IIT ROORKEE

FREE ONLINE EDUCATION swayam शिक्षित भारत, उन्नत भारत

NPTEL ONLINE CERTIFICATION COURSE

Performance of Logistic Model-III

Dr A. RAMESH

DEPARTMENT OF MANAGEMENT STUDIES

Agenda

Python demo for accuracy prediction in logistic regression model using Receiver operating characteristics curve

Sensitivity and Specificity

- For checking, what type of error we are making; we use two parameters-

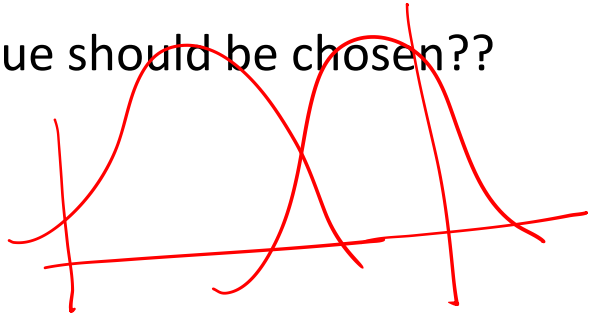
1. Sensitivity = tp/(tp+fn) ⟶ True Positive Rate(tpr)

2. Specificity = tn/(tn+fp) ⟶ True Negative Rate (tnr)

Specificity and Sensitivity Relationship with Threshold

Threshold (Lower)	Sensitivity (↑)	Specificity (↓)
Threshold (Higher)	Sensitivity (↓)	Specificity (↑)

Which threshold value should be chosen??



Measuring Accuracy, Specificity and Sensitivity

In [20]:
```
Accuracy = (tp + tn) / (tp + tn + fp + fn)
print("Accuracy {:0.2f}".format(Accuracy))
```

```
Accuracy 0.76
```

In [21]:
```
Specificity = tn/(tn+fp)
print("Specificity {:0.2f}".format(Specificity))
```

```
Specificity 0.94
```

In [22]:
```
Sensitivity = tp/(tp+fn)
print("Sensitivity {:0.2f}".format(Sensitivity))
```

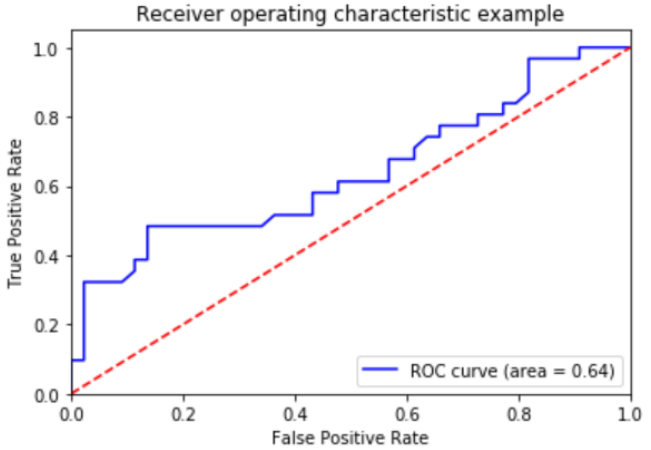
```
Sensitivity 0.44
```

$t = 0.5$

ROC Curve for Training dataset

In [23]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc
log_ROC_AUC1 = roc_auc_score(y_train, y_predict_train)
fpr1, tpr1, thresholds1= roc_curve(y_train, y_prob_train)
roc_auc1 = auc(fpr1, tpr1)
```

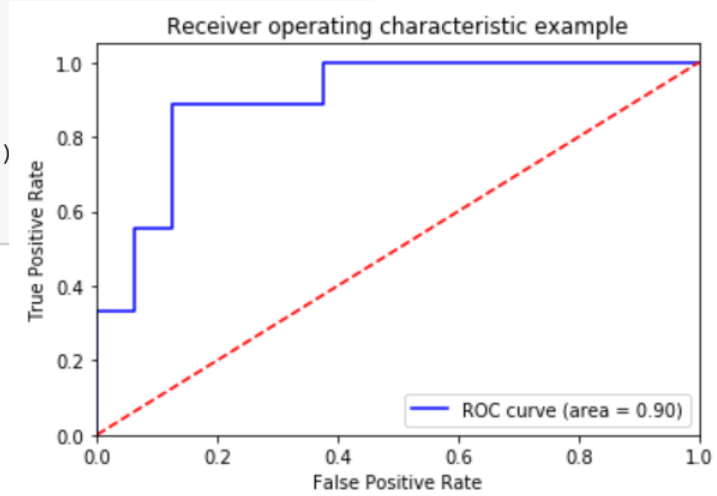
In [24]:
```
plt.figure()
plt.plot(fpr1, tpr1, color='blue', label='ROC curve (area = %0.2f)' % roc_auc1)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



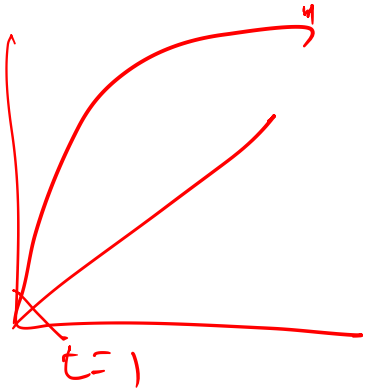
ROC Curve for Test data set

In [25]:
```
log_ROC_AUC = roc_auc_score(y_test, y_predict)
fpr, tpr, thresholds= roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
```

In [26]:
```
plt.figure()
plt.plot(fpr, tpr, color='blue', label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1],'r--')
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 example')
plt.legend(loc="lower right")
plt.show()
```



Threshold value selection



- The outcome of logistic regression model is a probability.
- Selecting a good threshold value is often challenging.
- Threshold values on ROC curve –

Threshold = 1	TPR = 0	FPR = 0
Threshold = 0	TPR = 1	FPR = 1

- Threshold values are often selected based on which errors are bettor.

Accuracy checking for different threshold values

In [27]:
```
from sklearn.preprocessing import binarize
y_predict_class1 = binarize(y_prob.reshape(1,-1), 0.35)[0]
y_predict_class1
```

Out[27]:
```
array([1., 1., 1., 0., 0., 0., 0., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1.,
       1., 0., 1., 1., 1., 1., 1., 0.])
```

In [28]:
```
#converting the array from float data type to integer data type
y_predict_class1 = y_predict_class1.astype(int)
y_predict_class1
```

Out[28]:
```
array([1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 0])
```

In [29]:
```
confusion_matrix_2 = confusion_matrix(y_test, y_predict_class1)
print(confusion_matrix_2)
```

```
[[8 8]
 [0 9]]
```

Accuracy checking for different threshold values

In [30]:

```
tn, fp, fn, tp = confusion_matrix(y_test, y_predict_class1).ravel()
print("True Negatives: ",tn)
print("False Positives: ",fp)
print("False Negatives: ",fn)
print("True Positives: ",tp)
```

```
True Negatives:  8
False Positives:  8
False Negatives:  0
True Positives:  9
```

In [31]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict_class1))
```

	precision	recall	f1-score	support
0	1.00	0.50	0.67	16
1	0.53	1.00	0.69	9
micro avg	0.68	0.68	0.68	25
macro avg	0.76	0.75	0.68	25
weighted avg	0.83	0.68	0.68	25

Accuracy checking for different threshold values

In [32]:
```
from sklearn.preprocessing import binarize
y_predict_class2 = binarize(y_prob.reshape(1,-1), 0.50)[0]
y_predict_class2
```

Out[32]:
```
array([1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       1., 0., 0., 0., 1., 1., 0., 0.])
```

In [33]:
```
confusion_matrix_3 = confusion_matrix(y_test, y_predict_class2)
print(confusion_matrix_3)
```

```
[[15  1]
 [ 5  4]]
```

In [34]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict_class2))
```

```
              precision    recall  f1-score   support

           0       0.75      0.94      0.83        16
           1       0.80      0.44      0.57         9

   micro avg       0.76      0.76      0.76        25
   macro avg       0.78      0.69      0.70        25
weighted avg       0.77      0.76      0.74        25
```

Accuracy checking for different threshold values

In [35]:
```
from sklearn.preprocessing import binarize
y_predict_class3 = binarize(y_prob.reshape(1,-1), 0.70)[0]
y_predict_class3
```

Out[35]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [36]:
```
confusion_matrix_4 = confusion_matrix(y_test, y_predict_class3)
print(confusion_matrix_4)
```

```
[[16  0]
 [ 9  0]]
```

In [37]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict_class3))
```

```
              precision    recall  f1-score   support

           0       0.64      1.00      0.78        16
           1       0.00      0.00      0.00         9

   micro avg       0.64      0.64      0.64        25
   macro avg       0.32      0.50      0.39        25
weighted avg       0.41      0.64      0.50        25
```

Calculating Optimal Threshold Value

In [38]:
```
from sklearn.metrics import roc_curve, auc
```

In [39]:
```
fpr, tpr, thresholds= roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
```

In [40]:
```
print("Area under the ROC curve : %f" % roc_auc)
```

In [41]:
```
import numpy as np
i = np.arange(len(tpr)) # index for df
roc = pd.DataFrame({'fpr' : pd.Series(fpr, index=i),'tpr' : pd.Series(tpr, index = i),
                    '1-fpr' : pd.Series(1-fpr, index = i), 'tf' : pd.Series(tpr - (1-fpr), index = i),
                    'thresholds' : pd.Series(thresholds, index = i)})
roc.iloc[(roc.tf-0).abs().argsort()[:1]]
```

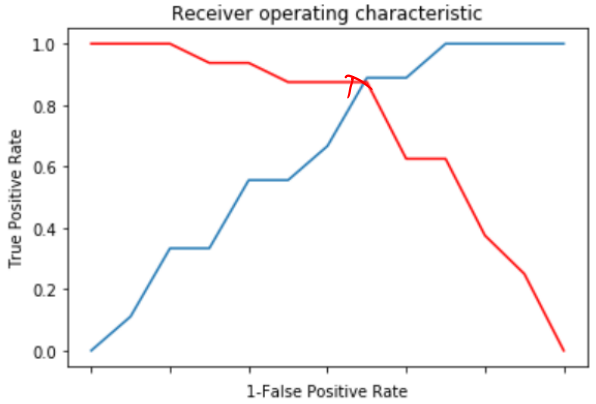
Out[41]:

	fpr	tpr	1-fpr	tf	thresholds
7	0.125	0.888889	0.875	0.013889	0.457033

Optimal Threshold Value in ROC Curve

In [42]:
```
1  fig, ax = plt.subplots()
2  plt.plot(roc['tpr'])
3  plt.plot(roc['1-fpr'], color = 'red')
4  plt.xlabel('1-False Positive Rate')
5  plt.ylabel('True Positive Rate')
6  plt.title('Receiver operating characteristic')
7  ax.set_xticklabels([])
```

Out[42]: []



Classification Report using Optimal Threshold Value

In [43]:
```
from sklearn.preprocessing import binarize
y_predict_class4 = binarize(y_prob.reshape(1,-1), 0.45)[0]
y_predict_class4
```

Out[43]:
```
array([1., 1., 1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 1.,
       1., 0., 1., 0., 1., 1., 0., 0.])
```

In [44]:
```
confusion_matrix_5 = confusion_matrix(y_test, y_predict_class4)
print(confusion_matrix_5)
```

```
[[14  2]
 [ 1  8]]
```

In [45]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict_class4))
```

```
              precision    recall  f1-score   support

           0       0.93      0.88      0.90        16
           1       0.80      0.89      0.84         9

   micro avg       0.88      0.88      0.88        25
   macro avg       0.87      0.88      0.87        25
weighted avg       0.89      0.88      0.88        25
```

Thank You