In [1]:
```
### BREAST CANCER CASES ###
###### NEURAL NETWORK CODE IN JUPYTER NOTEBOOK #####
```

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]:
```
BC.head()
```

Out[4]:

	PatStatus	Race	MarST	Gender	AgeDiag	Grade	Stability	No.Visits	Lstay	Laterality	...	LyNode	Amorp
0	1	3	1	0	52	3	0	5	1	4	...	1	
1	1	3	1	0	48	3	0	4	3	5	...	1	
2	0	3	0	0	69	2	0	7	9	8	...	1	
3	1	3	0	0	47	2	0	15	9	9	...	1	
4	1	3	0	0	66	3	0	9	5	4	...	1	

5 rows × 25 columns

In [5]:
```
#Import 'train_test_split' from 'sklearn.model_selection'
from sklearn.model_selection import train_test_split

#Import numpy#
import numpy as np
```

In [6]:
```
y = BC.PatStatus
x = BC.drop(['PatStatus'], axis = 1)
```

In [7]:
```
#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_state=
123)


## 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 [8]:
```
x_train = x_train_minmax
x_test = x_test_minmax
```

In [9]:
```
x_train.shape
```

Out[9]:
```
(80001, 24)
```

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

Out[10]:
```
(20001, 24)
```

In [11]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_cu
rve
from sklearn.model_selection import cross_val_score, cross_validate
```

In [12]:
```
##Fitting the neural network model using training dataset
tns_probs=[0 for _ in range(len(y_test))]
```

In [20]:
```
tmlp=MLPClassifier(hidden_layer_sizes=(6, 6, 6, 6), activation ='relu', solver = 'a
dam' ,alpha= 0.01, batch_size='auto', learning_rate = 'adaptive', max_iter = 10000,
learning_rate_init=0.001, power_t=0.5)
tmlp.fit(x_train, y_train)
```

Out[20]:
```
MLPClassifier(activation='relu', alpha=0.01, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(6, 6, 6, 6), learning_rate='adaptive',
              learning_rate_init=0.001, max_fun=15000, max_iter=10000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

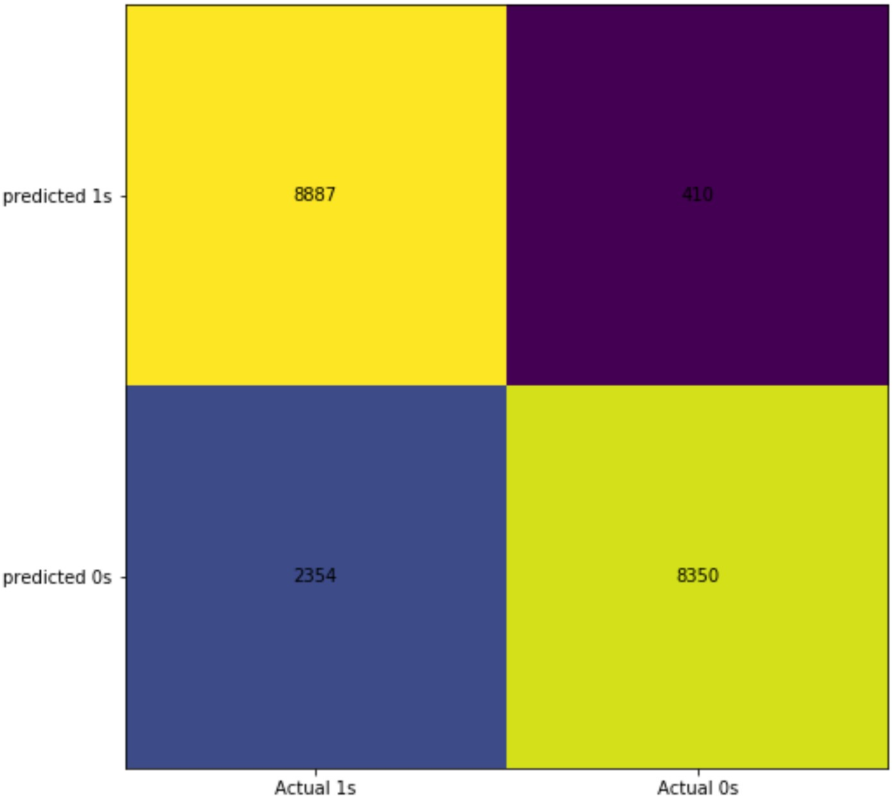
In [21]:
```
### PREDICTION ON THE TEST DATASET
```

In [22]:
```
### Getting the prediction for the Testing dataset
y_predict = tmlp.predict(x_test)
```

In [23]:
```
## Keeping the probabilities for Testing outcomes
y_pred = tmlp.predict_proba(x_test)
y_pred = y_pred[:,1]
```

In [24]:

```
## CONFUSION MATRIX FOR BOTH SEX DATA
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 [25]:

```
## 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)
```

```
0.13819309034548272
```

In [26]:

```
## Accuracy of prediction
1-test_error
```

Out[26]:

```
0.8618069096545173
```

In [27]:

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

```
0.9531963470319634
```

In [28]:

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

```
0.7905880259763366
```

In [29]:

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

```
0.7800822122571002
```

In [30]:

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

```
0.9558997526083682
```

In [31]:

```
## 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 [32]:

```
print(test_auc)
```

```
0.5
```

In [33]:

```
print(np.round(y_pred_auc, decimals = 2))
```

```
0.96
```

In [34]:

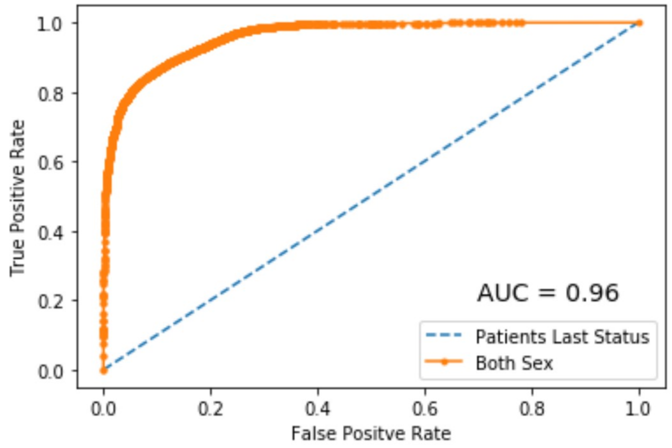
```
## calculate ROC Curves
test_fpr, test_tpr, _ = roc_curve(y_test, tns_probs)
y_pred_fpr, y_pred_tpr, _ = roc_curve(y_test, y_pred)
```

In [35]:

```
## 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[35]:

```
<matplotlib.legend.Legend at 0x208a93a54c8>
```



In [ ]:

In [ ]:

In [ ]:

In [1]:

```
## CONSIDER THE NEURAL NETWORK FOR EACH GENDER SEPARATELY
## Modules required
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
# Code
MBC = (pd.read_excel('MBC.xlsx'))
```

In [3]:

```
#Import 'train_test_split' from 'sklearn.model_selection'
from sklearn.model_selection import train_test_split
#Import numpy#
import numpy as np
```

In [4]:

```
#### THE MALE DATASET
my=MBC.PatStatus
mx=MBC.drop(['PatStatus','Gender'], axis=1)
```

In [5]:

```
## CONSIDER FITTING NEURAL NETWORK FOR THE MALE GENDER
```

```
In [6]: #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, random_
        state=124)
```

```
In [7]: mx_train.head()
```

Out[7]:

	Race	MarST	AgeDiag	Grade	Stability	No.Visits	Lstay	Laterality	FamHist	PrioBSurgy	...	LyNode
12630	1	1	77	3	0	10	9	5	1	0	...	1
6329	1	0	54	2	0	15	1	3	1	0	...	1
14089	10	0	63	3	0	16	9	8	1	0	...	1
15211	1	1	61	1	0	16	12	4	1	0	...	1
8895	1	1	69	3	0	16	9	9	1	0	...	1

5 rows × 23 columns

```
In [8]: mx_test.head()
```

Out[8]:

	Race	MarST	AgeDiag	Grade	Stability	No.Visits	Lstay	Laterality	FamHist	PrioBSurgy	...	LyNode
1541	1	1	70	2	0	10	9	8	1	0	...	1
8170	1	0	59	2	0	15	1	1	1	0	...	1
6776	1	0	46	3	0	14	9	3	1	0	...	1
4929	2	0	48	1	0	14	9	9	1	0	...	1
13849	1	0	58	1	0	15	9	8	1	0	...	1

5 rows × 23 columns

```
In [9]: mx_train.shape

Out[9]: (12479, 23)
```

```
In [10]: mx_test.shape

Out[10]: (3120, 23)
```

```
In [11]: ## Scaling the male data set
         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 [12]: mx_train = mx_train_minmax
         mx_test = mx_test_minmax
```

```
In [13]: ## FITTING NEURAL NETWORK FOR MALE DATA
```

```
In [14]: from sklearn.neural_network import MLPClassifier
         from sklearn.datasets import make_classification
         from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_cu
         rve
         from sklearn.model_selection import cross_val_score, cross_validate
```

In [15]:

```
##Fitting the neural network model using training dataset
tns_probs=[0 for _ in range(len(my_test))]
```

In [16]:

```
male_mlp=MLPClassifier(hidden_layer_sizes=(6, 6, 6, 6), activation ='relu', solver
= 'adam' ,alpha= 0.01, batch_size='auto', learning_rate = 'adaptive', max_iter = 10
000, learning_rate_init=0.001, power_t=0.5)
male_mlp.fit(mx_train, my_train)
```

Out[16]:

```
MLPClassifier(activation='relu', alpha=0.01, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(6, 6, 6, 6), learning_rate='adaptive',
              learning_rate_init=0.001, max_fun=15000, max_iter=10000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

In [17]:

```
### PREDICTION USING THE TEST DATASET
```

In [18]:

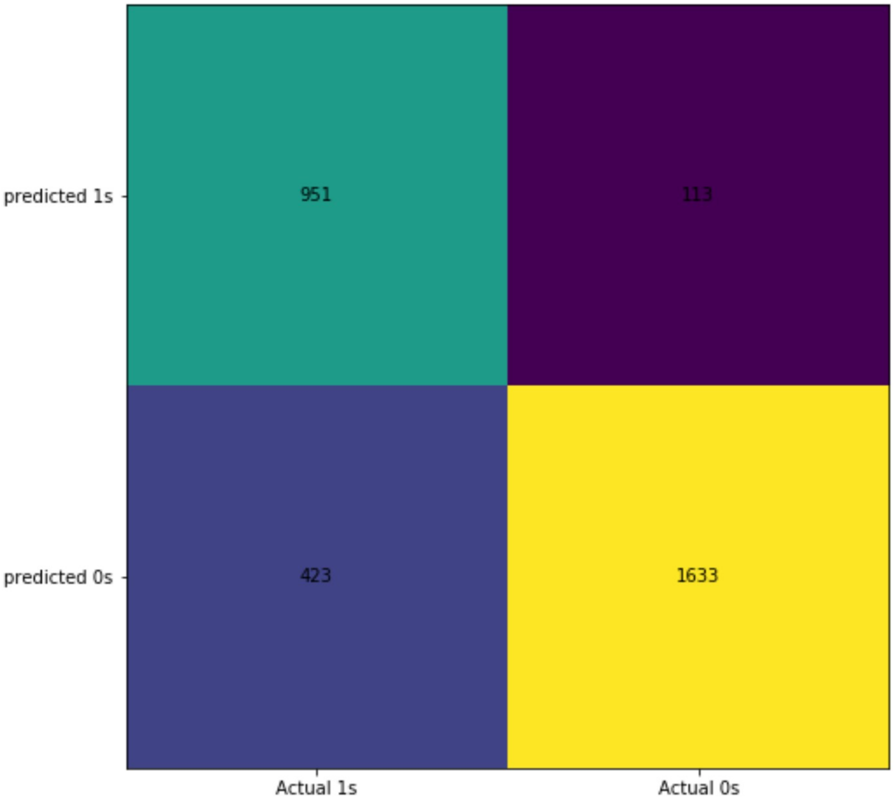
```
### Getting the prediction for the Testing dataset
my_predict = male_mlp.predict(mx_test)
```

In [19]:

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

In [20]:

```
## CONFUSION MATRIX FOR MALE DATA
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 [21]:

```
## 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)
```

```
0.1717948717948718
```

In [22]:

```
## Accuracy of prediction
1-mtest_error
```

Out[22]:

```
0.8282051282051281
```

In [23]:

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

```
0.9352806414662085
```

In [24]:

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

```
0.6921397379912664
```

In [25]:

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

```
0.794260700389105
```

In [26]:

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

```
0.893796992481203
```

In [27]:

```
## The AUC Score
tns_probs=[0 for _ in range(len(my_test))]
mtest_auc = roc_auc_score(my_test, tns_probs)
my_pred_auc = np.round(roc_auc_score(my_test, my_pred), decimals = 2)
```

In [28]:

```
print(mtest_auc)
```

```
0.5
```

In [29]:

```
print(np.round(my_pred_auc, decimals = 2))
```

```
0.88
```

In [30]:

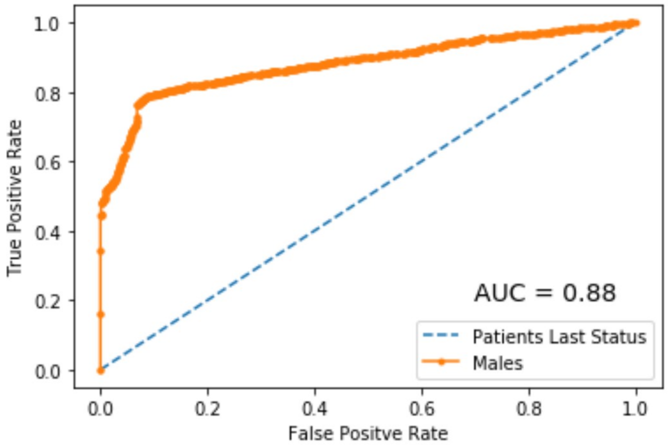
```
## calculate ROC Curves
mtest_fpr, mtest_tpr, _ = roc_curve(my_test, tns_probs)
my_pred_fpr, my_pred_tpr, _ = roc_curve(my_test, my_pred)
```

In [31]:

```
## 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[31]:

```
<matplotlib.legend.Legend at 0x187a1ba82c8>
```



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
FBC = (pd.read_excel('FBC.xlsx'))
```

In [3]:

```
# splitting data into x and y
fy=FBC.PatStatus
fx=FBC.drop(['PatStatus','Gender'], axis=1)
```

In [4]:

```
#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, random_
state=125)
```

In [5]:
```
# Scaling the female 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 [6]:
```
fx_train = fx_train_minmax
fx_test = fx_test_minmax
```

In [7]:
```
### FITTING THE NEURAL NETWORK USING THE FEMALE TRAINING DATASET
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, plot_roc_cu
rve
from sklearn.model_selection import cross_val_score, cross_validate

tns_probs=[0 for _ in range(len(fy_test))]
```

In [8]:
```
female_mlp=MLPClassifier(hidden_layer_sizes=(6, 6, 6, 6), activation ='relu', solve
r = 'adam' ,alpha= 0.01, batch_size='auto', learning_rate = 'adaptive', max_iter =
10000, learning_rate_init=0.001, power_t=0.5)
female_mlp.fit(fx_train, fy_train)
```

Out[8]:
```
MLPClassifier(activation='relu', alpha=0.01, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(6, 6, 6, 6), learning_rate='adaptive',
              learning_rate_init=0.001, max_fun=15000, max_iter=10000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

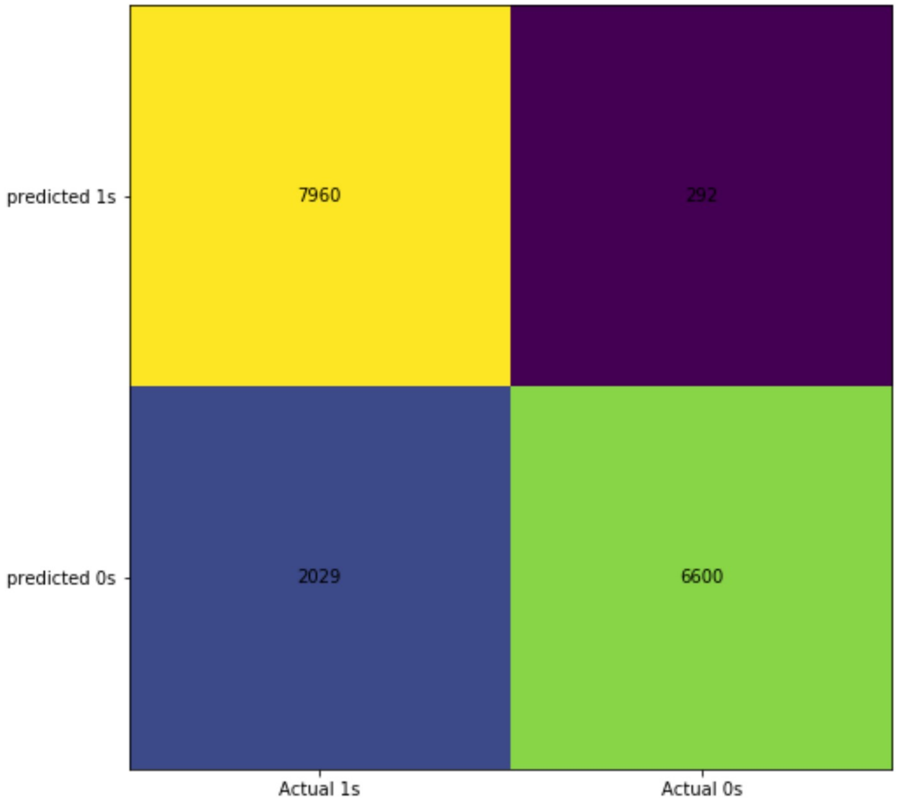
In [9]:
```
## PREDICTION USING THE TEST DATASET
```

In [10]:
```
### Getting the prediction for the Testing dataset
fy_predict = female_mlp.predict(fx_test)
```

In [11]:
```
## Keeping the probabilities for Testing outcomes
fy_pred = female_mlp.predict_proba(fx_test)
fy_pred = fy_pred[:,1]
```

In [12]:

```
## confusion matrix for female gender
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 [13]:

```
## 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)
```

```
0.13749185474794148
```

In [14]:

```
## Accuracy of prediction
1-ftest_error
```

Out[14]:

```
0.8625081452520585
```

In [15]:

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

```
0.9576320371445154
```

In [16]:

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

```
0.7968765642206427
```

In [17]:

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

```
0.764862672383822
```

In [18]:

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

```
0.9646146388754241
```

In [19]:

```
## 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 [20]:

```
print(ftest_auc)
```

```
0.5
```

In [21]:

```
print(np.round(fy_pred_auc, decimals = 2))
```

```
0.95
```

In [22]:

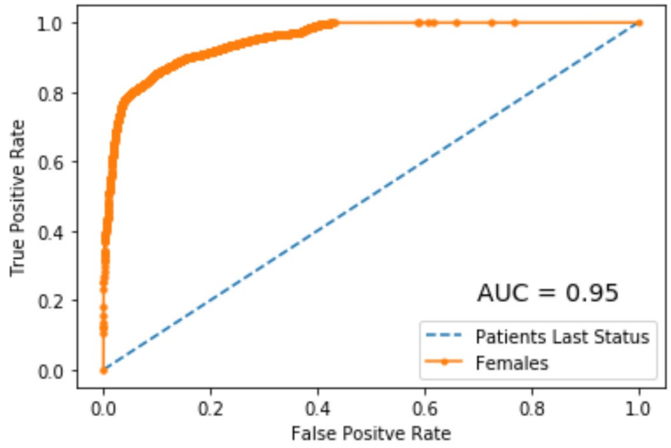
```
## calculate ROC Curves
ftest_fpr, ftest_tpr, _ = roc_curve(fy_test, tns_probs)
fy_pred_fpr, fy_pred_tpr, _ = roc_curve(fy_test, fy_pred)
```

In [23]:

```
## 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[23]:

```
<matplotlib.legend.Legend at 0x1b4b9c836c8>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: