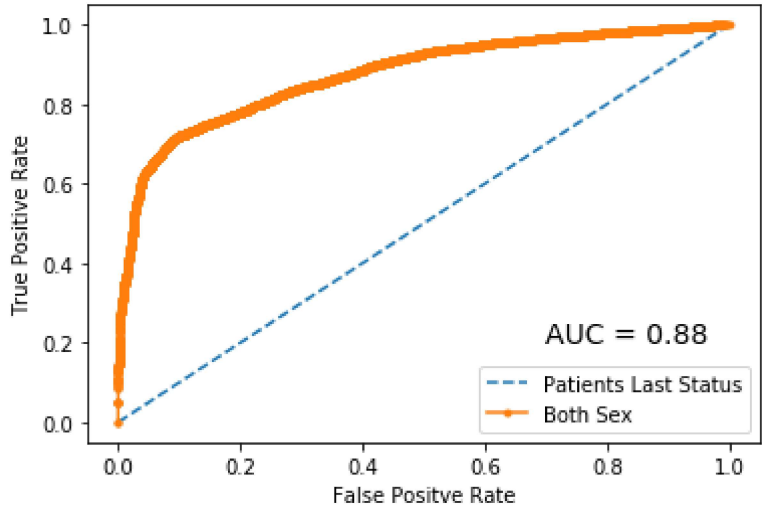
In [29]:
```
print(np.round(y_pred_auc, decimals = 2))
```

```
0.88
```

In [30]:
```
## calculate ROC Curves
test_fpr, test_tpr, _ = roc_curve(y_test, tns_probs)
y_pred_fpr, y_pred_tpr, _ = roc_curve(y_test, y_pred)
```

In [31]:
```
## Plot Curve for the model
import numpy as np
import matplotlib.pyplot as plt
plt.plot(test_fpr, test_tpr, linestyle = '--', label = 'Patients Last Status')
plt.plot(y_pred_fpr, y_pred_tpr, marker = '.', label = 'Both Sex')
plt.text(0.7, 0.2, "AUC = " + str(y_pred_auc), fontsize = 14)
## Axis lable
plt.xlabel("False Positve Rate")
plt.ylabel("True Positive Rate")
## Show Legend
plt.legend()
```

Out[31]: <matplotlib.legend.Legend at 0x1b868388208>



In [ ]:

In [ ]:

In [1]:
```
## WE CONSIDER THE MALE AND FEMALE GENDER SEPERATELY FOR THE ANALYSIS
```

In [2]:
```
## Modules required
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
# Code
BC = (pd.read_excel('cancer.xlsx'))
```

In [4]:
```
#Import 'train_test_split' from 'sklearn.model_selection'
from sklearn.model_selection import train_test_split
#Import numpy#
import numpy as np

## SPLITTING DATA INTO MALE AND FEMALE
BCM=BC[BC.Gender==1]
BCF=BC[BC.Gender==0]
```

In [5]:
```
#### THE MALE DATASET
my=BCM.PatStatus
mx=BCM.drop(['PatStatus','PAT_ID', 'Gender'], axis=1)
```

In [6]:
```
## THE FEMALE DATASET
fy=BCF.PatStatus
fx=BCF.drop(['PatStatus','PAT_ID', 'Gender'],axis=1)
```

In [7]:
```
#Split the Male data into train and test sets #
mx_train, mx_test, my_train, my_test=train_test_split(mx,my, test_size=0.2, randc
```

In [8]:
```
## CONSIDERING THE MALE DATA
mx_train.head()
```

Out[8]:

	Race	MarST	AgeDiag	Grade	Stability	No.Visits	Lstay	Laterality	FamHist	PrioBSurgy
41599	1	1	65	3	0	16	9	2	1	0
25664	1	1	61	3	1	16	9	5	1	0
44540	6	0	74	2	0	10	9	8	1	0
46913	1	0	90	3	0	10	12	3	1	0
31494	1	1	71	3	0	10	9	4	1	0

5 rows × 23 columns

In [9]:
```
mx_test.head()
```

Out[9]:

	Race	MarST	AgeDiag	Grade	Stability	No.Visits	Lstay	Laterality	FamHist	PrioBSurgy
3055	1	0	62	1	0	4	5	5	1	0
30043	1	0	76	3	0	10	9	4	1	0
27137	1	0	85	2	0	10	9	4	1	0
22922	1	1	78	2	0	10	9	8	1	0
44064	1	1	45	2	0	14	9	2	1	0

5 rows × 23 columns

In [10]:
```
mx_train.shape
```

Out[10]:
```
(12479, 23)
```

In [11]:
```
mx_test.shape
```

Out[11]:
```
(3120, 23)
```

In [12]:
```
## Scaling the data
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler = preprocessing.MinMaxScaler()
mx_train_minmax = min_max_scaler.fit_transform(mx_train)
mx_test_minmax = min_max_scaler.fit_transform(mx_test)
```

In [13]:
```
 ## FITTING LOGISTIC REGRESSION FOR MALE DATA
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pylab
import math

from scipy.stats import norm
from scipy import stats
import statsmodels.api as sm
from sklearn import datasets, linear_model
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
%matplotlib inline

from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_
from sklearn.model_selection import cross_val_score, cross_validate
```

In [14]:
```
mlog_reg = sm.Logit(my_train, mx_train)
mlog_reg = mlog_reg.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.410336
         Iterations 8
```

In [15]:
```
print(mlog_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              PatStatus   No. Observations:                12479
Model:                          Logit   Df Residuals:                    12456
Method:                           MLE   Df Model:                           22
Date:                Thu, 16 Jul 2020   Pseudo R-squ.:                 0.3622
Time:                        10:37:40   Log-Likelihood:                -5120.6
converged:                       True   LL-Null:                       -8029.1
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Race           -0.0008      0.002     -0.353      0.724      -0.005       0.004
MarST          -0.0004      0.050     -0.009      0.993      -0.099       0.098
AgeDiag         0.0026      0.002      1.357      0.175      -0.001       0.006
Grade           0.1192      0.035      3.425      0.001       0.051       0.187
Stability       0.0759      0.086      0.883      0.377      -0.093       0.245
No.Visits      -0.0218      0.010     -2.180      0.029      -0.041      -0.002
Lstay          -0.1344      0.012    -10.926      0.000      -0.158      -0.110
Laterality     -0.0121      0.009     -1.288      0.198      -0.030       0.006
FamHist        -2.6038      0.405     -6.423      0.000      -3.398      -1.809
PrioBSurgy     -0.3056      0.131     -2.325      0.020      -0.563      -0.048
Suture          0.1467      0.062      2.373      0.018       0.026       0.268
 Density        0.2654      0.063      4.196      0.000       0.141       0.389
NipRet          0.8825      0.147      6.009      0.000       0.595       1.170
LyNode          2.6183      0.231     11.352      0.000       2.166       3.070
Amorph          1.4480      0.173      8.354      0.000       1.108       1.788
 Size           0.5909      0.068      8.684      0.000       0.458       0.724
Eggshell        0.1048      0.067      1.561      0.118      -0.027       0.236
Milk           -0.8173      0.180     -4.553      0.000      -1.169      -0.465
AxiAden         0.0887      0.063      1.400      0.161      -0.035       0.213
Distroph       -0.1351      0.162     -0.835      0.404      -0.452       0.182
Lucent          0.9937      0.144      6.921      0.000       0.712       1.275
Dermal          0.1767      0.063      2.796      0.005       0.053       0.300
SkinnLesson    -2.7320      0.068    -40.352      0.000      -2.865      -2.599
==============================================================================
```

In [16]:
```
## Exclude insignificant variables in the model
my=BCM.PatStatus
mx=BCM.drop(['PatStatus','PAT_ID', 'Race', 'MarST','AgeDiag', 'Stability', 'Later
```

In [17]:
```
#Split the data into train and test sets #
mx_train, mx_test, my_train, my_test=train_test_split(mx,my, test_size=0.2, rando

## Scaling the data
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler = preprocessing.MinMaxScaler()
mx_train_minmax = min_max_scaler.fit_transform(mx_train)
mx_test_minmax = min_max_scaler.fit_transform(mx_test)
```

In [18]:
```
## The new fitted logistic regression model with selected variables
import statsmodels.api as sm
mlog_reg = sm.Logit(my_train, mx_train)
mlog_reg = mlog_reg.fit()

mlog_reg1 = LogisticRegression()
mlog_reg1.fit(mx_train_minmax, my_train)
```

```
Optimization terminated successfully.
         Current function value: 0.410184
         Iterations 8

C:\Users\eagye\anaconda1\lib\site-packages\sklearn\linear_model\_logistic.py:94
0: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[18]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [19]:
```
print(mlog_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              PatStatus   No. Observations:                12479
Model:                          Logit   Df Residuals:                    12463
Method:                           MLE   Df Model:                           15
Date:                Thu, 16 Jul 2020   Pseudo R-squ.:                 0.3625
Time:                        11:02:20   Log-Likelihood:                -5118.7
converged:                       True   LL-Null:                       -8029.1
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Gender          3.2186      1.137      2.832      0.005       0.991       5.446
Grade           0.1079      0.035      3.086      0.002       0.039       0.176
No.Visits      -0.0300      0.009     -3.302      0.001      -0.048      -0.012
Lstay          -0.1381      0.012    -11.135      0.000      -0.162      -0.114
FamHist        -4.7898      1.066     -4.493      0.000      -6.879      -2.700
PrioBSurgy     -0.3844      0.133     -2.888      0.004      -0.645      -0.124
Suture          0.1390      0.062      2.251      0.024       0.018       0.260
 Density        0.2094      0.065      3.204      0.001       0.081       0.338
NipRet          0.8124      0.149      5.441      0.000       0.520       1.105
LyNode          2.6812      0.220     12.215      0.000       2.251       3.111
Amorph          1.3794      0.173      7.975      0.000       1.040       1.718
 Size           0.5364      0.069      7.803      0.000       0.402       0.671
Milk           -0.9406      0.185     -5.089      0.000      -1.303      -0.578
Lucent          0.7765      0.159      4.895      0.000       0.466       1.087
Dermal          0.1519      0.059      2.560      0.010       0.036       0.268
SkinnLesson    -2.7614      0.069    -40.273      0.000      -2.896      -2.627
==============================================================================
```

In [20]:
```
## Score of the model giving the accuracy of the model
print("Accuracy", (mlog_reg1.score(mx_test_minmax, my_test)))
```

```
Accuracy 0.8233974358974359
```

In [21]:
```
### PREDICTION ON THE TEST DATASET
```

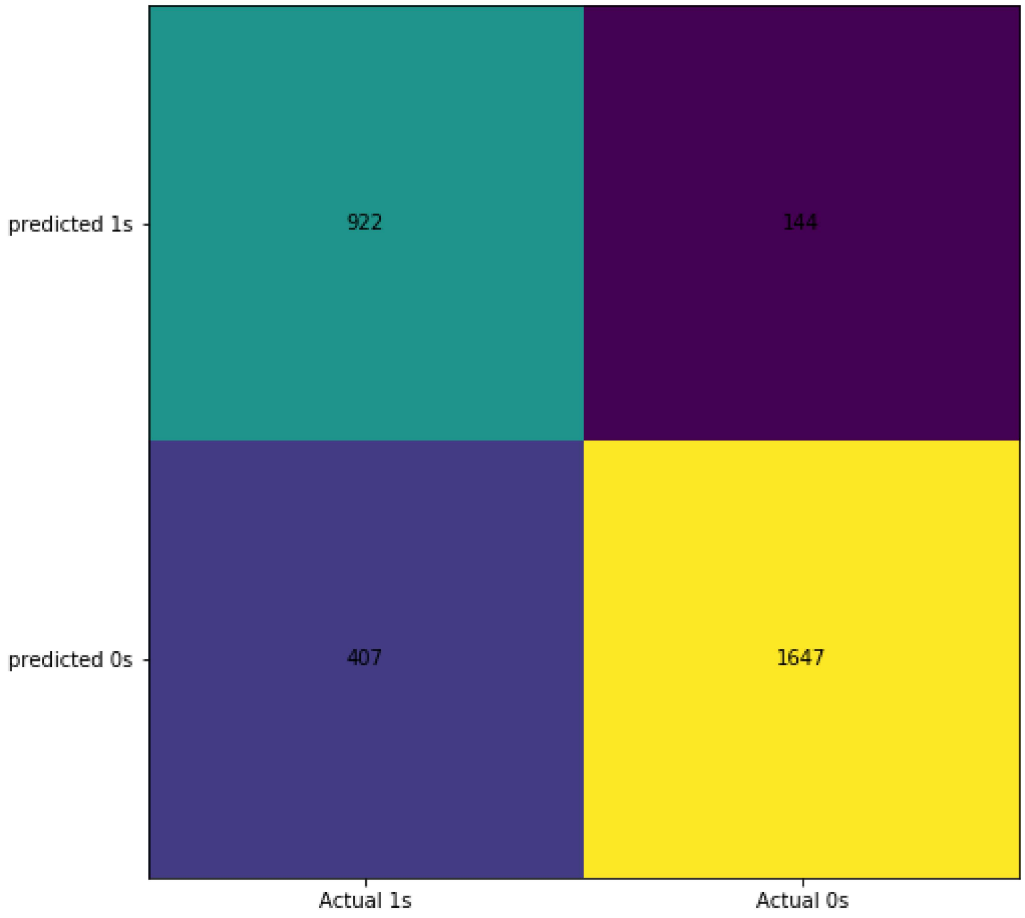
In [22]:
```
### Getting the prediction for the Testing dataset
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_
from sklearn.model_selection import cross_val_score, cross_validate

mx_train = mx_train_minmax
mx_test = mx_test_minmax
```

In [23]:
```
tns_probs=[0 for _ in range(len(my_test))]
my_predict = mlog_reg1.predict(mx_test)
```

In [24]:
```
## Keeping the probabilities for Testing outcomes
my_pred = mlog_reg1.predict_proba(mx_test)
my_pred = my_pred[:, 1]
```

In [25]:
```
 ## confusion matrix
mtest_cm = confusion_matrix(my_test, np.round(my_predict))
fig, ax = plt.subplots(figsize = (8, 8))
ax.imshow(mtest_cm)
ax.grid(False)
ax.xaxis.set(ticks=(0,1), ticklabels=('Actual 1s', 'Actual 0s'))
ax.yaxis.set(ticks=(0,1), ticklabels=('predicted 1s', 'predicted 0s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, mtest_cm[i, j], ha= 'center', va= 'center', color= 'black')
plt.show()
```



In [26]:
```
 ## Error for the prediction for test dataset outcomes
mtest_error = (mtest_cm[0,1] + mtest_cm[1,0])/np.sum(mtest_cm)
print(mtest_error)
```

```
0.1766025641025641
```

In [27]:
```
## Accuracy of prediction
1-mtest_error
```

Out[27]:
```
0.8233974358974359
```

In [28]:
```
## Sensitivity Analysis
mtest_sens = mtest_cm[1, 1]/(mtest_cm[1, 1] + mtest_cm[0, 1])
print(mtest_sens)
```

```
0.9195979899497487
```

In [29]:
```
 ## Specificity Analysis
mtest_spec = mtest_cm[0, 0]/(mtest_cm[0, 0] + mtest_cm[1, 0])
print(mtest_spec)
```

```
0.6937547027840482
```

In [30]:
```
 ## PPV Analysis
mtest_npv = mtest_cm[1, 1]/(mtest_cm[1, 1] + mtest_cm[1, 0])
print(mtest_npv)
```

```
0.8018500486854917
```

In [31]:
```
## NPV Analysis
mtest_npv = mtest_cm[0, 0]/(mtest_cm[0, 0] + mtest_cm[0, 1])
print(mtest_npv)
```

```
0.8649155722326454
```

In [32]:
```
## The AUC Score
mtest_auc = roc_auc_score(my_test, tns_probs)
my_pred_auc = np.round(roc_auc_score(my_test, my_pred), decimals = 2)
```

In [33]:
```
 print(mtest_auc)
```

```
0.5
```

In [34]:
```
print(np.round(my_pred_auc, decimals = 2))
```

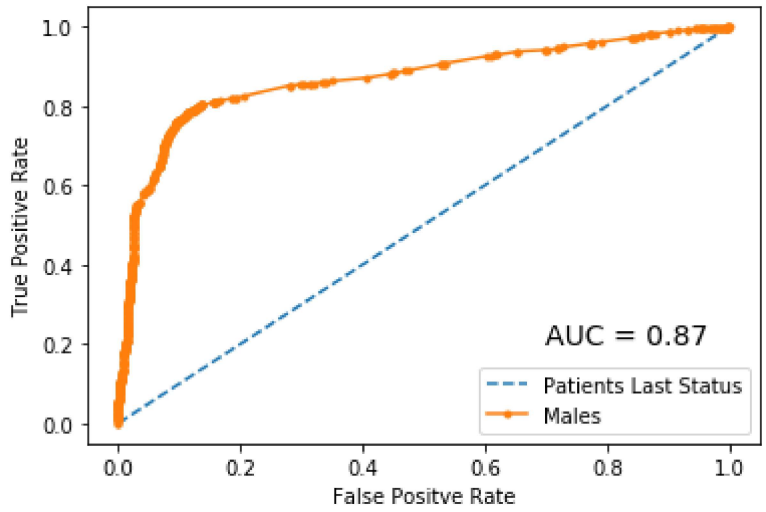
```
0.87
```

In [35]:
```
## calculate ROC Curves
mtest_fpr, mtest_tpr, _ = roc_curve(my_test, tns_probs)
my_pred_fpr, my_pred_tpr, _ = roc_curve(my_test, my_pred)
```

In [37]:
```
## Plot Curve for the model
import numpy as np
import matplotlib.pyplot as plt

plt.plot(mtest_fpr, mtest_tpr, linestyle = '--', label = 'Patients Last Status')
plt.plot(my_pred_fpr, my_pred_tpr, marker = '.', label = 'Males')
plt.text(0.7, 0.2, "AUC = " + str(my_pred_auc), fontsize = 14)
## Axis lable
plt.xlabel("False Positve Rate")
plt.ylabel("True Positive Rate")
## Show Legend
plt.legend()
```

Out[37]: <matplotlib.legend.Legend at 0x188f82b6e48>



In [ ]:

In [ ]:

In [ ]:

In [1]:
```
## CONSIDERING THE FEMALE DATA
## The new fitted logistic regression model with selected variables
## Modules required
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
BC = (pd.read_excel('cancer.xlsx'))
BCF=BC[BC.Gender==0]

fy=BCF.PatStatus
fx=BCF.drop(['PatStatus','PAT_ID', 'Gender'],axis=1)
```

In [2]:
```
#Import 'train_test_split' from 'sklearn.model_selection'
from sklearn.model_selection import train_test_split

#Import numpy#
import numpy as np
#Split the Male data into train and test sets #
fx_train, fx_test, fy_train, fy_test=train_test_split(fx,fy, test_size=0.2, randc
```

In [3]:
```
# Scaling the data
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler = preprocessing.MinMaxScaler()
fx_train_minmax = min_max_scaler.fit_transform(fx_train)
fx_test_minmax = min_max_scaler.fit_transform(fx_test)
```

In [4]:
```
 ## FITTING LOGISTIC REGRESSION FOR FEMALE DATA
from scipy.stats import norm
from scipy import stats
import statsmodels.api as sm
from sklearn import datasets, linear_model
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
%matplotlib inline

from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_
from sklearn.model_selection import cross_val_score, cross_validate
```

In [5]:
```
flog_reg = sm.Logit(fy_train, fx_train)
flog_reg = flog_reg.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.433132
         Iterations 7
```

In [6]:
```
print(flog_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              PatStatus   No. Observations:                67522
Model:                          Logit   Df Residuals:                    67499
Method:                           MLE   Df Model:                           22
Date:                Thu, 16 Jul 2020   Pseudo R-squ.:                 0.3749
Time:                        12:56:46   Log-Likelihood:                -29246.
converged:                       True   LL-Null:                       -46789.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Race           0.0059      0.001      9.034      0.000       0.005       0.007
MarST          0.0730      0.021      3.523      0.000       0.032       0.114
AgeDiag        0.0078      0.001     11.084      0.000       0.006       0.009
Grade         -0.0393      0.014     -2.863      0.004      -0.066      -0.012
Stability      0.1699      0.039      4.314      0.000       0.093       0.247
No.Visits      0.0507      0.003     17.406      0.000       0.045       0.056
Lstay         -0.1656      0.004    -45.392      0.000      -0.173      -0.158
Laterality    -0.0942      0.004    -23.309      0.000      -0.102      -0.086
FamHist       -3.2622      0.064    -50.609      0.000      -3.388      -3.136
PrioBSurgy    -0.2662      0.026    -10.256      0.000      -0.317      -0.215
Suture        -0.0323      0.024     -1.321      0.187      -0.080       0.016
 Density       0.1879      0.015     12.304      0.000       0.158       0.218
NipRet         2.2790      0.035     64.601      0.000       2.210       2.348
LyNode         0.1978      0.031      6.426      0.000       0.137       0.258
Amorph        -0.1152      0.033     -3.479      0.001      -0.180      -0.050
 Size         -0.2425      0.016    -14.893      0.000      -0.274      -0.211
Eggshell      -0.5637      0.027    -20.885      0.000      -0.617      -0.511
Milk           0.9166      0.026     35.257      0.000       0.866       0.968
AxiAden        1.2250      0.023     52.419      0.000       1.179       1.271
Distroph      -0.6857      0.040    -17.217      0.000      -0.764      -0.608
Lucent         2.0236      0.040     51.090      0.000       1.946       2.101
Dermal        -0.0202      0.026     -0.775      0.438      -0.071       0.031
SkinnLesson   -1.4800      0.022    -66.078      0.000      -1.524      -1.436
==============================================================================
```

In [7]:
```
### The new fitted logistic regression model with selected variables
## Modules required
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
BC = (pd.read_excel('cancer.xlsx'))
BCF=BC[BC.Gender==0]

fy=BCF.PatStatus
fx=BCF.drop(['PatStatus','PAT_ID', 'Gender','Suture','Dermal'],axis=1)
```

In [8]:
```
#Import 'train_test_split' from 'sklearn.model_selection'
from sklearn.model_selection import train_test_split

#Import numpy#
import numpy as np
#Split the Male data into train and test sets #
fx_train, fx_test, fy_train, fy_test=train_test_split(fx,fy, test_size=0.2, randc
```

In [9]:
```
fx_train.head()
```

Out[9]:

	Race	MarST	AgeDiag	Grade	Stability	No.Visits	Lstay	Laterality	FamHist	PrioBSurgy
88023	1	0	42	3	0	9	15	4	1	1
92578	1	0	66	1	0	8	8	3	1	1
75760	1	1	61	3	0	9	9	5	1	1
51727	1	1	50	1	0	15	10	4	1	1
3377	1	1	78	2	1	5	2	8	1	1

5 rows × 21 columns

In [10]:
```
fx_test.head()
```

Out[10]:

	Race	MarST	AgeDiag	Grade	Stability	No.Visits	Lstay	Laterality	FamHist	PrioBSurgy
41774	1	1	67	2	0	16	9	2	1	0
84188	16	1	41	3	0	8	8	8	1	1
55207	1	1	50	1	0	6	10	9	1	0
51157	1	1	78	3	0	1	10	9	1	1
99358	1	1	77	2	0	10	12	3	1	0

5 rows × 21 columns

In [11]:
```
fx_train.shape
```

Out[11]:
```
(67522, 21)
```

In [12]:
```
fx_test.shape
```

Out[12]:
```
(16881, 21)
```

In [13]:
```
# Scaling the data
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler = preprocessing.MinMaxScaler()
fx_train_minmax = min_max_scaler.fit_transform(fx_train)
fx_test_minmax = min_max_scaler.fit_transform(fx_test)
```

In [14]:
```
 ## FITTING LOGISTIC REGRESSION FOR FEMALE DATA
from scipy.stats import norm
from scipy import stats
import statsmodels.api as sm
from sklearn import datasets, linear_model
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
%matplotlib inline

from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_
from sklearn.model_selection import cross_val_score, cross_validate
```

In [15]:
```
flog_reg = sm.Logit(fy_train, fx_train)
flog_reg = flog_reg.fit()

flog_reg1 = LogisticRegression()
flog_reg1.fit(fx_train, fy_train)
```

```
Optimization terminated successfully.
         Current function value: 0.433149
         Iterations 7

C:\Users\eagye\anaconda1\lib\site-packages\sklearn\linear_model\_logistic.py:94
0: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[15]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [16]:
```
print(flog_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              PatStatus   No. Observations:                67522
Model:                          Logit   Df Residuals:                    67501
Method:                           MLE   Df Model:                           20
Date:                Thu, 16 Jul 2020   Pseudo R-squ.:                 0.3749
Time:                        12:59:29   Log-Likelihood:                -29247.
converged:                       True   LL-Null:                       -46789.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Race           0.0059      0.001      9.026      0.000       0.005       0.007
MarST          0.0728      0.021      3.513      0.000       0.032       0.113
AgeDiag        0.0078      0.001     11.070      0.000       0.006       0.009
Grade         -0.0393      0.014     -2.862      0.004      -0.066      -0.012
Stability      0.1703      0.039      4.325      0.000       0.093       0.247
No.Visits      0.0504      0.003     17.358      0.000       0.045       0.056
Lstay         -0.1658      0.004    -45.434      0.000      -0.173      -0.159
Laterality    -0.0941      0.004    -23.298      0.000      -0.102      -0.086
FamHist       -3.2673      0.064    -50.755      0.000      -3.393      -3.141
PrioBSurgy    -0.2673      0.026    -10.311      0.000      -0.318      -0.216
 Density       0.1860      0.015     12.236      0.000       0.156       0.216
NipRet         2.2776      0.035     64.651      0.000       2.209       2.347
LyNode         0.1914      0.030      6.281      0.000       0.132       0.251
Amorph        -0.1097      0.033     -3.348      0.001      -0.174      -0.045
 Size         -0.2427      0.016    -14.960      0.000      -0.275      -0.211
Eggshell      -0.5663      0.027    -21.028      0.000      -0.619      -0.514
Milk           0.9140      0.026     35.258      0.000       0.863       0.965
AxiAden        1.2251      0.023     52.431      0.000       1.179       1.271
Distroph      -0.6975      0.038    -18.382      0.000      -0.772      -0.623
Lucent         2.0218      0.039     51.615      0.000       1.945       2.099
SkinnLesson   -1.4795      0.022    -66.083      0.000      -1.523      -1.436
==============================================================================
```

In [17]:
```
## Score of the model giving the accuracy of the model
print("Accuracy", (flog_reg1.score(fx_test_minmax, fy_test)))
```

```
Accuracy 0.7837213435223032
```

In [18]:
```
### PREDICTION ON THE TEST DATASET
```

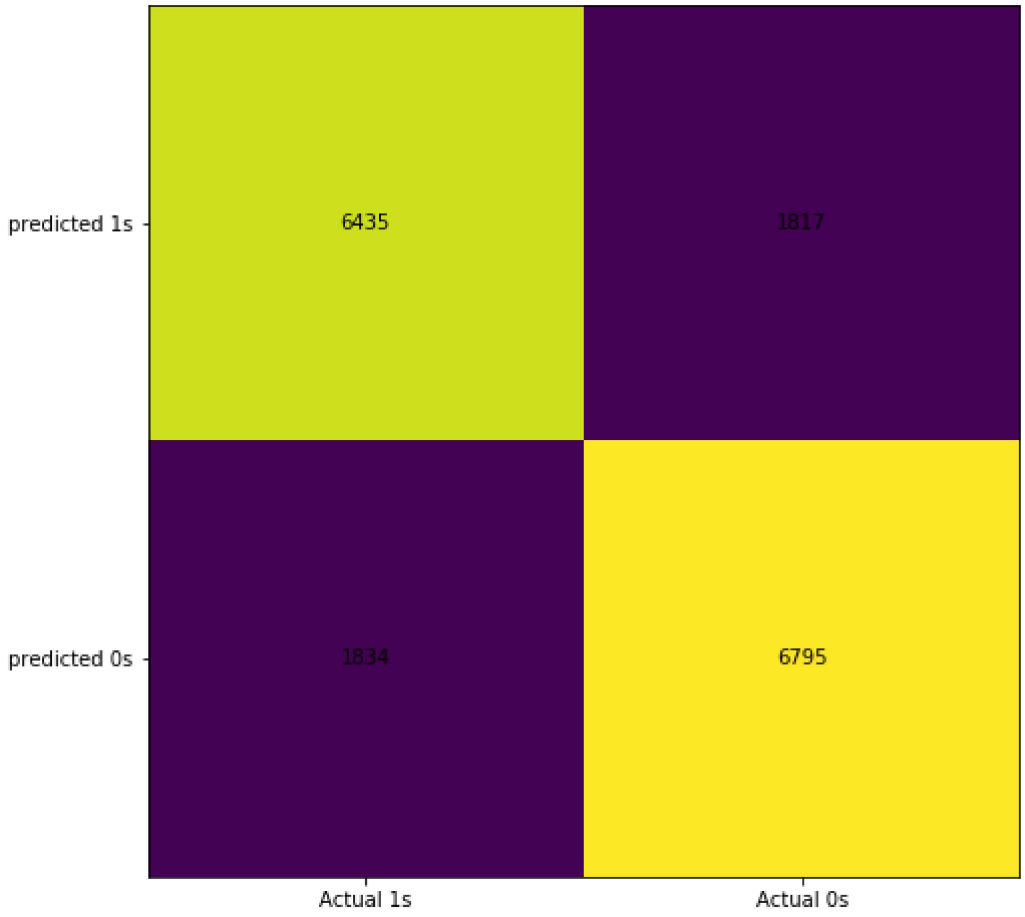
In [20]:
```
### Getting the prediction for the Testing dataset
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_
from sklearn.model_selection import cross_val_score, cross_validate

fx_train = fx_train_minmax
fx_test = fx_test_minmax
```

In [24]:
```
tns_probs=[0 for _ in range(len(fy_test))]
fy_predict = flog_reg1.predict(fx_test)
```

In [25]:
```
## Keeping the probabilities for Testing outcomes
fy_pred = flog_reg1.predict_proba(fx_test)
fy_pred = fy_pred[:, 1]
```

In [26]:
```
 ## confusion matrix
ftest_cm = confusion_matrix(fy_test, np.round(fy_predict))
fig, ax = plt.subplots(figsize = (8, 8))
ax.imshow(ftest_cm)
ax.grid(False)
ax.xaxis.set(ticks=(0,1), ticklabels=('Actual 1s', 'Actual 0s'))
ax.yaxis.set(ticks=(0,1), ticklabels=('predicted 1s', 'predicted 0s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, ftest_cm[i, j], ha= 'center', va= 'center', color= 'black')
plt.show()
```



In [27]:
```
## Error for the prediction for test dataset outcomes
ftest_error = (ftest_cm[0,1] + ftest_cm[1,0])/np.sum(ftest_cm)
print(ftest_error)
```

```
0.2162786564776968
```

In [28]:
```
## Accuracy of prediction
1-ftest_error
```

Out[28]:
```
0.7837213435223032
```

In [29]:
```
## Sensitivity Analysis
ftest_sens = ftest_cm[1, 1]/(ftest_cm[1, 1] + ftest_cm[0, 1])
print(ftest_sens)
```

```
0.7890153274500696
```

In [30]:
```
## Specificity Analysis
ftest_spec = ftest_cm[0, 0]/(ftest_cm[0, 0] + ftest_cm[1, 0])
print(ftest_spec)
```

```
0.7782077639375983
```

In [31]:
```
## PPV Analysis
ftest_npv = ftest_cm[1, 1]/(ftest_cm[1, 1] + ftest_cm[1, 0])
print(ftest_npv)
```

```
0.7874608877042532
```

In [32]:
```
## NPV Analysis
ftest_npv = ftest_cm[0, 0]/(ftest_cm[0, 0] + ftest_cm[0, 1])
print(ftest_npv)
```

```
0.7798109549200194
```

In [33]:
```
## The AUC Score
ftest_auc = roc_auc_score(fy_test, tns_probs)
fy_pred_auc = np.round(roc_auc_score(fy_test, fy_pred), decimals = 2)
```

In [34]:
```
 print(ftest_auc)
```

```
0.5
```

In [35]:
```
print(np.round(fy_pred_auc, decimals = 2))
```

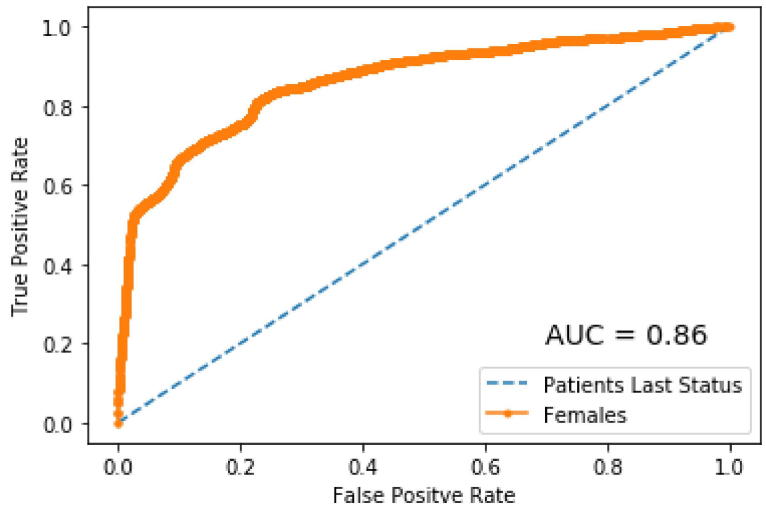
```
0.86
```

In [36]:
```
## calculate ROC Curves
ftest_fpr, ftest_tpr, _ = roc_curve(fy_test, tns_probs)
fy_pred_fpr, fy_pred_tpr, _ = roc_curve(fy_test, fy_pred)
```

In [37]:
```
## Plot Curve for the model
import numpy as np
import matplotlib.pyplot as plt

plt.plot(ftest_fpr, ftest_tpr, linestyle = '--', label = 'Patients Last Status')
plt.plot(fy_pred_fpr, fy_pred_tpr, marker = '.', label = 'Females')
plt.text(0.7, 0.2, "AUC = " + str(fy_pred_auc), fontsize = 14)
## Axis lable
plt.xlabel("False Positve Rate")
plt.ylabel("True Positive Rate")
## Show Legend
plt.legend()
```

Out[37]: <matplotlib.legend.Legend at 0x1eb06c904c8>



In [ ]:

In [ ]:

In [1]:
```
## ANALYSIS OF BREAST CANCER DATA USING LOGISTIC REGRESSION
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pylab
import math
```

In [2]:
```
from scipy.stats import norm
from scipy import stats
import statsmodels.api as sm
from sklearn import datasets, linear_model
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
%matplotlib inline
```

In [3]:
```
BC = pd.read_excel('cancer.xlsx')
```

In [4]:
```
BC.head()
```

Out[4]:

	PAT_ID	Race	MarST	Gender	PatStatus	AgeDiag	Grade	Stability	No.Visits	Lstay	...	LyN
0	32400010	3	1	0	1	52	3	0	5	1	...	
1	32400023	3	1	0	1	48	3	0	4	3	...	
2	32400073	3	0	1	1	61	3	0	9	1	...	
3	32400073	3	1	1	1	63	2	0	3	5	...	
4	32400396	3	0	0	0	69	2	0	7	9	...	

5 rows × 26 columns

In [5]:
```
## We Consider Both Sex

#Import 'train_test_split' from 'sklearn.model_selection'
from sklearn.model_selection import train_test_split

#Import numpy#
import numpy as np
```

In [7]:
```
y = BC.PatStatus
x = BC.drop(['PatStatus','PAT_ID'], axis = 1)
```

In [8]:
```
#Split the data into train and test sets #
x_train, x_test, y_train, y_test=train_test_split(x,y, test_size=0.2, random_stat

## Scaling the data
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler = preprocessing.MinMaxScaler()
x_train_minmax = min_max_scaler.fit_transform(x_train)
x_test_minmax = min_max_scaler.fit_transform(x_test)
```

In [9]:
```
log_reg = sm.Logit(y_train, x_train)
log_reg = log_reg.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.451584
         Iterations 7
```

In [10]:
```
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              PatStatus   No. Observations:                80001
Model:                          Logit   Df Residuals:                    79977
Method:                           MLE   Df Model:                           23
Date:                Thu, 16 Jul 2020   Pseudo R-squ.:                 0.3466
Time:                        00:02:16   Log-Likelihood:                -36127.
converged:                       True   LL-Null:                       -55289.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Race           0.0056      0.001      8.874      0.000       0.004       0.007
MarST          0.0538      0.019      2.895      0.004       0.017       0.090
Gender        -0.0592      0.026     -2.255      0.024      -0.111      -0.008
AgeDiag        0.0066      0.001     10.397      0.000       0.005       0.008
Grade         -0.0159      0.012     -1.286      0.198      -0.040       0.008
Stability      0.2178      0.035      6.280      0.000       0.150       0.286
No.Visits      0.0335      0.003     12.467      0.000       0.028       0.039
Lstay         -0.1740      0.003    -51.229      0.000      -0.181      -0.167
Laterality    -0.0810      0.004    -22.634      0.000      -0.088      -0.074
FamHist       -3.3407      0.063    -53.148      0.000      -3.464      -3.217
PrioBSurgy    -0.3006      0.025    -12.034      0.000      -0.350      -0.252
Suture         0.0427      0.022      1.932      0.053      -0.001       0.086
 Density       0.2365      0.015     16.194      0.000       0.208       0.265
NipRet         2.1587      0.033     64.839      0.000       2.093       2.224
LyNode         0.3620      0.029     12.287      0.000       0.304       0.420
Amorph         0.1636      0.030      5.422      0.000       0.104       0.223
 Size         -0.1563      0.015    -10.446      0.000      -0.186      -0.127
Eggshell      -0.4962      0.024    -20.468      0.000      -0.544      -0.449
Milk           0.8963      0.025     35.627      0.000       0.847       0.946
AxiAden        1.0621      0.021     50.061      0.000       1.021       1.104
Distroph      -0.7367      0.038    -19.439      0.000      -0.811      -0.662
Lucent         1.9973      0.037     53.432      0.000       1.924       2.071
Dermal         0.0292      0.023      1.254      0.210      -0.016       0.075
SkinnLesson   -1.6329      0.020    -80.003      0.000      -1.673      -1.593
==============================================================================
```

In [11]:
```
## exclude insignificant variables in the model
y = BC.PatStatus
x = BC.drop(['PatStatus', 'PAT_ID','Grade','Suture','Dermal'], axis = 1)
```

In [12]:
```
#Split the data into train and test sets #
x_train, x_test, y_train, y_test=train_test_split(x,y, test_size=0.2, random_stat


## Scaling the data
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler = preprocessing.MinMaxScaler()
x_train_minmax = min_max_scaler.fit_transform(x_train)
x_test_minmax = min_max_scaler.fit_transform(x_test)
```

In [13]:
```
## The new fitted logistic regression model with selected variables
import statsmodels.api as sm
log_reg = sm.Logit(y_train, x_train)
log_reg = log_reg.fit()

log_reg1 = LogisticRegression()
log_reg1.fit(x_train_minmax, y_train)
```

```
Optimization terminated successfully.
         Current function value: 0.451626
         Iterations 7
```

Out[13]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [14]:
```
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              PatStatus   No. Observations:                80001
Model:                          Logit   Df Residuals:                    79980
Method:                           MLE   Df Model:                           20
Date:                Thu, 16 Jul 2020   Pseudo R-squ.:                 0.3465
Time:                        00:38:12   Log-Likelihood:                -36131.
converged:                       True   LL-Null:                       -55289.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Race           0.0056      0.001      8.858      0.000       0.004       0.007
MarST          0.0535      0.019      2.882      0.004       0.017       0.090
Gender        -0.0567      0.026     -2.166      0.030      -0.108      -0.005
AgeDiag        0.0065      0.001     10.360      0.000       0.005       0.008
Stability      0.2168      0.035      6.256      0.000       0.149       0.285
No.Visits      0.0336      0.003     12.592      0.000       0.028       0.039
Lstay         -0.1741      0.003    -51.358      0.000      -0.181      -0.167
Laterality    -0.0812      0.004    -22.710      0.000      -0.088      -0.074
FamHist       -3.3429      0.062    -53.584      0.000      -3.465      -3.221
PrioBSurgy    -0.3001      0.025    -12.021      0.000      -0.349      -0.251
 Density       0.2380      0.015     16.366      0.000       0.209       0.266
NipRet         2.1586      0.033     65.045      0.000       2.094       2.224
LyNode         0.3696      0.029     12.677      0.000       0.312       0.427
Amorph         0.1559      0.030      5.216      0.000       0.097       0.214
 Size         -0.1575      0.015    -10.587      0.000      -0.187      -0.128
Eggshell      -0.4944      0.024    -20.452      0.000      -0.542      -0.447
Milk           0.8986      0.025     35.828      0.000       0.849       0.948
AxiAden        1.0612      0.021     50.045      0.000       1.020       1.103
Distroph      -0.7165      0.036    -19.853      0.000      -0.787      -0.646
Lucent         1.9938      0.037     54.102      0.000       1.922       2.066
SkinnLesson   -1.6347      0.020    -80.147      0.000      -1.675      -1.595
==============================================================================
```

In [15]:
```
## Score of the model giving the accuracy of the model
print("Accuracy", (log_reg1.score(x_test_minmax, y_test)))
```

```
Accuracy 0.8024098795060247
```

In [16]:
```
### PREDICTION ON THE TEST DATASET
```

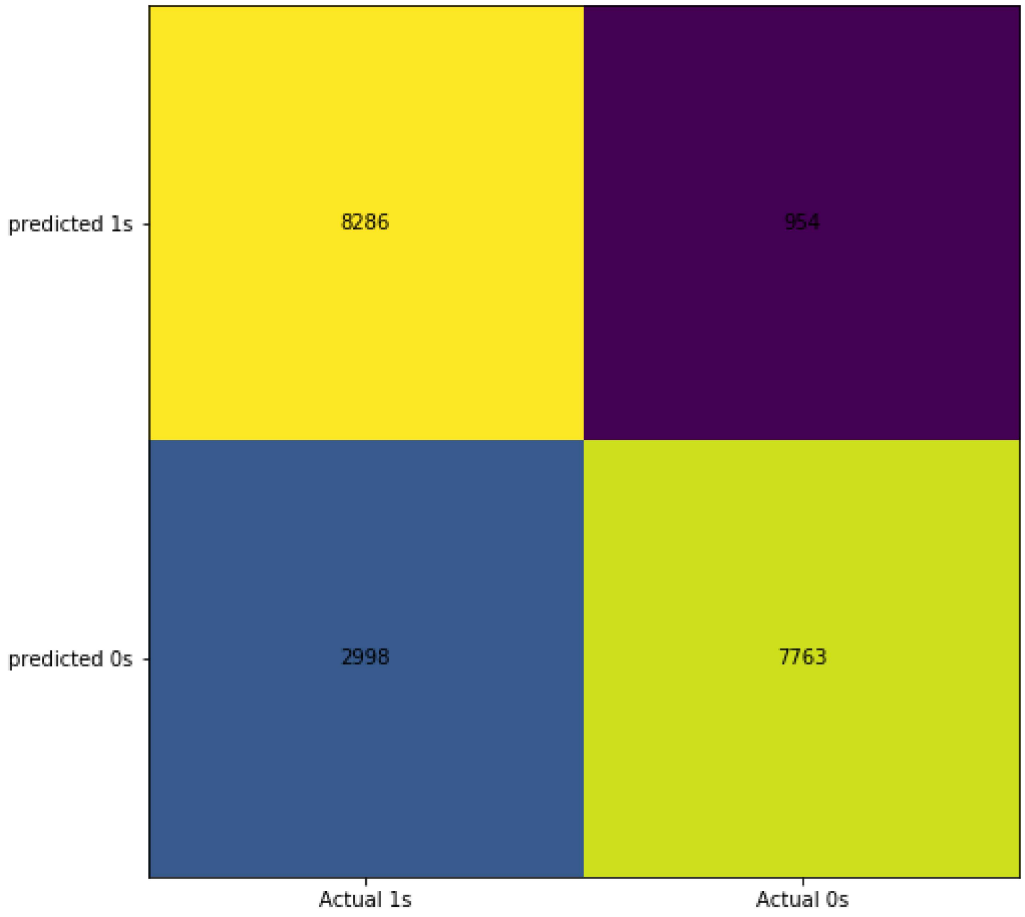
In [17]:
```
### Getting the prediction for the Testing dataset
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_
from sklearn.model_selection import cross_val_score, cross_validate

x_train = x_train_minmax
x_test = x_test_minmax
```

In [18]:
```
tns_probs=[0 for _ in range(len(y_test))]
y_predict = log_reg1.predict(x_test)
```

In [19]:
```
## Keeping the probabilities for Testing outcomes
y_pred = log_reg1.predict_proba(x_test)
y_pred = y_pred[:,1]
```

In [20]:
```
## confusion matrix
test_cm = confusion_matrix(y_test, np.round(y_predict))
fig, ax = plt.subplots(figsize = (8, 8))
ax.imshow(test_cm)
ax.grid(False)
ax.xaxis.set(ticks=(0,1), ticklabels=('Actual 1s', 'Actual 0s'))
ax.yaxis.set(ticks=(0,1), ticklabels=('predicted 1s', 'predicted 0s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, test_cm[i, j], ha= 'center', va= 'center', color= 'black')
plt.show()
```



In [21]:
```
## Error for the prediction for test dataset outcomes
test_error = (test_cm[0,1] + test_cm[1,0])/np.sum(test_cm)
print(test_error)
```

```
0.1975901204939753
```

In [22]:
```
## Accuracy of prediction
1-test_error
```

Out[22]:
```
0.8024098795060247
```

In [23]:
```
## Sensitivity Analysis
test_sens = test_cm[1, 1]/(test_cm[1, 1] + test_cm[0, 1])
print(test_sens)
```

```
0.8905586784444189
```

In [24]:
```
 ## Specificity Analysis
test_spec = test_cm[0, 0]/(test_cm[0, 0]+test_cm[1, 0])
print(test_spec)
```

```
0.7343140730237504
```

In [25]:
```
 ## PPV Analysis
test_npv = test_cm[1, 1]/(test_cm[1, 1] + test_cm[1, 0])
print(test_npv)
```

```
0.7214013567512313
```

In [26]:
```
## NPV Analysis
test_npv = test_cm[0, 0]/(test_cm[0, 0]+test_cm[0, 1])
print(test_npv)
```

```
0.8967532467532467
```

In [27]:
```
## The AUC Score
test_auc = roc_auc_score(y_test, tns_probs)
y_pred_auc = np.round(roc_auc_score(y_test, y_pred), decimals = 2)
```

In [28]:
```
 print(test_auc)
```

```
0.5
```

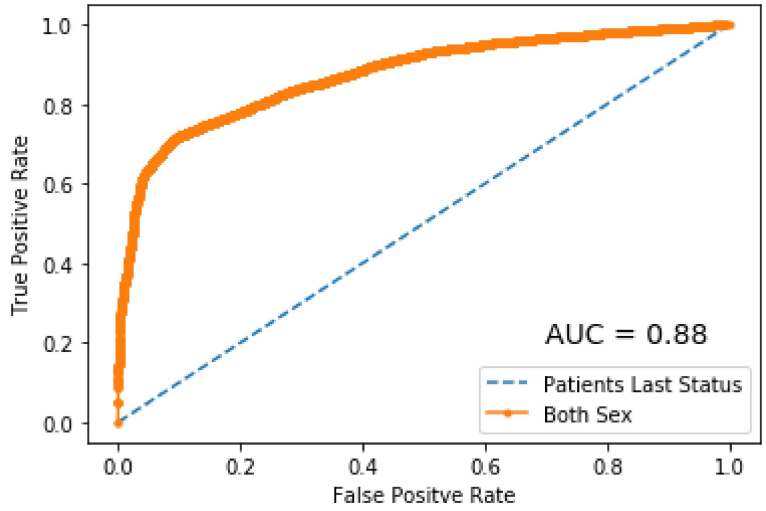
In [29]:
```
print(np.round(y_pred_auc, decimals = 2))
```

```
0.88
```

In [30]:
```
## calculate ROC Curves
test_fpr, test_tpr, _ = roc_curve(y_test, tns_probs)
y_pred_fpr, y_pred_tpr, _ = roc_curve(y_test, y_pred)
```

In [31]:
```
## Plot Curve for the model
import numpy as np
import matplotlib.pyplot as plt
plt.plot(test_fpr, test_tpr, linestyle = '--', label = 'Patients Last Status')
plt.plot(y_pred_fpr, y_pred_tpr, marker = '.', label = 'Both Sex')
plt.text(0.7, 0.2, "AUC = " + str(y_pred_auc), fontsize = 14)
## Axis lable
plt.xlabel("False Positve Rate")
plt.ylabel("True Positive Rate")
## Show Legend
plt.legend()
```

Out[31]: <matplotlib.legend.Legend at 0x1b868388208>



In [ ]:

In [ ]:

In [1]:
```
## WE CONSIDER THE MALE AND FEMALE GENDER SEPERATELY FOR THE ANALYSIS
```

In [2]:
```
## Modules required
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
# Code
BC = (pd.read_excel('cancer.xlsx'))
```

In [4]:
```
#Import 'train_test_split' from 'sklearn.model_selection'
from sklearn.model_selection import train_test_split
#Import numpy#
import numpy as np

## SPLITTING DATA INTO MALE AND FEMALE
BCM=BC[BC.Gender==1]
BCF=BC[BC.Gender==0]
```

In [5]:
```
#### THE MALE DATASET
my=BCM.PatStatus
mx=BCM.drop(['PatStatus','PAT_ID', 'Gender'], axis=1)
```

In [6]:
```
## THE FEMALE DATASET
fy=BCF.PatStatus
fx=BCF.drop(['PatStatus','PAT_ID', 'Gender'],axis=1)
```

In [7]:
```
#Split the Male data into train and test sets #
mx_train, mx_test, my_train, my_test=train_test_split(mx,my, test_size=0.2, randc
```

In [8]:
```
## CONSIDERING THE MALE DATA
mx_train.head()
```

Out[8]:

	Race	MarST	AgeDiag	Grade	Stability	No.Visits	Lstay	Laterality	FamHist	PrioBSurgy
41599	1	1	65	3	0	16	9	2	1	0
25664	1	1	61	3	1	16	9	5	1	0
44540	6	0	74	2	0	10	9	8	1	0
46913	1	0	90	3	0	10	12	3	1	0
31494	1	1	71	3	0	10	9	4	1	0

5 rows × 23 columns

In [9]:
```
mx_test.head()
```

Out[9]:

	Race	MarST	AgeDiag	Grade	Stability	No.Visits	Lstay	Laterality	FamHist	PrioBSurgy
3055	1	0	62	1	0	4	5	5	1	0
30043	1	0	76	3	0	10	9	4	1	0
27137	1	0	85	2	0	10	9	4	1	0
22922	1	1	78	2	0	10	9	8	1	0
44064	1	1	45	2	0	14	9	2	1	0

5 rows × 23 columns

In [10]:
```
mx_train.shape
```

Out[10]:
```
(12479, 23)
```

In [11]:
```
mx_test.shape
```

Out[11]:
```
(3120, 23)
```

In [12]:
```
## Scaling the data
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler = preprocessing.MinMaxScaler()
mx_train_minmax = min_max_scaler.fit_transform(mx_train)
mx_test_minmax = min_max_scaler.fit_transform(mx_test)
```

In [13]:
```
 ## FITTING LOGISTIC REGRESSION FOR MALE DATA
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pylab
import math

from scipy.stats import norm
from scipy import stats
import statsmodels.api as sm
from sklearn import datasets, linear_model
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
%matplotlib inline

from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_
from sklearn.model_selection import cross_val_score, cross_validate
```

In [14]:
```
mlog_reg = sm.Logit(my_train, mx_train)
mlog_reg = mlog_reg.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.410336
         Iterations 8
```

In [15]:
```
print(mlog_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              PatStatus   No. Observations:                12479
Model:                          Logit   Df Residuals:                    12456
Method:                           MLE   Df Model:                           22
Date:                Thu, 16 Jul 2020   Pseudo R-squ.:                 0.3622
Time:                        10:37:40   Log-Likelihood:                -5120.6
converged:                       True   LL-Null:                       -8029.1
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Race           -0.0008      0.002     -0.353      0.724      -0.005       0.004
MarST          -0.0004      0.050     -0.009      0.993      -0.099       0.098
AgeDiag         0.0026      0.002      1.357      0.175      -0.001       0.006
Grade           0.1192      0.035      3.425      0.001       0.051       0.187
Stability       0.0759      0.086      0.883      0.377      -0.093       0.245
No.Visits      -0.0218      0.010     -2.180      0.029      -0.041      -0.002
Lstay          -0.1344      0.012    -10.926      0.000      -0.158      -0.110
Laterality     -0.0121      0.009     -1.288      0.198      -0.030       0.006
FamHist        -2.6038      0.405     -6.423      0.000      -3.398      -1.809
PrioBSurgy     -0.3056      0.131     -2.325      0.020      -0.563      -0.048
Suture          0.1467      0.062      2.373      0.018       0.026       0.268
 Density        0.2654      0.063      4.196      0.000       0.141       0.389
NipRet          0.8825      0.147      6.009      0.000       0.595       1.170
LyNode          2.6183      0.231     11.352      0.000       2.166       3.070
Amorph          1.4480      0.173      8.354      0.000       1.108       1.788
 Size           0.5909      0.068      8.684      0.000       0.458       0.724
Eggshell        0.1048      0.067      1.561      0.118      -0.027       0.236
Milk           -0.8173      0.180     -4.553      0.000      -1.169      -0.465
AxiAden         0.0887      0.063      1.400      0.161      -0.035       0.213
Distroph       -0.1351      0.162     -0.835      0.404      -0.452       0.182
Lucent          0.9937      0.144      6.921      0.000       0.712       1.275
Dermal          0.1767      0.063      2.796      0.005       0.053       0.300
SkinnLesson    -2.7320      0.068    -40.352      0.000      -2.865      -2.599
==============================================================================
```

In [16]:
```
## Exclude insignificant variables in the model
my=BCM.PatStatus
mx=BCM.drop(['PatStatus','PAT_ID', 'Race', 'MarST','AgeDiag', 'Stability', 'Later
```

In [17]:
```
#Split the data into train and test sets #
mx_train, mx_test, my_train, my_test=train_test_split(mx,my, test_size=0.2, rando

## Scaling the data
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler = preprocessing.MinMaxScaler()
mx_train_minmax = min_max_scaler.fit_transform(mx_train)
mx_test_minmax = min_max_scaler.fit_transform(mx_test)
```

In [18]:
```
## The new fitted logistic regression model with selected variables
import statsmodels.api as sm
mlog_reg = sm.Logit(my_train, mx_train)
mlog_reg = mlog_reg.fit()

mlog_reg1 = LogisticRegression()
mlog_reg1.fit(mx_train_minmax, my_train)
```

```
Optimization terminated successfully.
         Current function value: 0.410184
         Iterations 8

C:\Users\eagye\anaconda1\lib\site-packages\sklearn\linear_model\_logistic.py:94
0: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[18]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [19]:
```
print(mlog_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              PatStatus   No. Observations:                12479
Model:                          Logit   Df Residuals:                    12463
Method:                           MLE   Df Model:                           15
Date:                Thu, 16 Jul 2020   Pseudo R-squ.:                 0.3625
Time:                        11:02:20   Log-Likelihood:                -5118.7
converged:                       True   LL-Null:                       -8029.1
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Gender          3.2186      1.137      2.832      0.005       0.991       5.446
Grade           0.1079      0.035      3.086      0.002       0.039       0.176
No.Visits      -0.0300      0.009     -3.302      0.001      -0.048      -0.012
Lstay          -0.1381      0.012    -11.135      0.000      -0.162      -0.114
FamHist        -4.7898      1.066     -4.493      0.000      -6.879      -2.700
PrioBSurgy     -0.3844      0.133     -2.888      0.004      -0.645      -0.124
Suture          0.1390      0.062      2.251      0.024       0.018       0.260
 Density        0.2094      0.065      3.204      0.001       0.081       0.338
NipRet          0.8124      0.149      5.441      0.000       0.520       1.105
LyNode          2.6812      0.220     12.215      0.000       2.251       3.111
Amorph          1.3794      0.173      7.975      0.000       1.040       1.718
 Size           0.5364      0.069      7.803      0.000       0.402       0.671
Milk           -0.9406      0.185     -5.089      0.000      -1.303      -0.578
Lucent          0.7765      0.159      4.895      0.000       0.466       1.087
Dermal          0.1519      0.059      2.560      0.010       0.036       0.268
SkinnLesson    -2.7614      0.069    -40.273      0.000      -2.896      -2.627
==============================================================================
```

In [20]:
```
## Score of the model giving the accuracy of the model
print("Accuracy", (mlog_reg1.score(mx_test_minmax, my_test)))
```

```
Accuracy 0.8233974358974359
```

In [21]:
```
### PREDICTION ON THE TEST DATASET
```

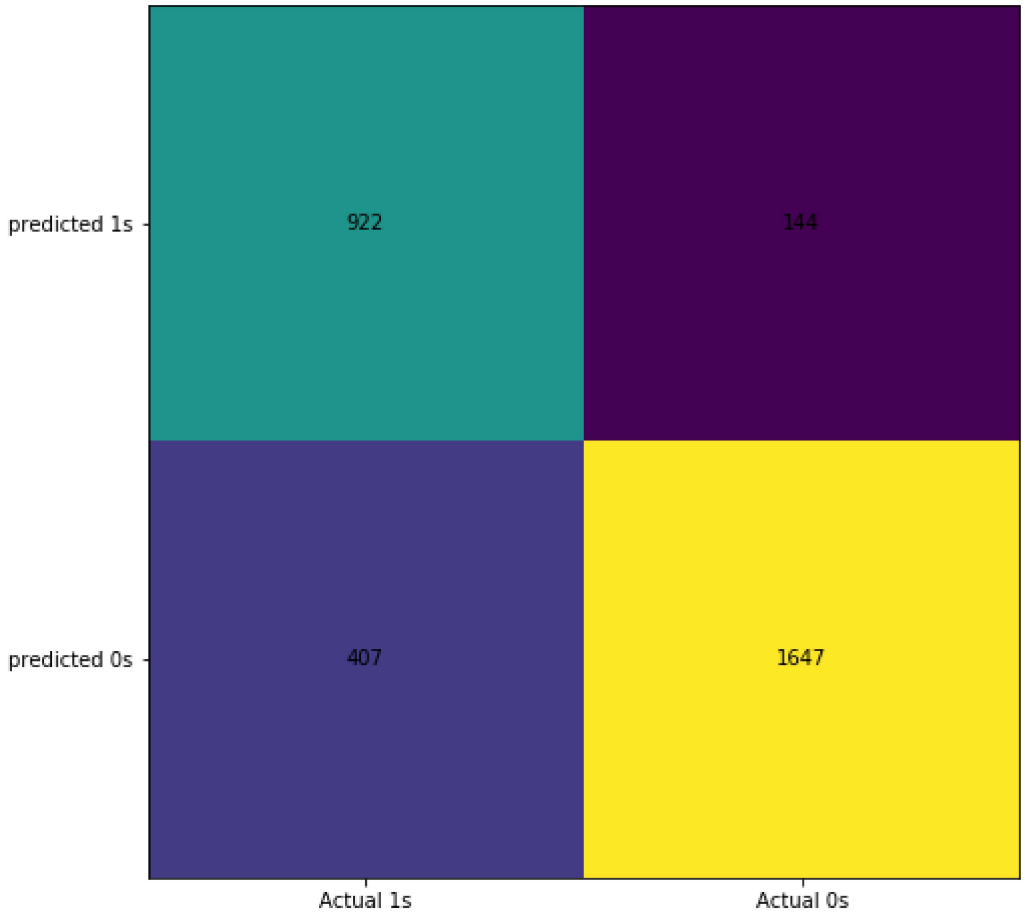
In [22]:
```
### Getting the prediction for the Testing dataset
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_
from sklearn.model_selection import cross_val_score, cross_validate

mx_train = mx_train_minmax
mx_test = mx_test_minmax
```

In [23]:
```
tns_probs=[0 for _ in range(len(my_test))]
my_predict = mlog_reg1.predict(mx_test)
```

In [24]:
```
## Keeping the probabilities for Testing outcomes
my_pred = mlog_reg1.predict_proba(mx_test)
my_pred = my_pred[:, 1]
```

In [25]:
```
 ## confusion matrix
mtest_cm = confusion_matrix(my_test, np.round(my_predict))
fig, ax = plt.subplots(figsize = (8, 8))
ax.imshow(mtest_cm)
ax.grid(False)
ax.xaxis.set(ticks=(0,1), ticklabels=('Actual 1s', 'Actual 0s'))
ax.yaxis.set(ticks=(0,1), ticklabels=('predicted 1s', 'predicted 0s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, mtest_cm[i, j], ha= 'center', va= 'center', color= 'black')
plt.show()
```



In [26]:
```
 ## Error for the prediction for test dataset outcomes
mtest_error = (mtest_cm[0,1] + mtest_cm[1,0])/np.sum(mtest_cm)
print(mtest_error)
```

```
0.1766025641025641
```

In [27]:
```
## Accuracy of prediction
1-mtest_error
```

Out[27]:
```
0.8233974358974359
```

In [28]:
```
## Sensitivity Analysis
mtest_sens = mtest_cm[1, 1]/(mtest_cm[1, 1] + mtest_cm[0, 1])
print(mtest_sens)
```

```
0.9195979899497487
```

In [29]:
```
 ## Specificity Analysis
mtest_spec = mtest_cm[0, 0]/(mtest_cm[0, 0] + mtest_cm[1, 0])
print(mtest_spec)
```

```
0.6937547027840482
```

In [30]:
```
 ## PPV Analysis
mtest_npv = mtest_cm[1, 1]/(mtest_cm[1, 1] + mtest_cm[1, 0])
print(mtest_npv)
```

```
0.8018500486854917
```

In [31]:
```
## NPV Analysis
mtest_npv = mtest_cm[0, 0]/(mtest_cm[0, 0] + mtest_cm[0, 1])
print(mtest_npv)
```

```
0.8649155722326454
```

In [32]:
```
## The AUC Score
mtest_auc = roc_auc_score(my_test, tns_probs)
my_pred_auc = np.round(roc_auc_score(my_test, my_pred), decimals = 2)
```

In [33]:
```
 print(mtest_auc)
```

```
0.5
```

In [34]:
```
print(np.round(my_pred_auc, decimals = 2))
```

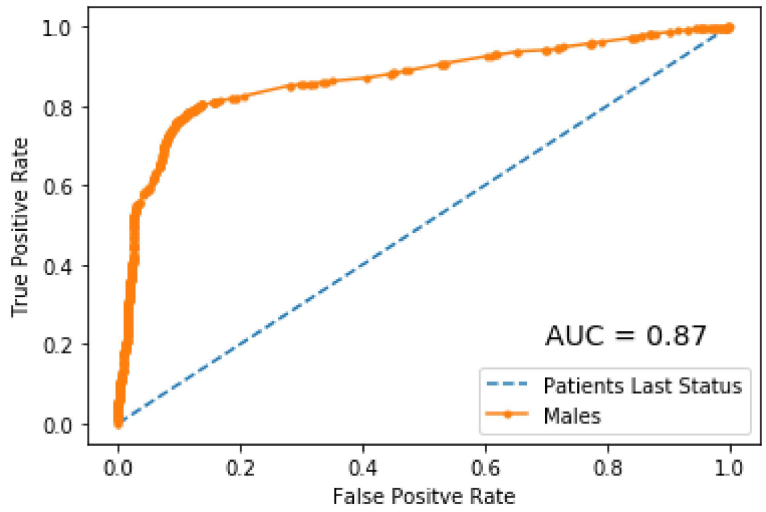
```
0.87
```

In [35]:
```
## calculate ROC Curves
mtest_fpr, mtest_tpr, _ = roc_curve(my_test, tns_probs)
my_pred_fpr, my_pred_tpr, _ = roc_curve(my_test, my_pred)
```

In [37]:
```
## Plot Curve for the model
import numpy as np
import matplotlib.pyplot as plt

plt.plot(mtest_fpr, mtest_tpr, linestyle = '--', label = 'Patients Last Status')
plt.plot(my_pred_fpr, my_pred_tpr, marker = '.', label = 'Males')
plt.text(0.7, 0.2, "AUC = " + str(my_pred_auc), fontsize = 14)
## Axis lable
plt.xlabel("False Positve Rate")
plt.ylabel("True Positive Rate")
## Show Legend
plt.legend()
```

Out[37]: <matplotlib.legend.Legend at 0x188f82b6e48>



In [ ]:

In [ ]:

In [ ]:

In [1]:
```
## CONSIDERING THE FEMALE DATA
## The new fitted logistic regression model with selected variables
## Modules required
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
BC = (pd.read_excel('cancer.xlsx'))
BCF=BC[BC.Gender==0]

fy=BCF.PatStatus
fx=BCF.drop(['PatStatus','PAT_ID', 'Gender'],axis=1)
```

In [2]:
```
#Import 'train_test_split' from 'sklearn.model_selection'
from sklearn.model_selection import train_test_split

#Import numpy#
import numpy as np
#Split the Male data into train and test sets #
fx_train, fx_test, fy_train, fy_test=train_test_split(fx,fy, test_size=0.2, randc
```

In [3]:
```
# Scaling the data
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler = preprocessing.MinMaxScaler()
fx_train_minmax = min_max_scaler.fit_transform(fx_train)
fx_test_minmax = min_max_scaler.fit_transform(fx_test)
```

In [4]:
```
 ## FITTING LOGISTIC REGRESSION FOR FEMALE DATA
from scipy.stats import norm
from scipy import stats
import statsmodels.api as sm
from sklearn import datasets, linear_model
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
%matplotlib inline

from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_
from sklearn.model_selection import cross_val_score, cross_validate
```

In [5]:
```
flog_reg = sm.Logit(fy_train, fx_train)
flog_reg = flog_reg.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.433132
         Iterations 7
```

In [6]:
```
print(flog_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              PatStatus   No. Observations:                67522
Model:                          Logit   Df Residuals:                    67499
Method:                           MLE   Df Model:                           22
Date:                Thu, 16 Jul 2020   Pseudo R-squ.:                 0.3749
Time:                        12:56:46   Log-Likelihood:                -29246.
converged:                       True   LL-Null:                       -46789.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Race           0.0059      0.001      9.034      0.000       0.005       0.007
MarST          0.0730      0.021      3.523      0.000       0.032       0.114
AgeDiag        0.0078      0.001     11.084      0.000       0.006       0.009
Grade         -0.0393      0.014     -2.863      0.004      -0.066      -0.012
Stability      0.1699      0.039      4.314      0.000       0.093       0.247
No.Visits      0.0507      0.003     17.406      0.000       0.045       0.056
Lstay         -0.1656      0.004    -45.392      0.000      -0.173      -0.158
Laterality    -0.0942      0.004    -23.309      0.000      -0.102      -0.086
FamHist       -3.2622      0.064    -50.609      0.000      -3.388      -3.136
PrioBSurgy    -0.2662      0.026    -10.256      0.000      -0.317      -0.215
Suture        -0.0323      0.024     -1.321      0.187      -0.080       0.016
 Density       0.1879      0.015     12.304      0.000       0.158       0.218
NipRet         2.2790      0.035     64.601      0.000       2.210       2.348
LyNode         0.1978      0.031      6.426      0.000       0.137       0.258
Amorph        -0.1152      0.033     -3.479      0.001      -0.180      -0.050
 Size         -0.2425      0.016    -14.893      0.000      -0.274      -0.211
Eggshell      -0.5637      0.027    -20.885      0.000      -0.617      -0.511
Milk           0.9166      0.026     35.257      0.000       0.866       0.968
AxiAden        1.2250      0.023     52.419      0.000       1.179       1.271
Distroph      -0.6857      0.040    -17.217      0.000      -0.764      -0.608
Lucent         2.0236      0.040     51.090      0.000       1.946       2.101
Dermal        -0.0202      0.026     -0.775      0.438      -0.071       0.031
SkinnLesson   -1.4800      0.022    -66.078      0.000      -1.524      -1.436
==============================================================================
```

In [7]:
```
### The new fitted logistic regression model with selected variables
## Modules required
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
BC = (pd.read_excel('cancer.xlsx'))
BCF=BC[BC.Gender==0]

fy=BCF.PatStatus
fx=BCF.drop(['PatStatus','PAT_ID', 'Gender','Suture','Dermal'],axis=1)
```

In [8]:
```
#Import 'train_test_split' from 'sklearn.model_selection'
from sklearn.model_selection import train_test_split

#Import numpy#
import numpy as np
#Split the Male data into train and test sets #
fx_train, fx_test, fy_train, fy_test=train_test_split(fx,fy, test_size=0.2, randc
```

In [9]:
```
fx_train.head()
```

Out[9]:

	Race	MarST	AgeDiag	Grade	Stability	No.Visits	Lstay	Laterality	FamHist	PrioBSurgy
88023	1	0	42	3	0	9	15	4	1	1
92578	1	0	66	1	0	8	8	3	1	1
75760	1	1	61	3	0	9	9	5	1	1
51727	1	1	50	1	0	15	10	4	1	1
3377	1	1	78	2	1	5	2	8	1	1

5 rows × 21 columns

In [10]:
```
fx_test.head()
```

Out[10]:

	Race	MarST	AgeDiag	Grade	Stability	No.Visits	Lstay	Laterality	FamHist	PrioBSurgy
41774	1	1	67	2	0	16	9	2	1	0
84188	16	1	41	3	0	8	8	8	1	1
55207	1	1	50	1	0	6	10	9	1	0
51157	1	1	78	3	0	1	10	9	1	1
99358	1	1	77	2	0	10	12	3	1	0

5 rows × 21 columns

In [11]:
```
fx_train.shape
```

Out[11]:
```
(67522, 21)
```

In [12]:
```
fx_test.shape
```

Out[12]:
```
(16881, 21)
```

In [13]:
```
# Scaling the data
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler = preprocessing.MinMaxScaler()
fx_train_minmax = min_max_scaler.fit_transform(fx_train)
fx_test_minmax = min_max_scaler.fit_transform(fx_test)
```

In [14]:
```
 ## FITTING LOGISTIC REGRESSION FOR FEMALE DATA
from scipy.stats import norm
from scipy import stats
import statsmodels.api as sm
from sklearn import datasets, linear_model
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
%matplotlib inline

from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_
from sklearn.model_selection import cross_val_score, cross_validate
```

In [15]:
```
flog_reg = sm.Logit(fy_train, fx_train)
flog_reg = flog_reg.fit()

flog_reg1 = LogisticRegression()
flog_reg1.fit(fx_train, fy_train)
```

```
Optimization terminated successfully.
         Current function value: 0.433149
         Iterations 7

C:\Users\eagye\anaconda1\lib\site-packages\sklearn\linear_model\_logistic.py:94
0: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[15]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [16]:
```
print(flog_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              PatStatus   No. Observations:                67522
Model:                          Logit   Df Residuals:                    67501
Method:                           MLE   Df Model:                           20
Date:                Thu, 16 Jul 2020   Pseudo R-squ.:                 0.3749
Time:                        12:59:29   Log-Likelihood:                -29247.
converged:                       True   LL-Null:                       -46789.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Race           0.0059      0.001      9.026      0.000       0.005       0.007
MarST          0.0728      0.021      3.513      0.000       0.032       0.113
AgeDiag        0.0078      0.001     11.070      0.000       0.006       0.009
Grade         -0.0393      0.014     -2.862      0.004      -0.066      -0.012
Stability      0.1703      0.039      4.325      0.000       0.093       0.247
No.Visits      0.0504      0.003     17.358      0.000       0.045       0.056
Lstay         -0.1658      0.004    -45.434      0.000      -0.173      -0.159
Laterality    -0.0941      0.004    -23.298      0.000      -0.102      -0.086
FamHist       -3.2673      0.064    -50.755      0.000      -3.393      -3.141
PrioBSurgy    -0.2673      0.026    -10.311      0.000      -0.318      -0.216
 Density       0.1860      0.015     12.236      0.000       0.156       0.216
NipRet         2.2776      0.035     64.651      0.000       2.209       2.347
LyNode         0.1914      0.030      6.281      0.000       0.132       0.251
Amorph        -0.1097      0.033     -3.348      0.001      -0.174      -0.045
 Size         -0.2427      0.016    -14.960      0.000      -0.275      -0.211
Eggshell      -0.5663      0.027    -21.028      0.000      -0.619      -0.514
Milk           0.9140      0.026     35.258      0.000       0.863       0.965
AxiAden        1.2251      0.023     52.431      0.000       1.179       1.271
Distroph      -0.6975      0.038    -18.382      0.000      -0.772      -0.623
Lucent         2.0218      0.039     51.615      0.000       1.945       2.099
SkinnLesson   -1.4795      0.022    -66.083      0.000      -1.523      -1.436
==============================================================================
```

In [17]:
```
## Score of the model giving the accuracy of the model
print("Accuracy", (flog_reg1.score(fx_test_minmax, fy_test)))
```

```
Accuracy 0.7837213435223032
```

In [18]:
```
### PREDICTION ON THE TEST DATASET
```

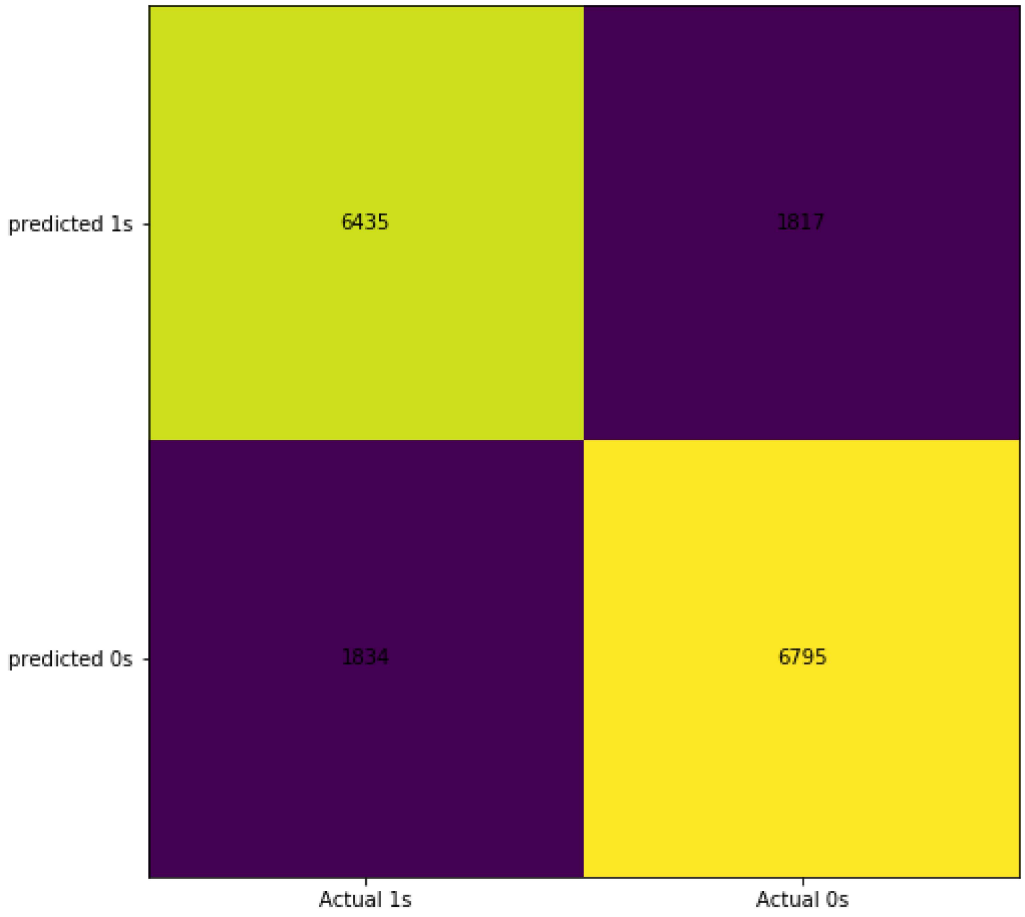
In [20]:
```
### Getting the prediction for the Testing dataset
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_
from sklearn.model_selection import cross_val_score, cross_validate

fx_train = fx_train_minmax
fx_test = fx_test_minmax
```

In [24]:
```
tns_probs=[0 for _ in range(len(fy_test))]
fy_predict = flog_reg1.predict(fx_test)
```

In [25]:
```
## Keeping the probabilities for Testing outcomes
fy_pred = flog_reg1.predict_proba(fx_test)
fy_pred = fy_pred[:, 1]
```

In [26]:
```
 ## confusion matrix
ftest_cm = confusion_matrix(fy_test, np.round(fy_predict))
fig, ax = plt.subplots(figsize = (8, 8))
ax.imshow(ftest_cm)
ax.grid(False)
ax.xaxis.set(ticks=(0,1), ticklabels=('Actual 1s', 'Actual 0s'))
ax.yaxis.set(ticks=(0,1), ticklabels=('predicted 1s', 'predicted 0s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, ftest_cm[i, j], ha= 'center', va= 'center', color= 'black')
plt.show()
```



In [27]:
```
## Error for the prediction for test dataset outcomes
ftest_error = (ftest_cm[0,1] + ftest_cm[1,0])/np.sum(ftest_cm)
print(ftest_error)
```

```
0.2162786564776968
```

In [28]:
```
## Accuracy of prediction
1-ftest_error
```

Out[28]:
```
0.7837213435223032
```

In [29]:
```
## Sensitivity Analysis
ftest_sens = ftest_cm[1, 1]/(ftest_cm[1, 1] + ftest_cm[0, 1])
print(ftest_sens)
```

```
0.7890153274500696
```

In [30]:
```
## Specificity Analysis
ftest_spec = ftest_cm[0, 0]/(ftest_cm[0, 0] + ftest_cm[1, 0])
print(ftest_spec)
```

```
0.7782077639375983
```

In [31]:
```
## PPV Analysis
ftest_npv = ftest_cm[1, 1]/(ftest_cm[1, 1] + ftest_cm[1, 0])
print(ftest_npv)
```

```
0.7874608877042532
```

In [32]:
```
## NPV Analysis
ftest_npv = ftest_cm[0, 0]/(ftest_cm[0, 0] + ftest_cm[0, 1])
print(ftest_npv)
```

```
0.7798109549200194
```

In [33]:
```
## The AUC Score
ftest_auc = roc_auc_score(fy_test, tns_probs)
fy_pred_auc = np.round(roc_auc_score(fy_test, fy_pred), decimals = 2)
```

In [34]:
```
 print(ftest_auc)
```

```
0.5
```

In [35]:
```
print(np.round(fy_pred_auc, decimals = 2))
```

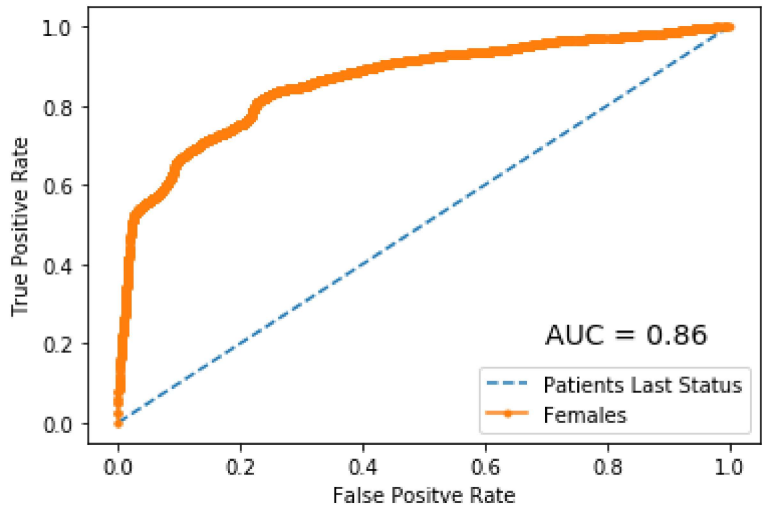
```
0.86
```

In [36]:
```
## calculate ROC Curves
ftest_fpr, ftest_tpr, _ = roc_curve(fy_test, tns_probs)
fy_pred_fpr, fy_pred_tpr, _ = roc_curve(fy_test, fy_pred)
```

In [37]:
```
## Plot Curve for the model
import numpy as np
import matplotlib.pyplot as plt

plt.plot(ftest_fpr, ftest_tpr, linestyle = '--', label = 'Patients Last Status')
plt.plot(fy_pred_fpr, fy_pred_tpr, marker = '.', label = 'Females')
plt.text(0.7, 0.2, "AUC = " + str(fy_pred_auc), fontsize = 14)
## Axis lable
plt.xlabel("False Positve Rate")
plt.ylabel("True Positive Rate")
## Show Legend
plt.legend()
```

Out[37]: <matplotlib.legend.Legend at 0x1eb06c904c8>



In [ ]:

In [ ]: