Classification Evaluation Metrics:

Accuracy = (TP+TN)/(TP+FP+FN+TN)

Precision = (TP)/(TP+FP)

Recall = (TP)/(TP+FN)

F1 Score: Harmonic mean of precision and recall

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

```
from sklearn.metrics import f1_score
y_true = [0, 1, 1, 0, 1, 1]
y_pred = [0, 0, 1, 0, 0, 1]


f1_score(y_true, y_pred)
```

F Score with Beta value (F Beta Score):

$$F_\beta = (1 + \beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{(\beta^2 \cdot \mathrm{precision}) + \mathrm{recall}}.$$

```
from sklearn.metrics import fbeta_score


y_true = [0, 1, 1, 0, 1, 1]
y_pred = [0, 0, 1, 0, 0, 1]


fbeta_score(y_true, y_pred,beta=0.5)
```

3. Log Loss/Binary Crossentropy

Log loss is a pretty good evaluation metric for binary classifiers and it is sometimes the optimization objective as well in case of Logistic regression and Neural Networks.

Binary Log loss for an example is given by the below formula where p is the probability of predicting 1.

$$-(y\log(p) + (1-y)\log(1-p))$$

How to Use?

```
from sklearn.metrics import log_loss

# where y_pred are probabilities and y_true are binary class labels
log_loss(y_true, y_pred, eps=1e-15)
```

4. Categorical Crossentropy

The log loss also generalizes to the multiclass problem. The classifier in a multiclass setting must assign a probability to each class for all examples. If there are N samples belonging to M classes, then the *Categorical Crossentropy* is the summation of - `ylogp` values:

$$LogarithmicLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log(p_{ij})$$

`y_ij` is 1 if the sample `i` belongs to class `j` else 0

`p_ij` is the probability our classifier predicts of sample `i` belonging to class `j`.

```
from sklearn.metrics import log_loss

# Where y_pred is a matrix of probabilities with shape = (n_samples,
n_classes) and y_true is an array of class labels

log_loss(y_true, y_pred, eps=1e-15)
```

AUC

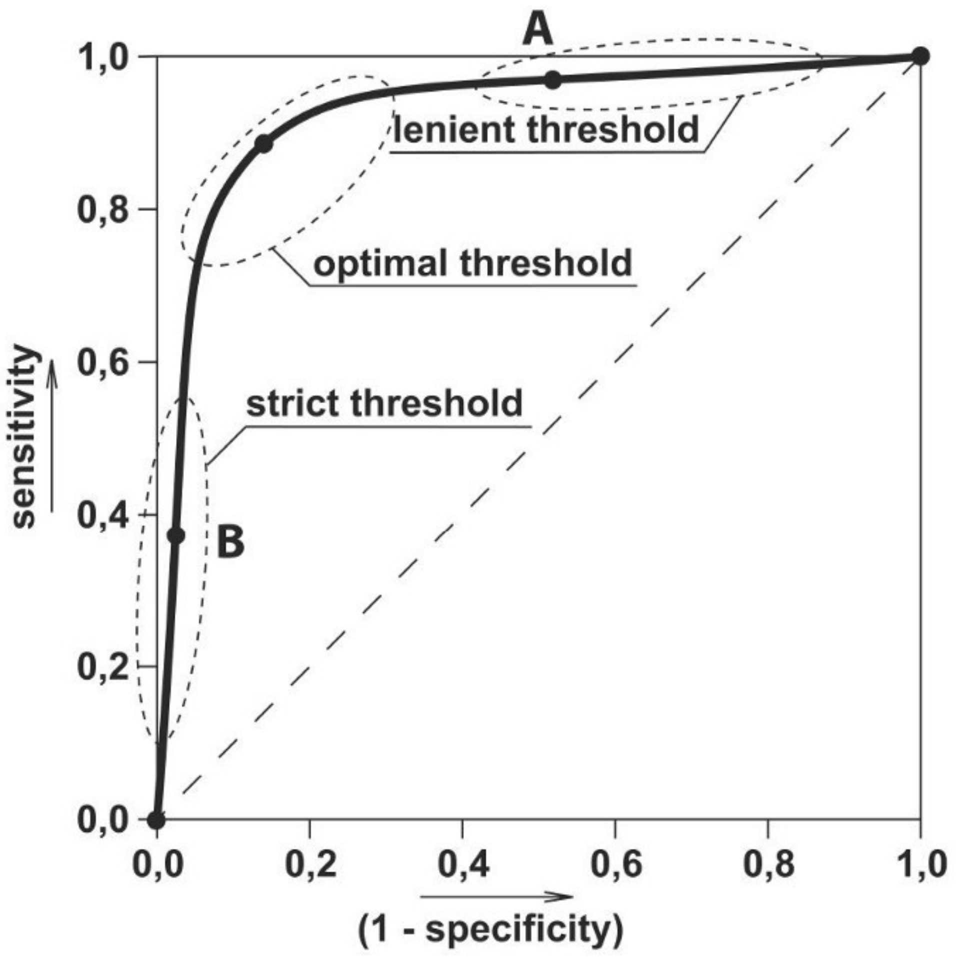
AUC is the area under the ROC curve.

AUC ROC indicates how well the probabilities from the positive classes are separated from the negative classes

What is the ROC curve?

Sensitivty = TPR(True Positive Rate)= Recall = TP/(TP+FN)

1- Specificity = FPR(False Positive Rate)= FP/(TN+FP)



```
import numpy as np
from sklearn.metrics import roc_auc_score
y_true = np.array([0, 0, 1, 1])
y_scores = np.array([0.1, 0.4, 0.35, 0.8])


print(roc_auc_score(y_true, y_scores))
```