Kmit

SONET

# Data Science

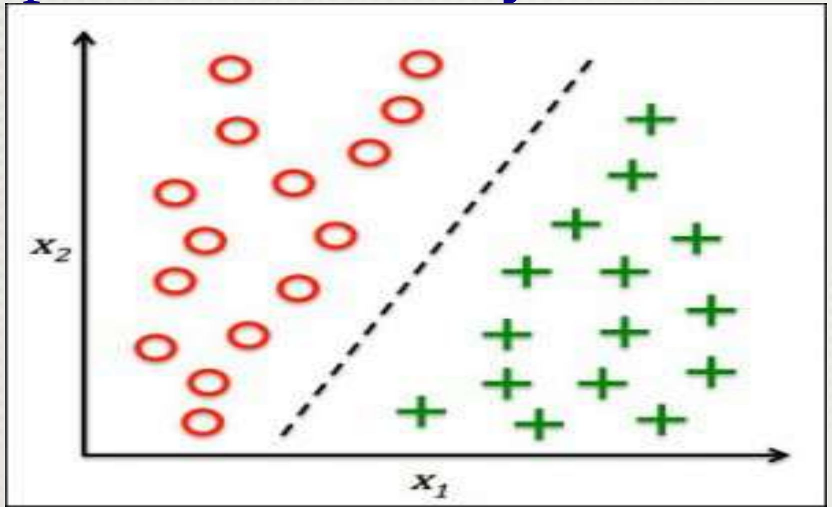
(Level-I)

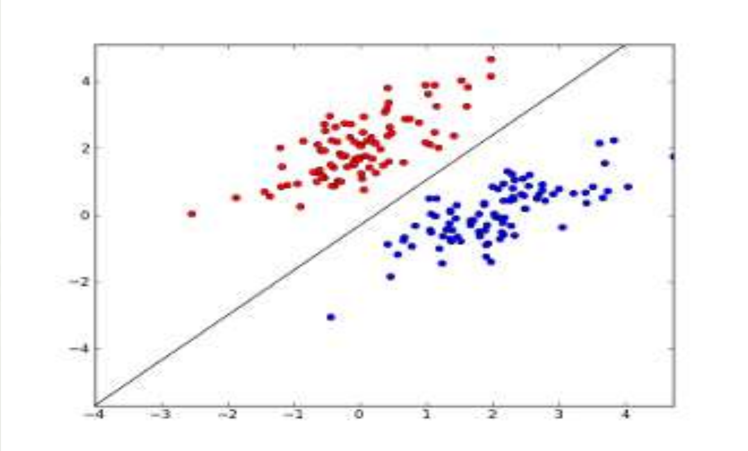

# Classification Metrics

# Introduction

- A classification problem is when the output variable is a categorical or class value, such as spam/non-spam or fraud/non-fraud.
- Many different models can be used, the simplest is the logistic regression, decision tree etc...
- The decision being modeled is to assign labels to new unlabelled pieces of data.
- Classification techniques predict discrete responses

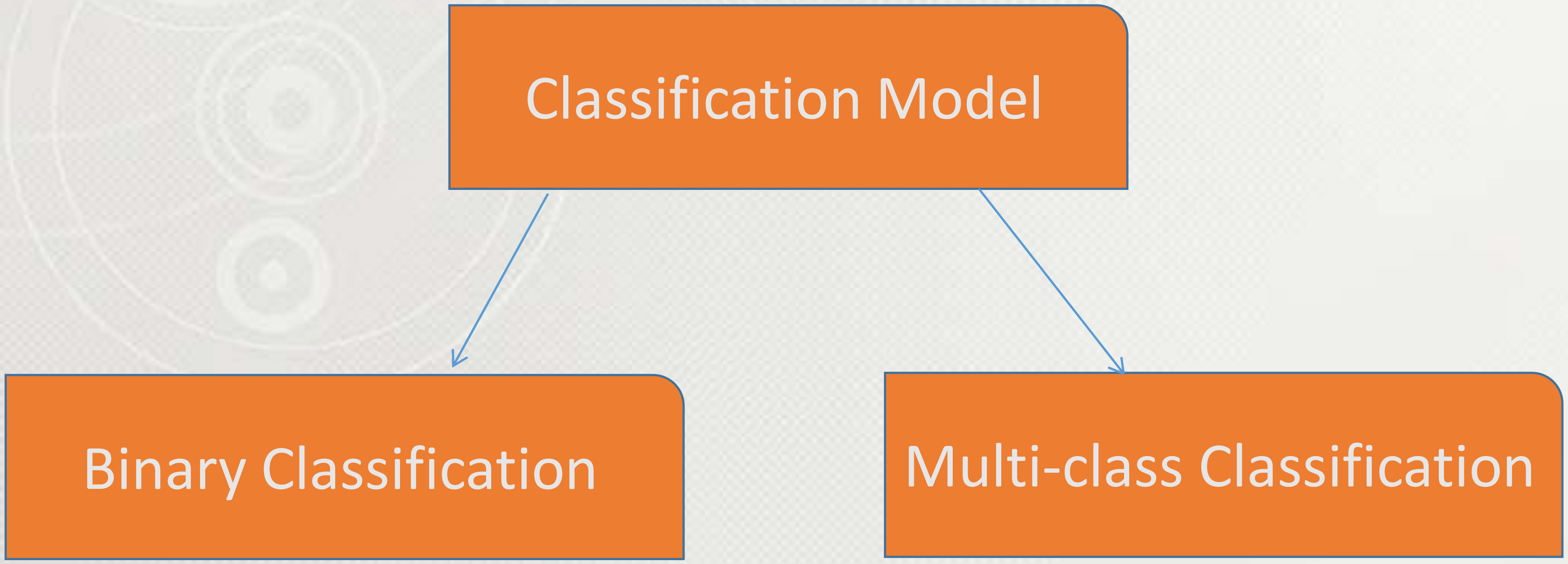
- Examples:
- Classify a machine learning program that will be able to detect cancerous tumors in lungs
- Classify there are linguistics researchers studying grammar structures in languages
- Classifying the new music to a user based on their music preferences
- Classify a computer processor factory



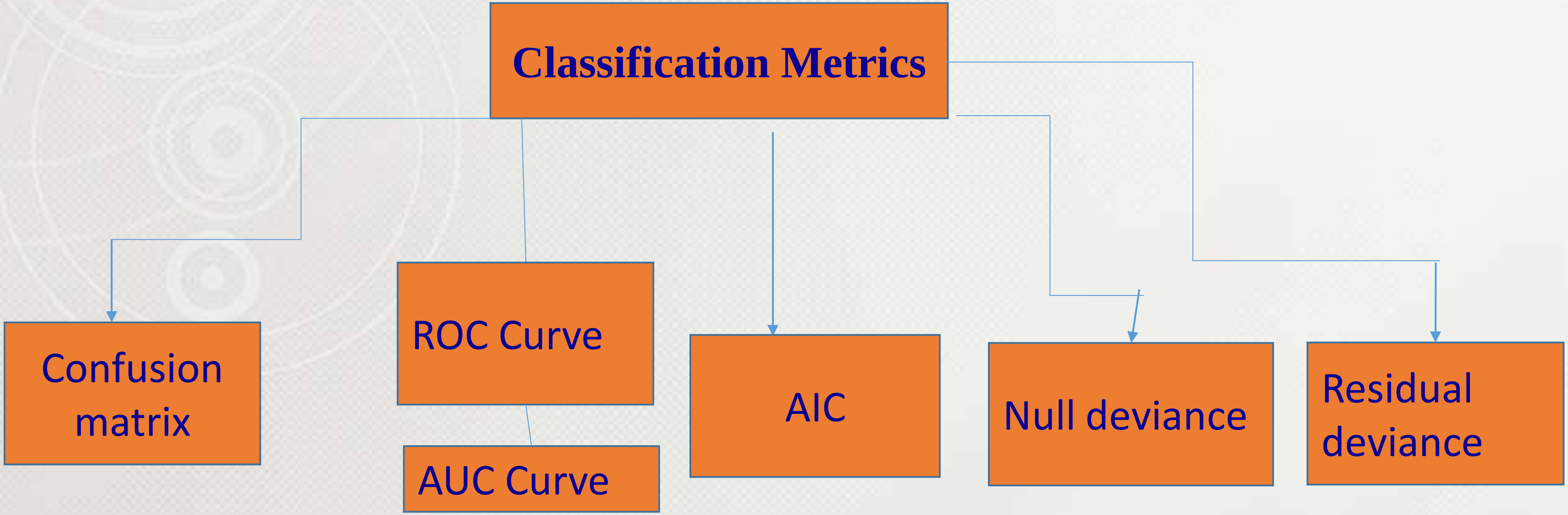



# Introduction

**Types Of Classification Models:**



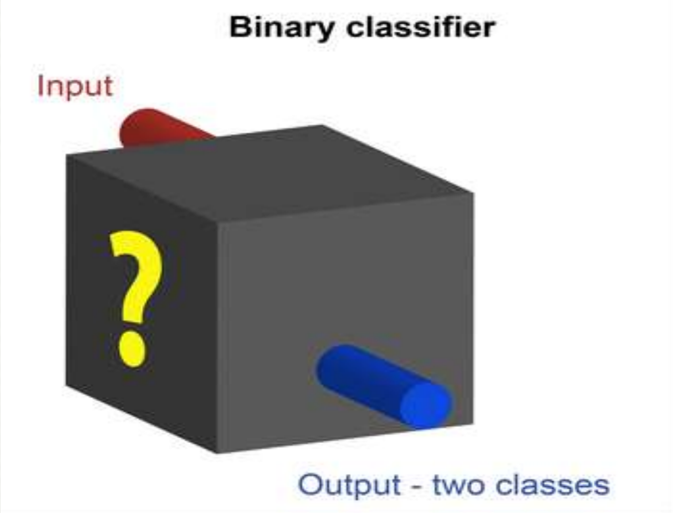
# Introduction




ROC: receiver operating characteristic curve
AIC: Akaike’s information criterion
AUC: *Area Under the ROC Curve*

# Confusion matrix

- **Basic evaluation measures from the confusion matrix**
- The confusion matrix is a two by two table that contains four outcomes produced by a binary classifier.
- ***Test datasets for binary classifier***
- A binary classifier produces output with two class values or labels, such as Yes/No and 1/0, for given input data. The class of interest is usually denoted as “positive” and the other as “negative”.

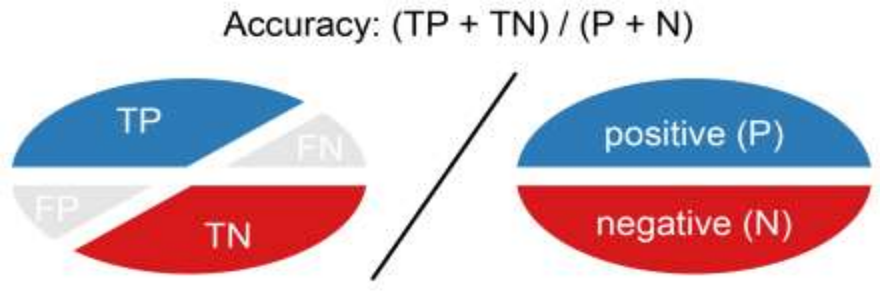


| | | Predicted | |
|---|---|---|---|
| | | Good | Bad |
| Actual | Good | True Positive (d) | False Negative (c) |
| | Bad | False Positive (b) | True Negative (a) |

- **Measures**
- ***Basic measures derived from the confusion matrix***
- Various measures can be derived from a confusion matrix.
- **basic measures from the confusion matrix**
- Accuracy (ACC) and Error rate (ERR) are the most common and intuitive measures derived from the confusion matrix.
- Accuracy = (TP + TN) / (TP + TN + FP +FN)

# Accuracy

- Accuracy (ACC) is calculated as the number of all correct predictions divided by the total number of the dataset.
- whereas the worst is 0.0. It can also be calculated by 1 – ERR.




- total number of two correct predictions (TP + TN) divided by the total number of a dataset (P + N).

$$ACC = \frac{TP + TN}{TP + TN + FN + FP} = \frac{TP + TN}{P + N}$$

- Recall = TP / (TP + FN)
- recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances.
- Precision = TP / (TP + FP)
- **precision** (also called positive predictive value) is the fraction of relevant instances among the retrieved instances,
- F1 Score = 2(Precision * Recall) / (Precision + Recall) [1-Best, 0-Worst]

- Examples:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
expected=["woman","man","man","woman"]
pred=["man","woman","woman","man"]

matrix=confusion_matrix(pred, expected)
print("Matrix")
print(matrix)
acc=accuracy_score(pred, expected)
print("Accuracy")
print(acc)
print("Classification")
report=classification_report(pred, expected)
print(report)
```

# Output:

```
Matrix
[[0 2]
 [2 0]]
Accuracy
0.0
Classification
             precision    recall  f1-score   support

        man       0.00      0.00      0.00         2
      woman       0.00      0.00      0.00         2

avg / total       0.00      0.00      0.00         4
```

```
expected=["woman","man","woman","man"]
pred=["woman","man","man","woman"]
matrix=confusion_matrix(pred, expected)
print("Matrix")
print(matrix)
acc=accuracy_score(pred, expected)
print("Accuracy")
print(acc)
print("Classification")
report=classification_report(pred, expected)
print(report)
```

```
Matrix
[[1 1]
 [1 1]]
Accuracy
0.5
Classification
             precision    recall  f1-score   support

        man       0.50      0.50      0.50         2
      woman       0.50      0.50      0.50         2

avg / total       0.50      0.50      0.50         4
```

```python
expected=["men","men","men","men"]
pred=["men","men","men","men"]
matrix=confusion_matrix(pred, expected)
print("Matrix")
print(matrix)
acc=accuracy_score(pred, expected)
print("Accuracy")
print(acc)
print("Classification")
report=classification_report(pred, expected)
print(report)
```

```
Matrix
[[4]]
Accuracy
1.0
Classification
             precision    recall  f1-score   support

        men       1.00      1.00      1.00         4

avg / total       1.00      1.00      1.00         4
```

SONET

# Titanic data set Example

Machine Learning

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
metrix_confusion=confusion_matrix(y_pred,y_test)
acc=accuracy_score(y_pred,y_test)
print(metrix_confusion)
print("accuracy:",acc)
report=classification_report(y_pred,y_test)
print("report:",report)
```

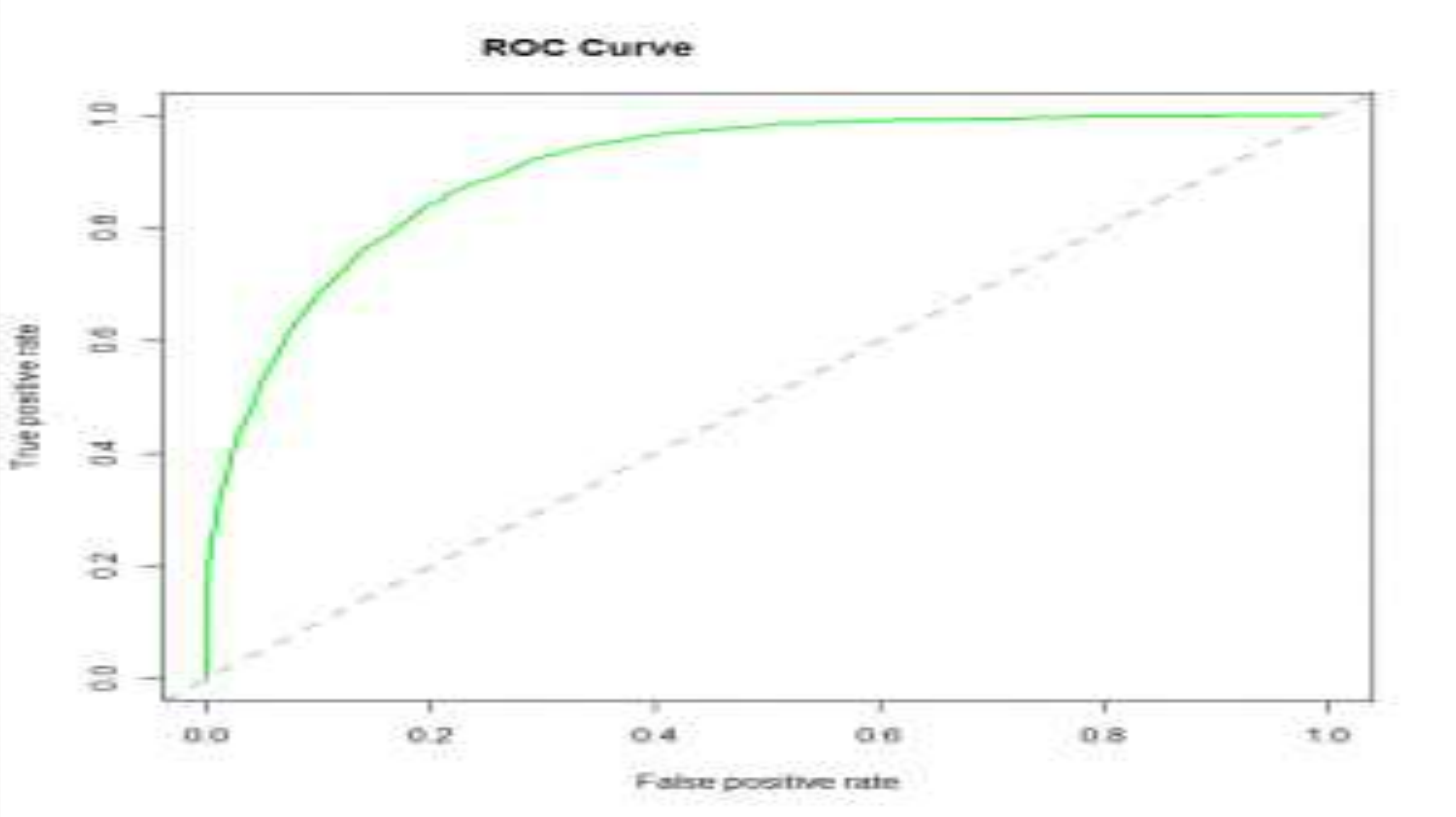
```
[[101  25]
 [ 24  64]]
accuracy: 0.7710280373831776
report:              precision    recall  f1-score   support

          0       0.81      0.80      0.80       126
          1       0.72      0.73      0.72        88

avg / total       0.77      0.77      0.77       214
```

# ROC Curve

- An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds.
- This curve plots two parameters:
- True Positive Rate
- False Positive Rate
- **True Positive Rate** (**TPR**) is a synonym for recall and is therefore defined as follows:
- TPR=TP/TP+FN
- **False Positive Rate** (**FPR**) is defined as follows:
- FPR=FP/FP+TN


ROC Curve
True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate

- *AUC: Area Under the ROC Curve*
- **AUC** stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).

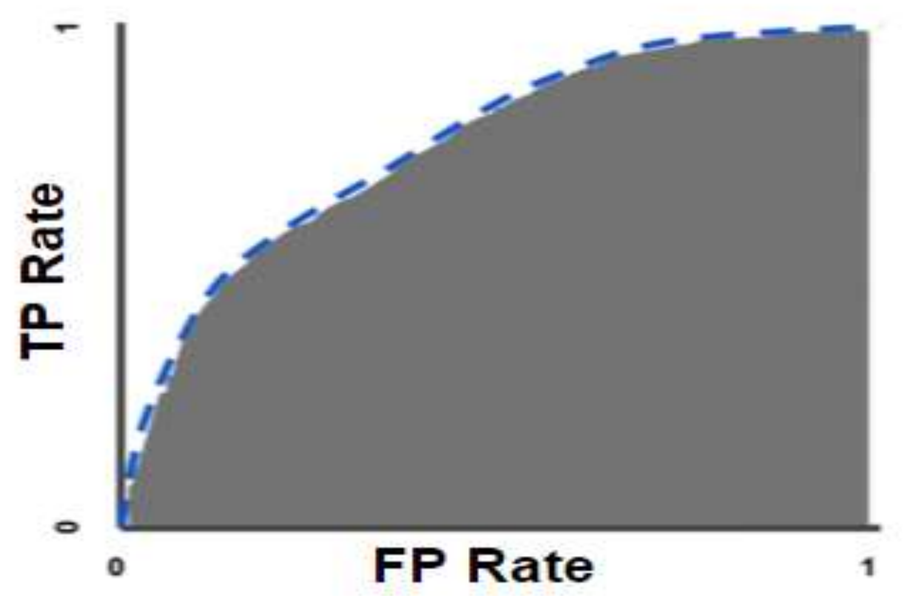
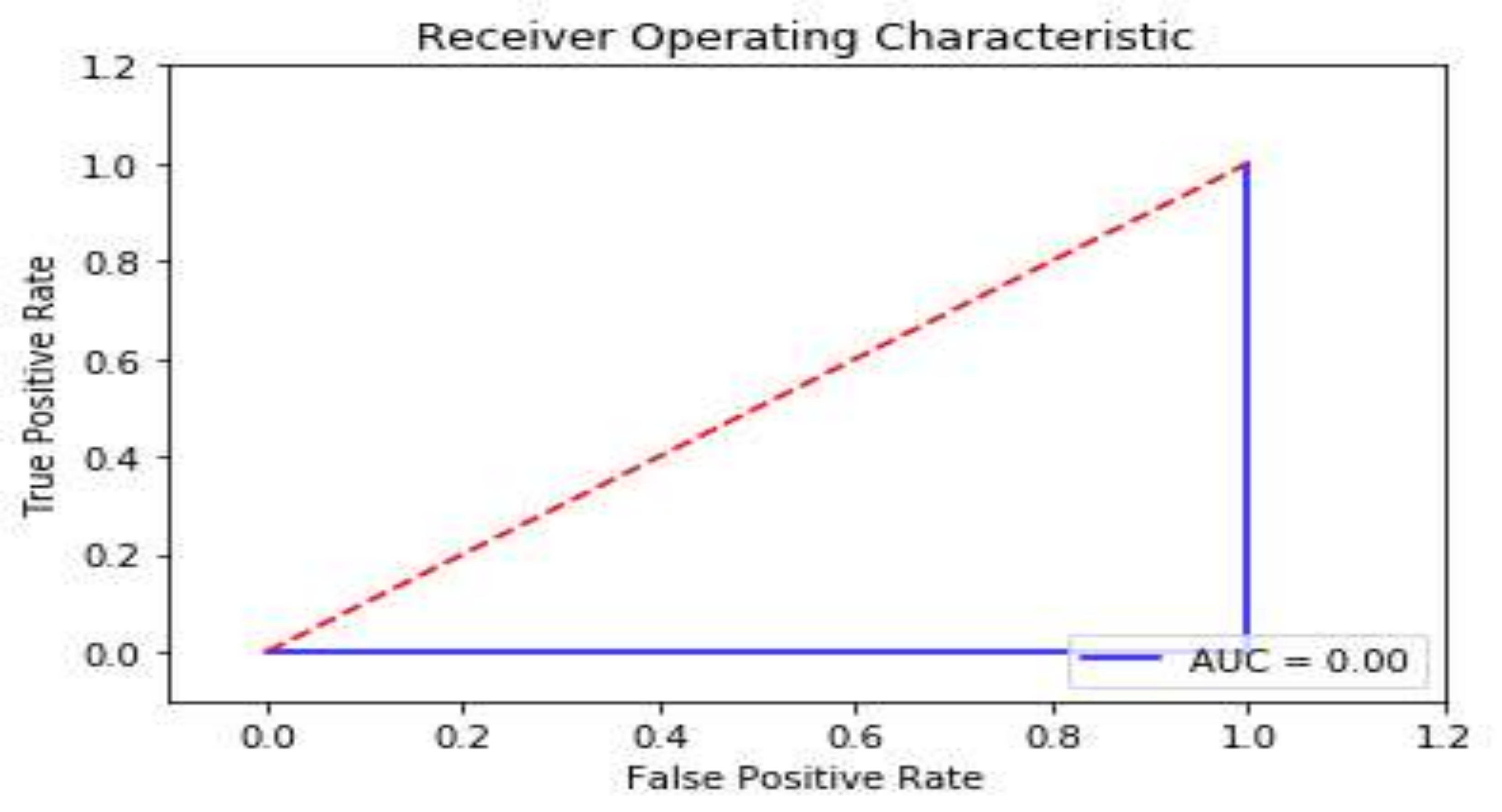



Figure 5. AUC (Area under the ROC Curve).

- AUC represents the probability that a random positive (1) example is positioned to the right of a random negative (0) example.
- AUC ranges in value from 0 to 1.
- For plotting ROC, it is advisable to assume $p > 0.5$ since we are more concerned about success rate.

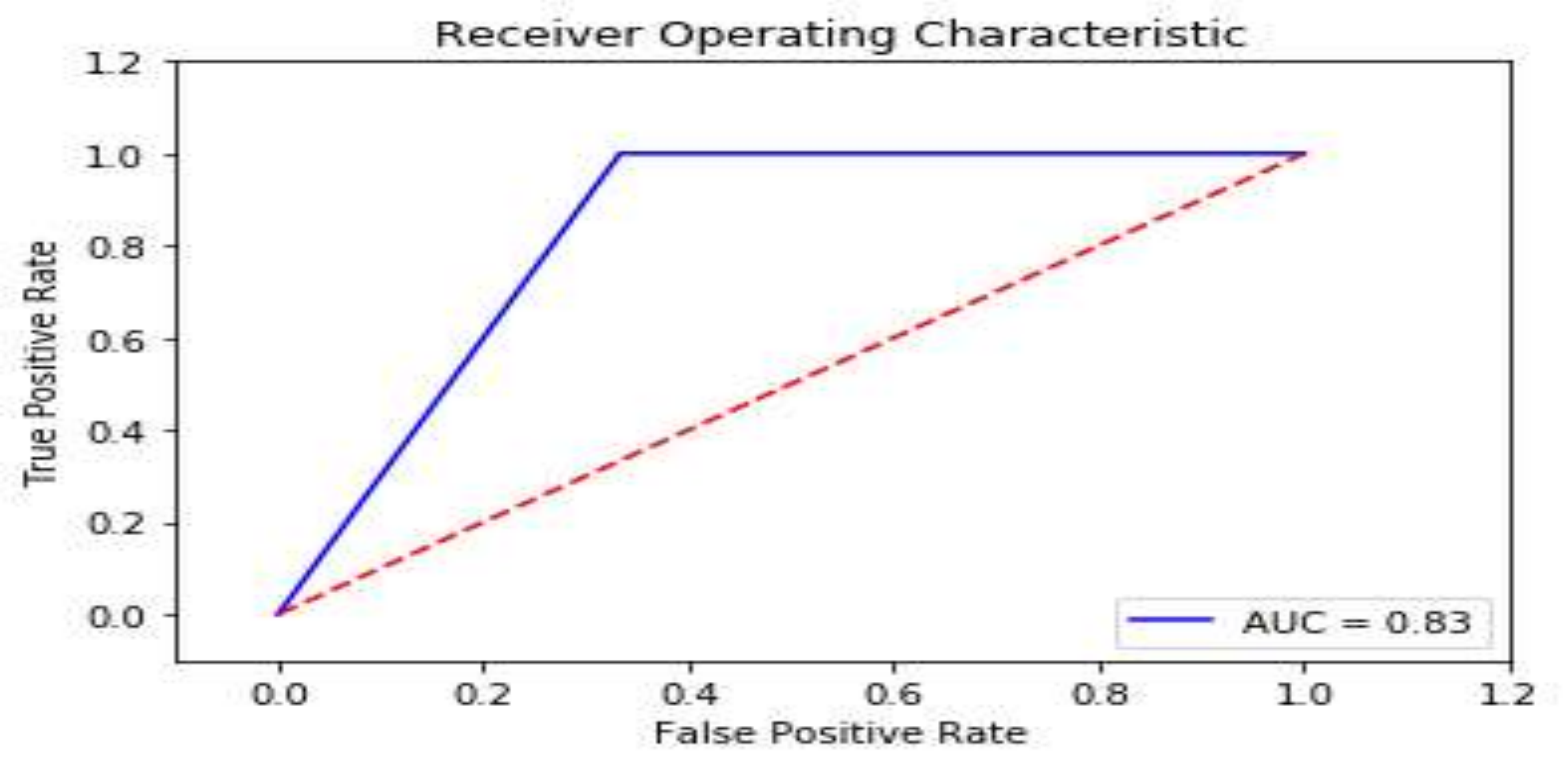
- Example:

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
import random
actual = [1,1,1,0,0,0]
predictions = [0,0,0,1,1,1]
false_positive_rate, true_positive_rate, thresholds = roc_curve(actual, predictions)
roc_auc = auc(false_positive_rate, true_positive_rate)
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',
label='AUC = %0.2f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

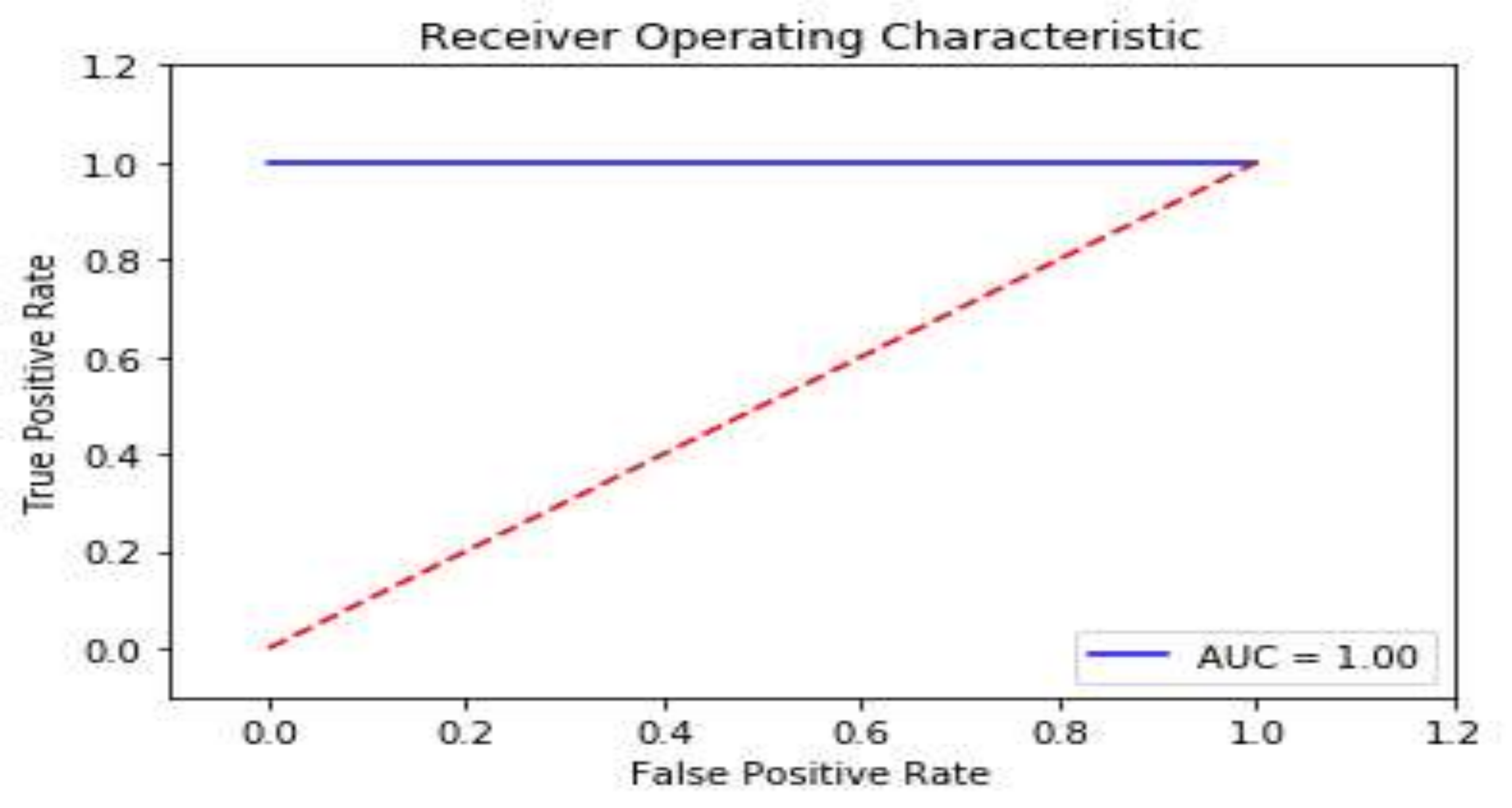

Receiver Operating Characteristic
True Positive Rate
False Positive Rate
AUC = 0.00
0.0
0.2
0.4
0.6
0.8
1.0
1.2

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
import random
actual = [1,1,1,0,0,0]
predictions = [1,1,1,0,0,1]

false_positive_rate, true_positive_rate, thresholds = roc_curve(actual, predictions)
roc_auc = auc(false_positive_rate, true_positive_rate)
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',
label='AUC = %0.2f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

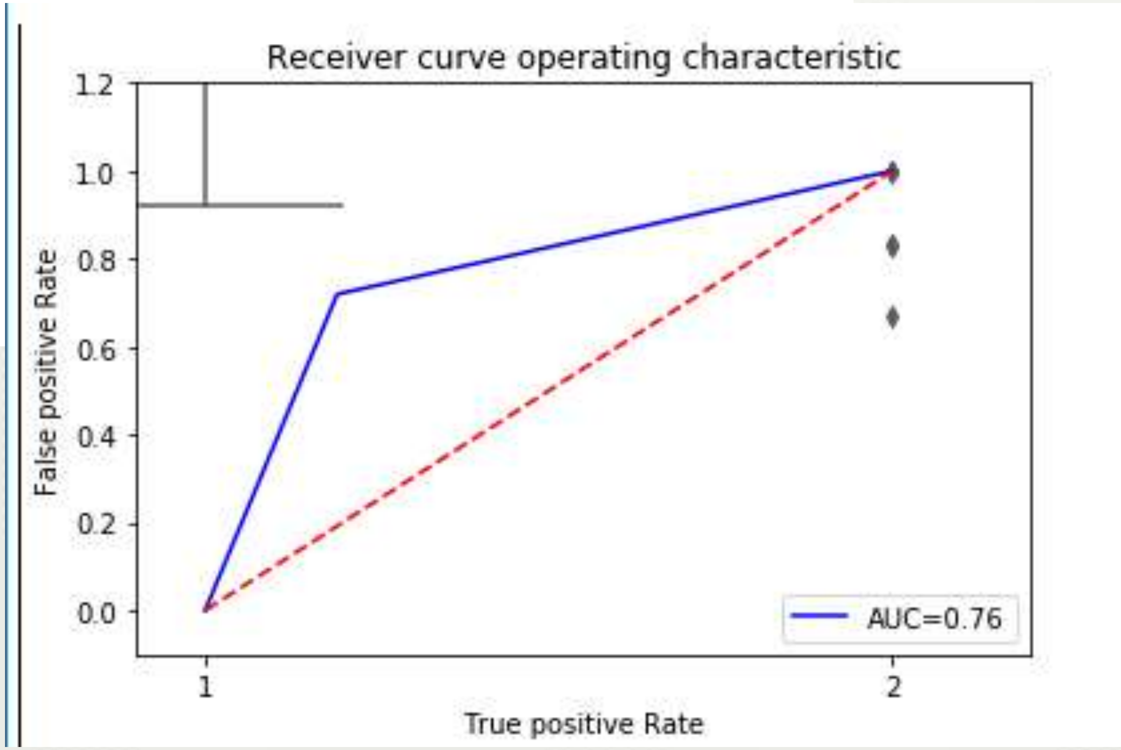

Receiver Operating Characteristic
True Positive Rate
False Positive Rate
AUC = 0.83
0.0
0.2
0.4
0.6
0.8
1.0
1.2

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
import random
actual = [1,1,1,0,0,0]
predictions = [1,1,1,0,0,0]
false_positive_rate, true_positive_rate, thresholds = roc_curve(actual, predictions)
roc_auc = auc(false_positive_rate, true_positive_rate)
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',
label='AUC = %0.2f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```


Receiver Operating Characteristic
1.2
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
1.2
False Positive Rate
AUC = 1.00

# Example Titanic data set

```
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds=roc_curve(y_test,y_pred)
roc_auc=auc(fpr,tpr)
plt.title("Receiver curve operating characteristic")
plt.plot(fpr,tpr,'b',label='AUC=%0.2f'%roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.xlabel("True positive Rate")
plt.ylabel("False positive Rate")
plt.show()
```



# AIC (Akaike Information Criteria):

- The analogous metric of adjusted $R^2$ in logistic regression is AIC.
- AIC is the measure of fit which penalizes model for the number of model coefficients.
- Therefore, we always prefer model with minimum AIC value.

$$AIC = n \log(\hat{\sigma^2}) + 2K$$

Where:

- $\hat{\sigma^2}$ = Residual Sum of Squares/n,
- n = sample size,
- K is the number of model parameters.

# Example

```
#AIC Calculation
import math
residual=0
for k in range(len(y_test)):
    residual =residual+((y_test[k] -y_pred[k]) ** 2)/len(y_test)
print(residual)
aic=len(y_test)*math.log(residual)+4
print(aic)
```

```
[0.22897196]
-311.46932341900214
```

1 - p

SONET

# Null Deviance and Residual Deviance:

Machine Learning

- Null Deviance indicates the response predicted by a model with nothing but an intercept.
- Lower the value, better the model.
- Residual deviance indicates the response predicted by a model on adding independent variables. Lower the value, better the model.
- data points with **p parameters**+ an intercept term, so you have p+1 parameters.
- If your Null Deviance is really small, it means that the Null Model explains the data pretty well.

# THANK YOU