**######LOGISTIC REGRESSION CODES######**

**# 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**

**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**

**BC = pd.read\_excel('cancer.xlsx')**

**BC.head()**

**## 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**

**y = BC.PatStatus**

**x = BC.drop(['PatStatus','PAT\_ID'], axis = 1)**

**#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)**

**log\_reg = sm.Logit(y\_train, x\_train)**

**log\_reg = log\_reg.fit()**

**## exclude insignificant variables in the model**

**y = BC.PatStatus**

**x = BC.drop(['PatStatus', 'PAT\_ID','Grade','Suture','Dermal'], axis = 1)**

**#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)**

**## 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)**

**## Score of the model giving the accuracy of the model**

**print("Accuracy", (log\_reg1.score(x\_test\_minmax, y\_test)))**

**### PREDICTION ON THE TEST DATASET**

**### 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\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**x\_train = x\_train\_minmax**

**x\_test = x\_test\_minmax**

**tns\_probs=[0 for \_ in range(len(y\_test))]**

**y\_predict = log\_reg1.predict(x\_test)**

**## Keeping the probabilities for Testing outcomes**

**y\_pred = log\_reg1.predict\_proba(x\_test)**

**y\_pred = y\_pred[:,1]**

**## 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()**

**## 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)**

**## Accuracy of prediction**

**1-test\_error**

**## Sensitivity Analysis**

**test\_sens = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[0, 1])**

**print(test\_sens)**

**## Specificity Analysis**

**test\_spec = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[1, 0])**

**print(test\_spec)**

**## PPV Analysis**

**test\_npv = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[1, 0])**

**print(test\_npv)**

**## NPV Analysis**

**test\_npv = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[0, 1])**

**print(test\_npv)**

**## 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)**

**print(test\_auc)**

**print(np.round(y\_pred\_auc, decimals = 2))**

**## 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**

**## 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()**

**## WE CONSIDER THE MALE AND FEMALE GENDER SEPERATELY FOR THE ANALYSIS**

**## Modules required**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**# Code**

**BC = (pd.read\_excel('cancer.xlsx'))**

**#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**

**#### THE MALE DATASET**

**my=BCM.PatStatus**

**mx=BCM.drop(['PatStatus','PAT\_ID', 'Gender'], axis=1)**

**## THE FEMALE DATASET**

**fy=BCF.PatStatus**

**fx=BCF.drop(['PatStatus','PAT\_ID', 'Gender'],axis=1)**

**#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)**

**## CONSIDERING THE MALE DATA**

**mx\_train.head()**

**## 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)**

**## 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\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**mlog\_reg = sm.Logit(my\_train, mx\_train)**

**mlog\_reg = mlog\_reg.fit()**

**## Exclude insignificant variables in the model**

**my=BCM.PatStatus**

**mx=BCM.drop(['PatStatus','PAT\_ID', 'Race', 'MarST','AgeDiag', 'Stability', 'Laterality', 'Eggshell', 'AxiAden', 'Distroph'], axis=1)**

**#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, random\_state=124)**

**## 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)**

**## 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)**

**## Score of the model giving the accuracy of the model**

**print("Accuracy", (mlog\_reg1.score(mx\_test\_minmax, my\_test)))**

**### PREDICTION ON THE TEST DATASET**

**### 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\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**mx\_train = mx\_train\_minmax**

**mx\_test = mx\_test\_minmax**

**tns\_probs=[0 for \_ in range(len(my\_test))]**

**my\_predict = mlog\_reg1.predict(mx\_test)**

**## Keeping the probabilities for Testing outcomes**

**my\_pred = mlog\_reg1.predict\_proba(mx\_test)**

**my\_pred = my\_pred[:, 1]**

**## 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()**

**## 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)**

**## Accuracy of prediction**

**1-mtest\_error**

**## Sensitivity Analysis**

**mtest\_sens = mtest\_cm[1, 1]/(mtest\_cm[1, 1] + mtest\_cm[0, 1])**

**print(mtest\_sens)**

**## Specificity Analysis**

**mtest\_spec = mtest\_cm[0, 0]/(mtest\_cm[0, 0] + mtest\_cm[1, 0])**

**print(mtest\_spec)**

**## PPV Analysis**

**mtest\_npv = mtest\_cm[1, 1]/(mtest\_cm[1, 1] + mtest\_cm[1, 0])**

**print(mtest\_npv)**

**## NPV Analysis**

**mtest\_npv = mtest\_cm[0, 0]/(mtest\_cm[0, 0] + mtest\_cm[0, 1])**

**print(mtest\_npv)**

**## 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)**

**print(mtest\_auc)**

**print(np.round(my\_pred\_auc, decimals = 2))**

**## 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)**

**## 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()**

**# 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**

**BCF= (pd.read\_excel('cancer.xlsx'))**

**BCF=BC[BC.Gender==0]**

**fy=BCF.PatStatus**

**fx=BCF.drop(['PatStatus','PAT\_ID', 'Gender'],axis=1)**

**#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)**

**# 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)**

**## 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\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**flog\_reg = sm.Logit(fy\_train, fx\_train)**

**flog\_reg = flog\_reg.fit()**

**print(flog\_reg.summary())**

**### 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)**

**#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)**

**# 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)**

**## 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\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**flog\_reg = sm.Logit(fy\_train, fx\_train)**

**flog\_reg = flog\_reg.fit()**

**flog\_reg1 = LogisticRegression()**

**flog\_reg1.fit(fx\_train, fy\_train)**

**print(flog\_reg.summary())**

**## Score of the model giving the accuracy of the model**

**print("Accuracy", (flog\_reg1.score(fx\_test\_minmax, fy\_test)))**

**### PREDICTION ON THE TEST DATASET**

**### 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\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**fx\_train = fx\_train\_minmax**

**fx\_test = fx\_test\_minmax**

**tns\_probs=[0 for \_ in range(len(fy\_test))]**

**fy\_predict = flog\_reg1.predict(fx\_test)**

**## Keeping the probabilities for Testing outcomes**

**fy\_pred = flog\_reg1.predict\_proba(fx\_test)**

**fy\_pred = fy\_pred[:, 1]**

**## 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()**

**## 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)**

**## Accuracy of prediction**

**1-ftest\_error**

**## Sensitivity Analysis**

**ftest\_sens = ftest\_cm[1, 1]/(ftest\_cm[1, 1] + ftest\_cm[0, 1])**

**print(ftest\_sens)**

**## Specificity Analysis**

**ftest\_spec = ftest\_cm[0, 0]/(ftest\_cm[0, 0] + ftest\_cm[1, 0])**

**print(ftest\_spec)**

**## PPV Analysis**

**ftest\_npv = ftest\_cm[1, 1]/(ftest\_cm[1, 1] + ftest\_cm[1, 0])**

**print(ftest\_npv)**

**## NPV Analysis**

**ftest\_npv = ftest\_cm[0, 0]/(ftest\_cm[0, 0] + ftest\_cm[0, 1])**

**print(ftest\_npv)**

**## 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)**

**print(ftest\_auc)**

**print(np.round(fy\_pred\_auc, decimals = 2))**

**## 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)**

**## 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()**

**### BREAST CANCER CASES ###**

**######MLP NEURAL NETWORK CODE IN JUPYTER NOTEBOOK #####**

**## Modules required**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**# Code**

**BC = (pd.read\_excel('cancer.xlsx'))**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**#Import numpy#**

**import numpy as np**

**y = BC.PatStatus**

**x = BC.drop(['PatStatus'], axis = 1)**

**#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)**

**x\_train = x\_train\_minmax**

**x\_test = x\_test\_minmax**

**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\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**##Fitting the neural network model using training dataset**

**tns\_probs=[0 for \_ in range(len(y\_test))]**

**tmlp=MLPClassifier(hidden\_layer\_sizes=(6, 6, 6, 6), activation ='relu', solver = 'adam' ,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)**

**### PREDICTION ON THE TEST DATASET**

**### Getting the prediction for the Testing dataset**

**y\_predict = tmlp.predict(x\_test)**

**## Keeping the probabilities for Testing outcomes**

**y\_pred = tmlp.predict\_proba(x\_test)**

**y\_pred = y\_pred[:,1]**

**## 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()**

**## 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)**

**## Accuracy of prediction**

**1-test\_error**

**## Sensitivity Analysis**

**test\_sens = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[0, 1])**

**print(test\_sens)**

**## Specificity Analysis**

**test\_spec = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[1, 0])**

**print(test\_spec)**

**## PPV Analysis**

**test\_npv = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[1, 0])**

**print(test\_npv)**

**## NPV Analysis**

**test\_npv = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[0, 1])**

**print(test\_npv)**

**## 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)**

**print(test\_auc)**

**print(np.round(y\_pred\_auc, decimals = 2))**

**## 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)**

**## 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()**

**## 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**

**# Code**

**MBC = (pd.read\_excel('MBC.xlsx'))**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**#Import numpy#**

**import numpy as np**

**#### THE MALE DATASET**

**my=MBC.PatStatus**

**mx=MBC.drop(['PatStatus','Gender'], axis=1)**

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

**#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)**

**mx\_train.head()**

**## 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)**

**mx\_train = mx\_train\_minmax**

**mx\_test = mx\_test\_minmax**

**## FITTING NEURAL NETWORK FOR MALE DATA**

**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\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**##Fitting the neural network model using training dataset**

**tns\_probs=[0 for \_ in range(len(my\_test))]**

**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 = 10000, learning\_rate\_init=0.001, power\_t=0.5)**

**male\_mlp.fit(mx\_train, my\_train)**

**### PREDICTION USING THE TEST DATASET**

**### Getting the prediction for the Testing dataset**

**my\_predict = male\_mlp.predict(mx\_test)**

**## Keeping the probabilities for Testing outcomes**

**my\_pred = male\_mlp.predict\_proba(mx\_test)**

**my\_pred = my\_pred[:,1]**

**## 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()**

**## 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)**

**## Accuracy of prediction**

**1-mtest\_error**

**## Sensitivity Analysis**

**mtest\_sens = mtest\_cm[1, 1]/(mtest\_cm[1, 1] + mtest\_cm[0, 1])**

**print(mtest\_sens)**

**## Specificity Analysis**

**mtest\_spec = mtest\_cm[0, 0]/(mtest\_cm[0, 0]+ mtest\_cm[1, 0])**

**print(mtest\_spec)**

**## PPV Analysis**

**mtest\_npv = mtest\_cm[1, 1]/(mtest\_cm[1, 1] + mtest\_cm[1, 0])**

**print(mtest\_npv)**

**## NPV Analysis**

**mtest\_npv = mtest\_cm[0, 0]/(mtest\_cm[0, 0] + mtest\_cm[0, 1])**

**print(mtest\_npv)**

**## 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)**

**print(mtest\_auc)**

**print(np.round(my\_pred\_auc, decimals = 2))**

**## 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)**

**## 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()**

**## 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'))**

**# splitting data into x and y**

**fy=FBC.PatStatus**

**fx=FBC.drop(['PatStatus','Gender'], axis=1)**

**#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)**

**# 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)**

**fx\_train = fx\_train\_minmax**

**fx\_test = fx\_test\_minmax**

**### 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\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**tns\_probs=[0 for \_ in range(len(fy\_test))]**

**female\_mlp=MLPClassifier(hidden\_layer\_sizes=(6, 6, 6, 6), activation ='relu', solver = '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)**

**## PREDICTION USING THE TEST DATASET**

**### Getting the prediction for the Testing dataset**

**fy\_predict = female\_mlp.predict(fx\_test)**

**## Keeping the probabilities for Testing outcomes**

**fy\_pred = female\_mlp.predict\_proba(fx\_test)**

**fy\_pred = fy\_pred[:,1]**

**## 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()**

**## 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)**

**## Accuracy of prediction**

**1-ftest\_error**

**## Sensitivity Analysis**

**ftest\_sens = ftest\_cm[1, 1]/(ftest\_cm[1, 1] + ftest\_cm[0, 1])**

**print(ftest\_sens)**

**## Specificity Analysis**

**ftest\_spec = ftest\_cm[0, 0]/(ftest\_cm[0, 0]+ ftest\_cm[1, 0])**

**print(ftest\_spec)**

**## PPV Analysis**

**ftest\_npv = ftest\_cm[1, 1]/(ftest\_cm[1, 1] + ftest\_cm[1, 0])**

**print(ftest\_npv)**

**## NPV Analysis**

**ftest\_npv = ftest\_cm[0, 0]/(ftest\_cm[0, 0] + ftest\_cm[0, 1])**

**print(ftest\_npv)**

**## 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)**

**print(ftest\_auc)**

**print(np.round(fy\_pred\_auc, decimals = 2))**

**## 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)**

**## 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()**

**### BREAST CANCER CASES ###**

**###### CONVOLUTIONAL NEURAL NETWORK CODE IN JUPYTER NOTEBOOK FOR BOTH SEX #####**

**## Modules required**

**import pandas as pd**

**import numpy as np**

**from scipy import misc**

**from PIL import Image**

**import glob**

**from matplotlib.pyplot import imshow**

**import seaborn as sn**

**import pickle**

**from keras.preprocessing import image**

**from keras.preprocessing.image import load\_img**

**from keras.preprocessing.image import img\_to\_array**

**from keras.applications.imagenet\_utils import decode\_predictions**

**from keras.utils import layer\_utils, np\_utils**

**from keras.utils.data\_utils import get\_file**

**from keras.applications.imagenet\_utils import preprocess\_input**

**from keras.utils.vis\_utils import model\_to\_dot**

**from keras.utils import plot\_model**

**from keras.initializers import glorot\_uniform**

**from keras import losses**

**import keras.backend as K**

**from keras.callbacks import ModelCheckpoint**

**from sklearn.metrics import confusion\_matrix, classification\_report**

**from keras import layers**

**from IPython.display import SVG**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**import keras**

**import tensorflow as tf**

**from tensorflow.keras.models import Sequential, Model, load\_model**

**from tensorflow.keras.layers import Dense,Dropout, Activation, Flatten, Input, Add, ZeroPadding2D, Conv2D, MaxPooling2D**

**from hyperopt import Trials, STATUS\_OK, tpe**

**from hyperas import optim**

**from hyperas.distributions import choice, uniform**

**# Code**

**BC = (pd.read\_excel('cancer.xlsx'))**

**## Reshaping into array**

**import random**

**random.seed(30)**

**BC.iloc[3,1:].values.reshape(6,4).astype('int8')**

**## Preprocessing the data**

**## Storing the independent variables array in form length, width, height into df\_x**

**random.seed(31)**

**df\_x = BC.iloc[:,1:].values.reshape(len(BC), 6, 4, 1)**

**## Storing the dependent variables in y**

**y = BC.iloc[:,0].values**

**# converting y to categorical**

**df\_y = keras.utils.to\_categorical(y, num\_classes = 2)**

**df\_x =np.array(df\_x)**

**df\_y = np.array(df\_y)**

**df\_y**

**df\_x.shape**

**df\_y.shape**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**#Import numpy#**

**import numpy as np**

**random.seed(32)**

**#Split the data into train and test sets #**

**x\_train, x\_test, y\_train, y\_test=train\_test\_split(df\_x,df\_y, test\_size=0.2, random\_state=123)**

**x\_test.shape**

**y\_test.shape**

**### CNN Model**

**random.seed(33)**

**model = Sequential()**

**model.add(Conv2D(64, (3,3), input\_shape = (6, 4, 1)))**

**model.add(Activation('relu'))**

**model.add(MaxPooling2D(pool\_size=(1,1)))**

**model.add(Dropout(0.25))**

**model.add(Flatten())**

**model.add(Dense(64))**

**model.add(Dropout(0.25))**

**model.add(Dense(2))**

**model.add(Activation('sigmoid'))**

**model.compile(loss="categorical\_crossentropy", optimizer="adam", metrics=['accuracy'])**

**model.summary()**

**## fitting the model with**

**CNN\_MODEL = model.fit(x\_train, y\_train, batch\_size=40, epochs=10, validation\_data=(x\_test, y\_test))**

**## MODEL EVALUATION FOR BOTH SEX**

**## Prediction loss and accuracy**

**test\_eval = model.evaluate(x\_test, y\_test, verbose=0)[1]**

**print('Test accuracy:', test\_eval)**

**##plot the accuracy and loss plots between training and validation data to check for over-fitting**

**import numpy as np**

**from keras.utils import to\_categorical**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**accuracy = CNN\_MODEL.history['accuracy']**

**val\_accuracy = CNN\_MODEL.history['val\_accuracy']**

**loss = CNN\_MODEL.history['loss']**

**val\_loss = CNN\_MODEL.history['val\_loss']**

**epochs = range(len(accuracy))**

**plt.plot(epochs, accuracy, 'bo', label='Training accuracy')**

**plt.plot(epochs, val\_accuracy, 'orange', label='Validation accuracy')**

**plt.title('Training and validation accuracy')**

**plt.legend()**

**plt.figure()**

**plt.plot(epochs, loss, 'bo', label='Training loss')**

**plt.plot(epochs, val\_loss, 'orange', label='Validation loss')**

**plt.title('Training and validation loss')**

**plt.legend()**

**plt.show()**

**##plot our training accuracy and validation accuracy**

**plt.plot(CNN\_MODEL.history['accuracy'])**

**plt.plot(CNN\_MODEL.history['val\_accuracy'])**

**plt.title('Model accuracy')**

**plt.ylabel('Accuracy')**

**plt.xlabel('Epoch')**

**plt.legend(['Train', 'Val'], loc='lower right')**

**plt.show()**

**## Predicting using CNN**

**CNN\_MODEL\_pred = model.predict(x\_test, batch\_size=32, verbose=1)**

**CNN\_MODEL\_predicted = np.argmax(CNN\_MODEL\_pred, axis=1)**

**# Confusion matrix for the CNN**

**CNN\_MODEL\_cm = confusion\_matrix(np.argmax(y\_test, axis=1), CNN\_MODEL\_predicted)**

**fig, ax = plt.subplots(figsize = (8, 8))**

**ax.imshow(CNN\_MODEL\_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, CNN\_MODEL\_cm[i, j], ha= 'center', va= 'center', color= 'red')**

**plt.show()**

**test\_cm = CNN\_MODEL\_cm**

**## Sensitivity Analysis**

**test\_sens = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[0, 1])**

**print(test\_sens)**

**## Specificity Analysis**

**test\_spec = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[1, 0])**

**print(test\_spec)**

**## PPV Analysis**

**test\_npv = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[1, 0])**

**print(test\_npv)**

**## NPV Analysis**

**test\_npv = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[0, 1])**

**print(test\_npv)**

**from sklearn.datasets import make\_classification**

**from sklearn.preprocessing import label\_binarize**

**from scipy import interp**

**from itertools import cycle**

**n\_classes = 1**

**from sklearn.metrics import roc\_curve, auc**

**# Plot linewidth.**

**lw = 8**

**# Compute ROC curve and ROC area for each class**

**fpr = dict()**

**tpr = dict()**

**roc\_auc = dict()**

**for i in range(n\_classes):**

**fpr[i], tpr[i], \_ = roc\_curve(y\_test[:, i], CNN\_MODEL\_pred[:, i])**

**roc\_auc[i] = auc(fpr[i], tpr[i])**

**# Compute micro-average ROC curve and ROC area**

**fpr["micro"], tpr["micro"], \_ = roc\_curve(y\_test.ravel(), CNN\_MODEL\_pred.ravel())**

**roc\_auc["micro"] = auc(fpr["micro"], tpr["micro"])**

**# Compute macro-average ROC curve and ROC area**

**# First aggregate all false positive rates**

**all\_fpr = np.unique(np.concatenate([fpr[i] for i in range(n\_classes)]))**

**# Then interpolate all ROC curves at this points**

**mean\_tpr = np.zeros\_like(all\_fpr)**

**for i in range(n\_classes):**

**mean\_tpr += interp(all\_fpr, fpr[i], tpr[i])**

**# Finally average it and compute AUC**

**mean\_tpr /= n\_classes**

**fpr["macro"] = all\_fpr**

**tpr["macro"] = mean\_tpr**

**roc\_auc["macro"] = auc(fpr["macro"], tpr["macro"])**

**# Plot all ROC curves**

**plt.figure(1)**

**plt.plot(fpr["micro"], tpr["micro"],**

**label='Both Sex (AUC = {0:0.1f})'**

**''.format(roc\_auc["micro"]),marker = '.',**

**color='orange', linestyle=':', linewidth=2)**

**plt.plot([0, 1], [0, 1], 'b--', label = 'Patients Last Status',linewidth=2, lw=lw)**

**plt.xlim([0.0, 1.0])**

**plt.ylim([0.0, 1.05])**

**plt.xlabel('False Positive Rate')**

**plt.ylabel('True Positive Rate')**

**plt.title('Receiver Operating Characteristic Curve')**

**plt.legend(loc="lower right")**

**plt.show()**

**## CONSIDER THE CONVOLUTIONAL NEURAL NETWORK FOR EACH GENDER SEPARATELY**

**## Modules required**

**import pandas as pd**

**import numpy as np**

**from scipy import misc**

**from PIL import Image**

**import glob**

**from matplotlib.pyplot import imshow**

**import seaborn as sn**

**import pickle**

**from keras.preprocessing import image**

**from keras.preprocessing.image import load\_img**

**from keras.preprocessing.image import img\_to\_array**

**from keras.applications.imagenet\_utils import decode\_predictions**

**from keras.utils import layer\_utils, np\_utils**

**from keras.utils.data\_utils import get\_file**

**from keras.applications.imagenet\_utils import preprocess\_input**

**from keras.utils.vis\_utils import model\_to\_dot**

**from keras.utils import plot\_model**

**from keras.initializers import glorot\_uniform**

**from keras import losses**

**import keras.backend as K**

**from keras.callbacks import ModelCheckpoint**

**from sklearn.metrics import confusion\_matrix, classification\_report**

**from keras import layers**

**from IPython.display import SVG**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**import keras**

**import tensorflow as tf**

**from tensorflow.keras.models import Sequential, Model, load\_model**

**from tensorflow.keras.layers import Dense,Dropout, Activation, Flatten, Input, Add, ZeroPadding2D, Conv2D, MaxPooling2D**

**from hyperopt import Trials, STATUS\_OK, tpe**

**from hyperas import optim**

**from hyperas.distributions import choice, uniform**

**# Code**

**MBC = (pd.read\_excel('MBC.xlsx'))**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**#Import numpy#**

**import numpy as np**

**## CONSIDER FITTING CONVOLUTIONAL NEURAL NETWORK FOR THE MALE GENDER**

**MBC.head()**

**## Reshaping into array**

**MBC.iloc[3,1:].values.reshape(6,4).astype('int8')**

**## Preprocessing the data**

**## Storing the independent variables array in form length, width, height into df\_x**

**df\_x = MBC.iloc[:,1:].values.reshape(len(MBC), 6, 4, 1)**

**## Storing the dependent variables in y**

**y = MBC.iloc[:,0].values**

**# converting y to categorical**

**df\_y = keras.utils.to\_categorical(y, num\_classes = 2)**

**df\_x =np.array(df\_x)**

**df\_y = np.array(df\_y)**

**df\_y**

**df\_x.shape**

**df\_y.shape**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**#Import numpy#**

**import numpy as np**

**#Split the data into train and test sets #**

**x\_train, x\_test, y\_train, y\_test=train\_test\_split(df\_x,df\_y, test\_size=0.2, random\_state=123)**

**x\_test.shape**

**y\_test.shape**

**## CNN Model**

**model = Sequential()**

**model.add(Conv2D(64, (3,3), input\_shape = (6, 4, 1)))**

**model.add(Activation("relu"))**

**model.add(MaxPooling2D(pool\_size=(1,1)))**

**model.add(Dropout(0.25))**

**model.add(Flatten())**

**model.add(Dense(64))**

**model.add(Dropout(0.25))**

**model.add(Dense(2))**

**model.add(Activation('sigmoid'))**

**model.compile(loss="categorical\_crossentropy", optimizer="adam", metrics=['accuracy'])**

**model.summary()**

**## fitting the model with**

**CNN\_MODEL = model.fit(x\_train, y\_train, batch\_size=40, epochs=10, validation\_data=(x\_test, y\_test))**

**## MODEL EVALUATION**

**## Prediction loss and accuracy**

**test\_eval = model.evaluate(x\_test, y\_test, verbose=0)[1]**

**print('Test accuracy:', test\_eval)**

**##plot the accuracy and loss plots between training and validation data to check for over-fitting**

**import numpy as np**

**from keras.utils import to\_categorical**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**accuracy = CNN\_MODEL.history['accuracy']**

**val\_accuracy = CNN\_MODEL.history['val\_accuracy']**

**loss = CNN\_MODEL.history['loss']**

**val\_loss = CNN\_MODEL.history['val\_loss']**

**epochs = range(len(accuracy))**

**plt.plot(epochs, accuracy, 'bo', label='Training accuracy')**

**plt.plot(epochs, val\_accuracy, 'orange', label='Validation accuracy')**

**plt.title('Training and validation accuracy')**

**plt.legend()**

**plt.figure()**

**plt.plot(epochs, loss, 'bo', label='Training loss')**

**plt.plot(epochs, val\_loss, 'orange', label='Validation loss')**

**plt.title('Training and validation loss')**

**plt.legend()**

**plt.show()**

**##plot our training accuracy and validation accuracy**

**plt.plot(CNN\_MODEL.history['accuracy'])**

**plt.plot(CNN\_MODEL.history['val\_accuracy'])**

**plt.title('Model accuracy')**

**plt.ylabel('Accuracy')**

**plt.xlabel('Epoch')**

**plt.legend(['Train', 'Val'], loc='lower right')**

**plt.show()**

**## Predicting using CNN**

**CNN\_MODEL\_pred = model.predict(x\_test, batch\_size=32, verbose=1)**

**CNN\_MODEL\_predicted = np.argmax(CNN\_MODEL\_pred, axis=1)**

**## Confusion matrix for the CNN**

**CNN\_MODEL\_cm = confusion\_matrix(np.argmax(y\_test, axis=1), CNN\_MODEL\_predicted)**

**fig, ax = plt.subplots(figsize = (8, 8))**

**ax.imshow(CNN\_MODEL\_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, CNN\_MODEL\_cm[i, j], ha= 'center', va= 'center', color= 'k')**

**plt.show()**

**test\_cm = CNN\_MODEL\_cm**

**## Sensitivity Analysis**

**test\_sens = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[0, 1])**

**print(test\_sens)**

**## Specificity Analysis**

**test\_spec = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[1, 0])**

**print(test\_spec)**

**## PPV Analysis**

**test\_npv = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[1, 0])**

**print(test\_npv)**

**## NPV Analysis**

**test\_npv = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[0, 1])**

**print(test\_npv)**

**from sklearn.datasets import make\_classification**

**from sklearn.preprocessing import label\_binarize**

**from scipy import interp**

**from itertools import cycle**

**n\_classes = 1**

**from sklearn.metrics import roc\_curve, auc**

**# Plot linewidth.**

**lw = 8**

**# Compute ROC curve and ROC area for each class**

**fpr = dict()**

**tpr = dict()**

**roc\_auc = dict()**

**for i in range(n\_classes):**

**fpr[i], tpr[i], \_ = roc\_curve(y\_test[:, i], CNN\_MODEL\_pred[:, i])**

**roc\_auc[i] = auc(fpr[i], tpr[i])**

**# Compute micro-average ROC curve and ROC area**

**fpr["micro"], tpr["micro"], \_ = roc\_curve(y\_test.ravel(), CNN\_MODEL\_pred.ravel())**

**roc\_auc["micro"] = auc(fpr["micro"], tpr["micro"])**

**# Compute macro-average ROC curve and ROC area**

**# First aggregate all false positive rates**

**all\_fpr = np.unique(np.concatenate([fpr[i] for i in range(n\_classes)]))**

**# Then interpolate all ROC curves at this points**

**mean\_tpr = np.zeros\_like(all\_fpr)**

**for i in range(n\_classes):**

**mean\_tpr += interp(all\_fpr, fpr[i], tpr[i])**

**# Finally average it and compute AUC**

**mean\_tpr /= n\_classes**

**fpr["macro"] = all\_fpr**

**tpr["macro"] = mean\_tpr**

**roc\_auc["macro"] = auc(fpr["macro"], tpr["macro"])**

**# Plot all ROC curves**

**plt.figure(1)**

**plt.plot(fpr["micro"], tpr["micro"],**

**label='Male (AUC = {0:0.2f})'**

**''.format(roc\_auc["micro"]),marker = '.',**

**color='orange', linestyle=':', linewidth=2)**

**plt.plot([0, 1], [0, 1], 'b--', label = 'Patients Last Status',linewidth=2, lw=lw)**

**plt.xlim([0.0, 1.0])**

**plt.ylim([0.0, 1.05])**

**plt.xlabel('False Positive Rate')**

**plt.ylabel('True Positive Rate')**

**plt.title('Receiver Operating Characteristic Curve')**

**plt.legend(loc="lower right")**

**plt.show()**

**## CONSIDERING THE FEMALE DATA SEPARATELY FOR THE ANALYSIS**

**## Modules required**

**import pandas as pd**

**import numpy as np**

**from scipy import misc**

**from PIL import Image**

**import glob**

**from matplotlib.pyplot import imshow**

**import seaborn as sn**

**import pickle**

**from keras.preprocessing import image**

**from keras.preprocessing.image import load\_img**

**from keras.preprocessing.image import img\_to\_array**

**from keras.applications.imagenet\_utils import decode\_predictions**

**from keras.utils import layer\_utils, np\_utils**

**from keras.utils.data\_utils import get\_file**

**from keras.applications.imagenet\_utils import preprocess\_input**

**from keras.utils.vis\_utils import model\_to\_dot**

**from keras.utils import plot\_model**

**from keras.initializers import glorot\_uniform**

**from keras import losses**

**import keras.backend as K**

**from keras.callbacks import ModelCheckpoint**

**from sklearn.metrics import confusion\_matrix, classification\_report**

**from keras import layers**

**from IPython.display import SVG**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**import keras**

**import tensorflow as tf**

**from tensorflow.keras.models import Sequential, Model, load\_model**

**from tensorflow.keras.layers import Dense,Dropout, Activation, Flatten, Input, Add, ZeroPadding2D, Conv2D, MaxPooling2D**

**from hyperopt import Trials, STATUS\_OK, tpe**

**from hyperas import optim**

**from hyperas.distributions import choice, uniform**

**# Code**

**FBC = (pd.read\_excel('FBC.xlsx'))**

**## CONSIDERING THE FEMALE DATA SEPARATELY FOR THE ANALYSIS**

**FBC.head()**

**# 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**

**import keras**

**import tensorflow as tf**

**from tensorflow.keras.models import Sequential**

**from tensorflow.keras.layers import Dense,Dropout, Activation, Flatten, Conv2D, MaxPooling2D**

**from hyperopt import Trials, STATUS\_OK, tpe**

**from hyperas import optim**

**from hyperas.distributions import choice, uniform**

**## Reshaping into array**

**FBC.iloc[3,1:].values.reshape(6,4).astype('int8')**

**## Preprocessing the data**

**## Storing the independent variables array in form length, width, height into df\_x**

**df\_x = FBC.iloc[:,1:].values.reshape(len(FBC), 6, 4, 1)**

**## Storing the dependent variables in y**

**y = FBC.iloc[:,0].values**

**# converting y to categorical**

**df\_y = keras.utils.to\_categorical(y, num\_classes = 2)**

**df\_x =np.array(df\_x)**

**df\_y = np.array(df\_y)**

**df\_y**

**df\_x.shape**

**df\_y.shape**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**#Import numpy#**

**import numpy as np**

**#Split the data into train and test sets #**

**x\_train, x\_test, y\_train, y\_test=train\_test\_split(df\_x,df\_y, test\_size=0.2, random\_state=123)**

**x\_test.shape**

**y\_test.shape**

**### CNN Model**

**model = Sequential()**

**model.add(Conv2D(64, (2,3), input\_shape = (6, 4, 1)))**

**model.add(Activation('relu'))**

**model.add(MaxPooling2D(pool\_size=(1,1)))**

**model.add(Dropout(0.25))**

**model.add(Flatten())**

**model.add(Dense(64))**

**model.add(Dropout(0.25))**

**model.add(Dense(2))**

**model.add(Activation('sigmoid'))**

**model.compile(loss="categorical\_crossentropy", optimizer="adam", metrics=['accuracy'])**

**model.summary()**

**## fitting the model with**

**CNN\_MODEL = model.fit(x\_train, y\_train, batch\_size=30, epochs=10, validation\_data=(x\_test, y\_test))**

**## MODEL EVALUATION**

**## Prediction loss and accuracy**

**test\_eval = model.evaluate(x\_test, y\_test, verbose=0)[1]**

**print('Test accuracy:', test\_eval)**

**##plot the accuracy and loss plots between training and validation data to check for over-fitting**

**import numpy as np**

**from keras.utils import to\_categorical**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**accuracy = CNN\_MODEL.history['accuracy']**

**val\_accuracy = CNN\_MODEL.history['val\_accuracy']**

**loss = CNN\_MODEL.history['loss']**

**val\_loss = CNN\_MODEL.history['val\_loss']**

**epochs = range(len(accuracy))**

**plt.plot(epochs, accuracy, 'bo', label='Training accuracy')**

**plt.plot(epochs, val\_accuracy, 'orange', label='Validation accuracy')**

**plt.title('Training and validation accuracy')**

**plt.legend()**

**plt.figure()**

**plt.plot(epochs, loss, 'bo', label='Training loss')**

**plt.plot(epochs, val\_loss, 'orange', label='Validation loss')**

**plt.title('Training and validation loss')**

**plt.legend()**

**plt.show()**

**##plot our training accuracy and validation accuracy**

**plt.plot(CNN\_MODEL.history['accuracy'])**

**plt.plot(CNN\_MODEL.history['val\_accuracy'])**

**plt.title('Model accuracy')**

**plt.ylabel('Accuracy')**

**plt.xlabel('Epoch')**

**plt.legend(['Train', 'Val'], loc='lower right')**

**plt.show()**

**## Predicting using CNN**

**CNN\_MODEL\_pred = model.predict(x\_test, batch\_size=32, verbose=1)**

**CNN\_MODEL\_predicted = np.argmax(CNN\_MODEL\_pred, axis=1)**

**## Confusion matrix for the CNN**

**CNN\_MODEL\_cm = confusion\_matrix(np.argmax(y\_test, axis=1), CNN\_MODEL\_predicted)**

**fig, ax = plt.subplots(figsize = (8, 8))**

**ax.imshow(CNN\_MODEL\_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, CNN\_MODEL\_cm[i, j], ha= 'center', va= 'center', color= 'k')**

**plt.show()**

**test\_cm = CNN\_MODEL\_cm**

**## Sensitivity Analysis**

**test\_sens = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[0, 1])**

**print(test\_sens)**

**## Specificity Analysis**

**test\_spec = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[1, 0])**

**print(test\_spec)**

**## PPV Analysis**

**test\_npv = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[1, 0])**

**print(test\_npv)**

**## NPV Analysis**

**test\_npv = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[0, 1])**

**print(test\_npv)**

**from sklearn.datasets import make\_classification**

**from sklearn.preprocessing import label\_binarize**

**from scipy import interp**

**from itertools import cycle**

**n\_classes = 1**

**from sklearn.metrics import roc\_curve, auc**

**# Plot linewidth.**

**lw = 8**

**# Compute ROC curve and ROC area for each class**

**fpr = dict()**

**tpr = dict()**

**roc\_auc = dict()**

**for i in range(n\_classes):**

**fpr[i], tpr[i], \_ = roc\_curve(y\_test[:, i], CNN\_MODEL\_pred[:, i])**

**roc\_auc[i] = auc(fpr[i], tpr[i])**

**# Compute micro-average ROC curve and ROC area**

**fpr["micro"], tpr["micro"], \_ = roc\_curve(y\_test.ravel(), CNN\_MODEL\_pred.ravel())**

**roc\_auc["micro"] = auc(fpr["micro"], tpr["micro"])**

**# Compute macro-average ROC curve and ROC area**

**# First aggregate all false positive rates**

**all\_fpr = np.unique(np.concatenate([fpr[i] for i in range(n\_classes)]))**

**# Then interpolate all ROC curves at this points**

**mean\_tpr = np.zeros\_like(all\_fpr)**

**for i in range(n\_classes):**

**mean\_tpr += interp(all\_fpr, fpr[i], tpr[i])**

**# Finally average it and compute AUC**

**mean\_tpr /= n\_classes**

**fpr["macro"] = all\_fpr**

**tpr["macro"] = mean\_tpr**

**roc\_auc["macro"] = auc(fpr["macro"], tpr["macro"])**

**# Plot all ROC curves**

**plt.figure(1)**

**plt.plot(fpr["micro"], tpr["micro"],**

**label='Female (AUC = {0:0.1f})'**

**''.format(roc\_auc["micro"]),marker = '.',**

**color='orange', linestyle=':', linewidth=2)**

**plt.plot([0, 1], [0, 1], 'b--', label = 'Patients Last Status',linewidth=2, lw=lw)**

**plt.xlim([0.0, 1.0])**

**plt.ylim([0.0, 1.05])**

**plt.xlabel('False Positive Rate')**

**plt.ylabel('True Positive Rate')**

**plt.title('Receiver Operating Characteristic Curve')**

**plt.legend(loc="lower right")**

**plt.show()**

**### BREAST CANCER CASES ###**

**###### SVM WITH RBF KERNEL CODE IN JUPYTER NOTEBOOK #####**

**## Modules required**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**import math**

**import keras**

**import tensorflow as tf**

**import warnings**

**warnings.filterwarnings("ignore")**

**# Code**

**BC = (pd.read\_excel('cancer.xlsx'))**

**BC.head()**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**#Import numpy#**

**import numpy as np**

**y = BC.PatStatus**

**x = BC.drop(['PatStatus'], axis = 1)**

**#Split the data into train and test sets #**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**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)**

**x\_train = x\_train\_minmax**

**x\_test = x\_test\_minmax**

**x\_train.shape**

**x\_test.shape**

**## Fitting the model**

**## Models required**

**from keras.preprocessing import image**

**from keras.preprocessing.image import load\_img**

**from keras.preprocessing.image import img\_to\_array**

**from keras.applications.imagenet\_utils import decode\_predictions**

**from keras.utils import layer\_utils, np\_utils**

**from sklearn.metrics import confusion\_matrix, classification\_report**

**import tensorflow as tf**

**from hyperas import optim**

**from hyperas.distributions import choice, uniform**

**import warnings**

**warnings.filterwarnings("ignore")**

**from sklearn.datasets import make\_classification**

**from sklearn.metrics import confusion\_matrix, roc\_curve, roc\_auc\_score, plot\_roc\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**from sklearn.svm import SVC**

**# SVM classifier with Gaussian RBF kernel**

**classifier = SVC(kernel='rbf',random\_state=0)**

**classifier.fit(x\_train,y\_train)**

**# predict with splitted test data**

**y\_pred = classifier.predict(x\_test)**

**##Fitting the neural network model using training dataset**

**tns\_probs=[0 for \_ in range(len(y\_test))]**

**## CONFUSION MATRIX FOR BOTH SEX DATA**

**test\_cm = confusion\_matrix(y\_test, y\_pred)**

**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= 'red')**

**plt.show()**

**## 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)**

**## Accuracy of prediction**

**1-test\_error**

**## Sensitivity Analysis**

**test\_sens = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[0, 1])**

**print(test\_sens)**

**## Specificity Analysis**

**test\_spec = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[1, 0])**

**print(test\_spec)**

**## PPV Analysis**

**test\_npv = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[1, 0])**

**print(test\_npv)**

**## NPV Analysis**

**test\_npv = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[0, 1])**

**print(test\_npv)**

**## 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)**

**## 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)**

**## 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()**

**## 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**

**# Code**

**MBC = (pd.read\_excel('MBC.xlsx'))**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**#Import numpy#**

**import numpy as np**

**#### THE MALE DATASET**

**my=MBC.PatStatus**

**mx=MBC.drop(['PatStatus', 'Gender'], axis=1)**

**## CONSIDER RBF FITTING FOR THE MALE GENDER**

**#Split the Male data into train and test sets #**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**#Import numpy#**

**import numpy as np**

**mx\_train, mx\_test, my\_train, my\_test=train\_test\_split(mx,my, test\_size=0.2, random\_state=124)**

**mx\_train.head()**

**mx\_test.head()**

**mx\_train.shape**

**mx\_test.shape**

**## 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)**

**mx\_train = mx\_train\_minmax**

**mx\_test = mx\_test\_minmax**

**## FITTING NEURAL NETWORK FOR MALE DATA**

**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\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**##Fitting the neural network model using training dataset**

**tns\_probs=[0 for \_ in range(len(my\_test))]**

**## Fitting the model**

**# SVM classifier with Gaussian RBF kernel**

**from sklearn.svm import SVC**

**classifier = SVC(kernel='rbf',random\_state=0)**

**classifier.fit(mx\_train,my\_train)**

**### PREDICTION USING THE TEST DATASET**

**### Getting the prediction for the Testing dataset**

**my\_pred = classifier.predict(mx\_test)**

**# CONFUSION MATRIX FOR MALE DATA**

**mtest\_cm = confusion\_matrix(my\_test, my\_pred)**

**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= 'red')**

**plt.show()**

**# 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)**

**## Accuracy of prediction**

**1-mtest\_error**

**## Sensitivity Analysis**

**mtest\_sens = mtest\_cm[1, 1]/(mtest\_cm[1, 1] + mtest\_cm[0, 1])**

**print(mtest\_sens)**

**## Specificity Analysis**

**mtest\_spec = mtest\_cm[0, 0]/(mtest\_cm[0, 0]+ mtest\_cm[1, 0])**

**print(mtest\_spec)**

**## PPV Analysis**

**mtest\_npv = mtest\_cm[1, 1]/(mtest\_cm[1, 1] + mtest\_cm[1, 0])**

**print(mtest\_npv)**

**## NPV Analysis**

**mtest\_npv = mtest\_cm[0, 0]/(mtest\_cm[0, 0] + mtest\_cm[0, 1])**

**print(mtest\_npv)**

**## 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)**

**## 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)**

**## 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()**

**## CONSIDERING THE FEMALE DATA**

**## Modules required**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**FBC = (pd.read\_excel('FBC.xlsx'))**

**fy=FBC.PatStatus**

**fx=FBC.drop(['PatStatus','Gender'],axis=1)**

**#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)**

**# 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)**

**fx\_train = fx\_train\_minmax**

**fx\_test = fx\_test\_minmax**

**### 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\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**tns\_probs=[0 for \_ in range(len(fy\_test))]**

**## Fitting the model**

**# SVM classifier with Gaussian RBF kernel**

**from sklearn.svm import SVC**

**classifier = SVC(kernel='rbf',random\_state=0)**

**classifier.fit(fx\_train,fy\_train)**

**## PREDICTION USING THE TEST DATASET**

**### Getting the prediction for the Testing dataset**

**fy\_pred = classifier.predict(fx\_test)**

**## CONFUSION MATRIX FOR MALE DATA**

**ftest\_cm = confusion\_matrix(fy\_test, fy\_pred)**

**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= 'red')**

**plt.show()**

**## 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)**

**## Accuracy of prediction**

**1-ftest\_error**

**## Sensitivity Analysis**

**ftest\_sens = ftest\_cm[1, 1]/(ftest\_cm[1, 1] + ftest\_cm[0, 1])**

**print(ftest\_sens)**

**## Specificity Analysis**

**ftest\_spec = ftest\_cm[0, 0]/(ftest\_cm[0, 0]+ ftest\_cm[1, 0])**

**print(ftest\_spec)**

**## PPV Analysis**

**ftest\_npv = ftest\_cm[1, 1]/(ftest\_cm[1, 1] + ftest\_cm[1, 0])**

**print(ftest\_npv)**

**## NPV Analysis**

**ftest\_npv = ftest\_cm[0, 0]/(ftest\_cm[0, 0] + ftest\_cm[0, 1])**

**print(ftest\_npv)**

**## 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)**

**## 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)**

**## 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()**

**### BREAST CANCER CASES ###**

**###### Random Forest CODE IN JUPYTER NOTEBOOK #####**

**## Modules required**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**import math**

**import keras**

**import tensorflow as tf**

**import warnings**

**# Code**

**BC = (pd.read\_excel('cancer.xlsx'))**

**BC.head()**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**#Import numpy#**

**import numpy as np**

**y = BC.PatStatus**

**x = BC.drop(['PatStatus'], axis = 1)**

**#Split the data into train and test sets #**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**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)**

**x\_train = x\_train\_minmax**

**x\_test = x\_test\_minmax**

**x\_train.shape**

**x\_test.shape**

**## Fitting the model**

**## Models required**

**from keras.applications.imagenet\_utils import decode\_predictions**

**import tensorflow as tf**

**from hyperas.distributions import choice, uniform**

**from sklearn.datasets import make\_classification**

**from sklearn.metrics import confusion\_matrix, roc\_curve, roc\_auc\_score, plot\_roc\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**from sklearn.ensemble import RandomForestClassifier**

**#Create a Gaussian Classifier**

**clf=RandomForestClassifier(n\_estimators=100)**

**#Train the model using the training sets**

**rand\_forest\_model=clf.fit(x\_train,y\_train)**

**rand\_forest\_model**

**# predict with splitted test data**

**y\_pred = clf.predict(x\_test)**

**##Fitting the neural network model using training dataset**

**tns\_probs=[0 for \_ in range(len(y\_test))]**

**#Import scikit-learn metrics module for accuracy calculation**

**from sklearn import metrics**

**# Model Accuracy, how often is the classifier correct?**

**print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))**

**## CONFUSION MATRIX FOR BOTH SEX DATA**

**test\_cm = confusion\_matrix(y\_test, np.round(y\_pred))**

**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= 'red')**

**plt.show()**

**## 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)**

**## Accuracy of prediction**

**1-test\_error**

**## Sensitivity Analysis**

**test\_sens = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[0, 1])**

**print(test\_sens)**

**## Specificity Analysis**

**test\_spec = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[1, 0])**

**print(test\_spec)**

**## PPV Analysis**

**test\_npv = test\_cm[1, 1]/(test\_cm[1, 1] + test\_cm[1, 0])**

**print(test\_npv)**

**## NPV Analysis**

**test\_npv = test\_cm[0, 0]/(test\_cm[0, 0]+test\_cm[0, 1])**

**print(test\_npv)**

**## 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)**

**## 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)**

**## 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()**

**## 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**

**# Code**

**MBC = (pd.read\_excel('MBC.xlsx'))**

**#Import 'train\_test\_split' from 'sklearn.model\_selection'**

**from sklearn.model\_selection import train\_test\_split**

**#Import numpy#**

**import numpy as np**

**#### THE MALE DATASET**

**my=MBC.PatStatus**

**mx=MBC.drop(['PatStatus', 'Gender'], axis=1)**

**# CONSIDER RBF FITTING FOR THE MALE GENDER**

**#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)**

**mx\_train.head()**

**mx\_test.head()**

**mx\_train.shape**

**mx\_test.shape**

**## 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)**

**mx\_train = mx\_train\_minmax**

**mx\_test = mx\_test\_minmax**

**mx\_train = np.array(mx\_train)**

**mx\_test = np.array(mx\_test)**

**my\_train = np.array(my\_train)**

**my\_test = np.array(my\_test)**

**# FITTING NEURAL NETWORK FOR MALE DATA**

**from sklearn.datasets import make\_classification**

**from sklearn.metrics import confusion\_matrix, roc\_curve, roc\_auc\_score, plot\_roc\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**#Fitting the neural network model using training dataset**

**tns\_probs=[0 for \_ in range(len(my\_test))]**

**## Fitting the model**

**## Models required**

**from keras.applications.imagenet\_utils import decode\_predictions**

**import tensorflow as tf**

**from hyperas.distributions import choice, uniform**

**from sklearn.datasets import make\_classification**

**from sklearn.metrics import confusion\_matrix, roc\_curve, roc\_auc\_score, plot\_roc\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**from sklearn.ensemble import RandomForestClassifier**

**#Create a Gaussian Classifier**

**clf=RandomForestClassifier(n\_estimators=100, random\_state = 5)**

**#Train the model using the training sets**

**mrand\_forest\_model=clf.fit(mx\_train, my\_train)**

**mrand\_forest\_model**

**### PREDICTION USING THE TEST DATASET**

**# predict with splitted test data**

**my\_pred = clf.predict(mx\_test)**

**#Import scikit-learn metrics module for accuracy calculation**

**from sklearn import metrics**

**# Model Accuracy, how often is the classifier correct?**

**print("Accuracy:",metrics.accuracy\_score(my\_test, my\_pred))**

**feature importance variable**

**import pandas as pd**

**MBC = pd.DataFrame(MBC.values, columns=[["PatStatus",**

**"Race","MarST",**

**"Gender",**

**"AgeDiag",**

**"Grade",**

**"Stability",**

**"No.Visits",**

**"Lstay",**

**"Laterality",**

**"FamHist",**

**"PrioBSurgy",**

**"Suture",**

**"Density",**

**"NipRet",**

**"LyNode",**

**"Amorph",**

**"Size",**

**"Eggshell",**

**"Milk",**

**"AxiAden",**

**"Distroph",**

**"Lucent",**

**"Dermal",**

**"SkinnLesson"**

**]])**

**feature\_imp = pd.Series(clf.feature\_importances\_, index = [**

**"Race",**

**"MarST",**

**"AgeDiag",**

**"Grade",**

**"Stability",**

**"No.Visits",**

**"Lstay",**

**"Laterality",**

**"FamHist",**

**"PrioBSurgy",**

**"Suture",**

**"Density",**

**"NipRet",**

**"LyNode",**

**"Amorph",**

**"Size",**

**"Eggshell",**

**"Milk",**

**"AxiAden",**

**"Distroph",**

**"Lucent",**

**"Dermal",**

**"SkinnLesson"]).sort\_values(ascending =False)**

**feature\_imp**

**# List of features for later use**

**feature\_list = list(MBC.columns)**

**# Get numerical feature importances**

**importances = list(clf.feature\_importances\_)**

**# List of tuples with variable and importance**

**feature\_importances = [(feature, round(importance, 2)) for feature, importance in zip(feature\_list, importances)]**

**# list of x locations for plotting**

**x\_values = list(range(len(importances)))**

**# use the feature importance variable to see feature importance scores**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**%matplotlib inline**

**plt.figure(figsize=(8,8))**

**# Creating a bar plot**

**sns.barplot(x=feature\_imp, y=feature\_imp.index)**

**# Add labels to your graph**

**plt.xlabel('Feature Importance Score')**

**plt.ylabel('Features')**

**plt.title("Visualizing Important Features")**

**#plt.legend()**

**plt.show()**

**##Fitting the neural network model using training dataset**

**tns\_probs=[0 for \_ in range(len(my\_test))]**

**## CONFUSION MATRIX FOR MALE DATA**

**mtest\_cm = confusion\_matrix(my\_test, np.round(my\_pred))**

**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= 'red')**

**plt.show()**

**## 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)**

**## Accuracy of prediction**

**1-mtest\_error**

**## Sensitivity Analysis**

**mtest\_sens = mtest\_cm[1, 1]/(mtest\_cm[1, 1] + mtest\_cm[0, 1])**

**print(mtest\_sens)**

**# Specificity Analysis**

**mtest\_spec = mtest\_cm[0, 0]/(mtest\_cm[0, 0]+ mtest\_cm[1, 0])**

**print(mtest\_spec)**

**# PPV Analysis**

**mtest\_npv = mtest\_cm[1, 1]/(mtest\_cm[1, 1] + mtest\_cm[1, 0])**

**print(mtest\_npv)**

**## NPV Analysis**

**mtest\_npv = mtest\_cm[0, 0]/(mtest\_cm[0, 0] + mtest\_cm[0, 1])**

**print(mtest\_npv)**

**## 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)**

**## 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)**

**## 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()**

**## CONSIDERING THE FEMALE DATA**

**## Modules required**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**%matplotlib inline**

**FBC = (pd.read\_excel('FBC.xlsx'))**

**fy=FBC.PatStatus**

**fx=FBC.drop(['PatStatus', 'Gender'],axis=1)**

**FBC.shape**

**#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)**

**# 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)**

**fx\_train = fx\_train\_minmax**

**fx\_test = fx\_test\_minmax**

**### 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\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**tns\_probs=[0 for \_ in range(len(fy\_test))]**

**## Fitting the model**

**## Models required**

**from keras.applications.imagenet\_utils import decode\_predictions**

**import tensorflow as tf**

**from hyperas.distributions import choice, uniform**

**from sklearn.datasets import make\_classification**

**from sklearn.metrics import confusion\_matrix, roc\_curve, roc\_auc\_score, plot\_roc\_curve**

**from sklearn.model\_selection import cross\_val\_score, cross\_validate**

**from sklearn.ensemble import RandomForestClassifier**

**#Create a Gaussian Classifier**

**clf=RandomForestClassifier(n\_estimators=5, random\_state = 5)**

**#Train the model using the training sets**

**frand\_forest\_model=clf.fit(fx\_train,fy\_train)**

**frand\_forest\_model**

**## PREDICTION USING THE TEST DATASET**

**### Getting the prediction for the Testing dataset**

**fy\_pred = clf.predict(fx\_test)**

**#Import scikit-learn metrics module for accuracy calculation**

**from sklearn import metrics**

**# Model Accuracy, how often is the classifier correct?**

**print("Accuracy:",metrics.accuracy\_score(fy\_test, fy\_pred))**

**# feature importance variable**

**import pandas as pd**

**FBC = pd.DataFrame(FBC.values, columns=[["PatStatus",**

**"Race","MarST",**

**"Gender",**

**"AgeDiag",**

**"Grade",**

**"Stability",**

**"No.Visits",**

**"Lstay",**

**"Laterality",**

**"FamHist",**

**"PrioBSurgy",**

**"Suture",**

**"Density",**

**"NipRet",**

**"LyNode",**

**"Amorph",**

**"Size",**

**"Eggshell",**

**"Milk",**

**"AxiAden",**

**"Distroph",**

**"Lucent",**

**"Dermal",**

**"SkinnLesson"**

**]])**

**feature\_imp = pd.Series(clf.feature\_importances\_, index = [**

**"Race","MarST",**

**"AgeDiag",**

**"Grade",**

**"Stability",**

**"No.Visits",**

**"Lstay",**

**"Laterality",**

**"FamHist",**

**"PrioBSurgy",**

**"Suture",**

**"Density",**

**"NipRet",**

**"LyNode",**

**"Amorph",**

**"Size",**

**"Eggshell",**

**"Milk",**

**"AxiAden",**

**"Distroph",**

**"Lucent",**

**"Dermal",**

**"SkinnLesson"]).sort\_values(ascending =False)**

**feature\_imp**

**# List of features for later use**

**feature\_list = list(FBC.columns)**

**# Get numerical feature importances**

**importances = list(clf.feature\_importances\_)**

**# List of tuples with variable and importance**

**feature\_importances = [(feature, round(importance, 2)) for feature, importance in zip(feature\_list, importances)]**

**# list of x locations for plotting**

**x\_values = list(range(len(importances)))**

**# use the feature importance variable to see feature importance scores**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**%matplotlib inline**

**plt.figure(figsize=(8,8))**

**# Creating a bar plot**

**sns.barplot(x=feature\_imp, y=feature\_imp.index)**

**# Add labels to your graph**

**plt.xlabel('Feature Importance Score')**

**plt.ylabel('Features')**

**plt.title("Visualizing Important Features")**

**#plt.legend()**

**plt.show()**

**## CONFUSION MATRIX FOR MALE DATA**

**ftest\_cm = confusion\_matrix(fy\_test, np.round(fy\_pred))**

**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= 'red')**

**plt.show()**

**## 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)**

**## Accuracy of prediction**

**1-ftest\_error**

**## Sensitivity Analysis**

**ftest\_sens = ftest\_cm[1, 1]/(ftest\_cm[1, 1] + ftest\_cm[0, 1])**

**print(ftest\_sens)**

**## Specificity Analysis**

**ftest\_spec = ftest\_cm[0, 0]/(ftest\_cm[0, 0]+ ftest\_cm[1, 0])**

**print(ftest\_spec)**

**## PPV Analysis**

**ftest\_npv = ftest\_cm[1, 1]/(ftest\_cm[1, 1] + ftest\_cm[1, 0])**

**print(ftest\_npv)**

**## NPV Analysis**

**ftest\_npv = ftest\_cm[0, 0]/(ftest\_cm[0, 0] + ftest\_cm[0, 1])**

**print(ftest\_npv)**

**## 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)**

**## 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)**

**## 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()**