Sane's STATS Academy of Statistics

Model Evaluation

Types of Predicted Values

- **Categorical** like Yes/No , Purchased/Not Purchased and also other type of categorical values, not necessarily only binary. We use Classification Confusion Matrix for evaluation
- **Numeric** like Sales, Cost, Profit, Scores
  - We use for evaluation Mean Absolute Error, Mean Squared Error, R^2

Categorical: Example

- Suppose that we have predicted a categorical variable named defaulter which has values as Y (Defaulter) and N (Not a Defaulter) on the validation dataset using a model built on training dataset
- Here, we term defaulter(Y) as positive class and non-defaulter(N) as negative class
- Say, the validation set has got some 14 values as

```
In [54]: predicted
Out[54]:
array(['Y', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'N', 'N', 'Y',
       'N'], dtype=object)
```

Diagnosis

- In the following cases, we won't have errors:
  - We predict a defaulter as defaulter
  - We predict a non-defaulter as non-defaulter
- In the following cases we have errors:
  - We predict a defaulter as non-defaulter
  - We predict a non-defaulter as defaulter

Indicators Tabulated

	Predicted as Defaulter	Predicted as Non-Defaulter
Actually a Defaulter (+ve class)	True +ve	False –ve
Actually a Non-Defaulter(-ve class)	False +ve	True -ve

The Matrix shown above is called **Classification Confusion Matrix**

Basic quantitative quality indicators

- TP – True Positive : Correctly assigned observations to the positive class.
- TN – True Negative : Correctly assigned observations to the negative class.
- FP – False Positive : Wrongly assigned observations to the positive class. (Which actually belong to the negative class)
- FN – False Negative : Wrongly assigned observations to the negative class. (Which actually belong to the positive class)

Classification Confusion Matrix

Recall(Sensitivity) = TP / (TP + FN)

False Positive Rate = FP / (TN + FP)

Precision= TP / (TP + FP)

		Predicted as Defaulter	Predicted as Non-Defaulter
Actually a Defaulter (+ve class)		TP (Defaulter diagnosed as Defaulter)	FN (Defaulter diagnosed as Non-Defaulter)
Actually a Non-Defaulter(-ve class)		FP (Non-Defaulter diagnosed as Defaulter)	TN (Non-Defaulter diagnosed as Non-Defaulter)

False Negative Rate = FN / (TP + FN)

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Specificity= TN / (TN + FP)

Overall Prediction Correctness

ACC (Total Accuracy)

= P(correct prediction)

= number of correct decision/ total number of decisions

$$ACC = (TP + TN) / (TP + TN + FP + FN)$$

		Predicted as Defaulter	Predicted as Non-Defaulter
Actually a Defaulter	**(+ve class)**	TP (Defaulter diagnosed as Defaulter)	FN (Defaulter diagnosed as Non-Defaulter)
Actually a Non-Defaulter	**(-ve class)**	FP (Non-Defaulter diagnosed as Defaulter)	TN (Non-Defaulter diagnosed as Non-Defaulter)

Example

	Predicted Y	Predicted N	Total
Existing Y	5 (TP)	2 (FN)	7
Existing N	3 (FP)	4 (TN)	7
Total	8	6	14

$$Accuracy = \frac{(5+4)}{(5+2+3+4)} = 0.692308$$

$$Recall = \frac{TP}{TP+FN} = \frac{5}{5+2} = 0.714286$$

$$Precision(N) = \frac{TP}{TP+FP} = \frac{5}{5+3} = 0.625$$

$$F1\ Score = 2 * \frac{0.714286 * 0.625}{0.714286 + 0.625} = 0.67$$

Receiver Operating Characteristic Curve

ROC Curve

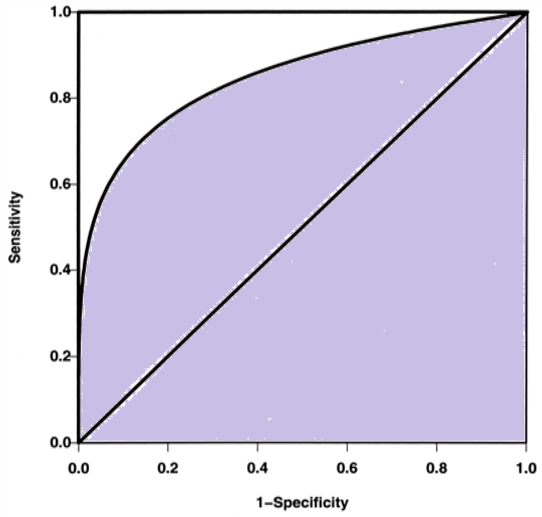
What is ROC curve?

- Receiver operating characteristic (ROC), or ROC curve, is a graphical plot that illustrates the performance of a binary classifier algorithm.
- The curve is created by plotting the Sensitivity(Y axis) or true positive rate (TPR) against the (1 – Specificity) (X axis) or false positive rate (FPR) at various threshold settings.

ROC Curve

- The area is measured of lower right portion of the curve.
- That area is termed as AUC or area under the curve
- The area to be considered has been indicated by the coloured portion
- Bigger the AUC better is the model

0 <= AUC <= 1
AUC = 0.5 for Random Guessing
= 1 for perfect classification
Usually,
AUC > 0.8 is considered as good



Sane's STATS Academy of Statistics

Drawbacks of ROC-AUC Metric

- Can only be used for binary classification
- Not a good metric in case of unbalanced classification problems

From where did the ROC come from?

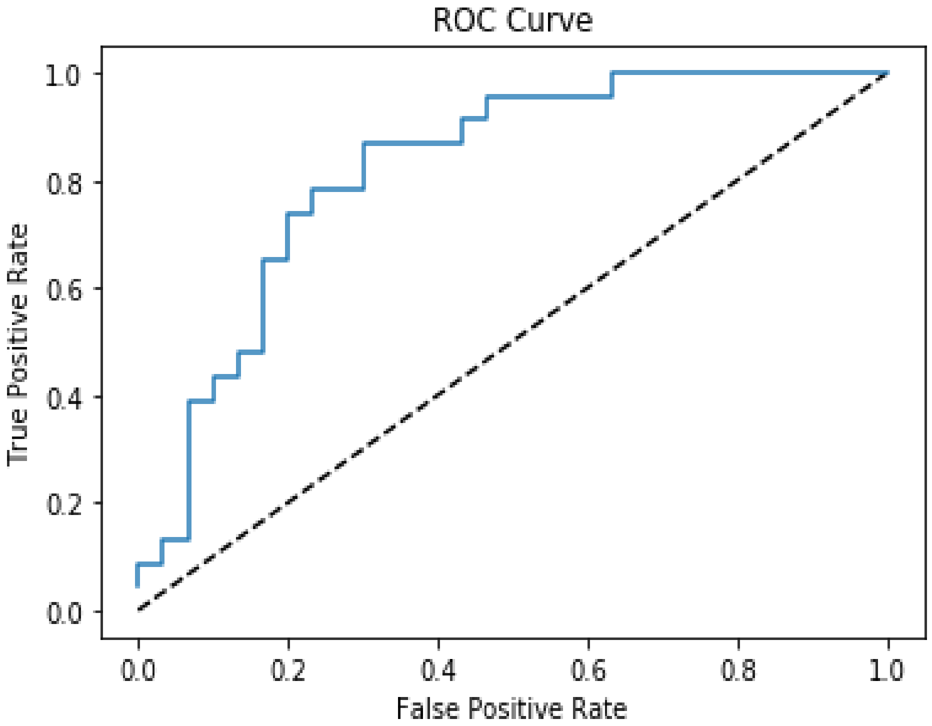
- The ROC curve was first developed by electrical engineers and radar engineers during World War II for detecting enemy objects in battlefields.
- They were building the "Chain Home" series of radar detectors to identify incoming German planes. But the radar detectors would also detect flocks of birds and other "false positive" signals.

Origin of ROC

- The term “receiver operating characteristic” came from tests of the ability of World War II radar operators to determine whether a blip on the radar screen represented an object (signal) or noise.
- The science of “signal detection theory” was later applied to diagnostic medicine and later in the other branches of research and analysis.

ROC in Python

```
In [57]: from sklearn.metrics import roc_curve, roc_auc_score
    ...:
    ...: # Compute predicted probabilities: y_pred_prob
    ...: y_pred_prob = logreg.predict_proba(X_test)[:,1]
    ...:
    ...: # Generate ROC curve values: fpr, tpr, thresholds
    ...: fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
    ...:
    ...: # Plot ROC curve
    ...: import matplotlib.pyplot as plt
    ...: plt.plot([0, 1], [0, 1], 'k--')
    ...: plt.plot(fpr, tpr)
    ...: plt.xlabel('False Positive Rate')
    ...: plt.ylabel('True Positive Rate')
    ...: plt.title('ROC Curve')
    ...: plt.show()
    ...:
    ...: roc_auc_score(y_test, y_pred_prob)
```




```
Out[57]: 0.8217391304347825
```

Sane's STATS Academy of Statistics

Log Loss / Cross Entropy

- This is the loss function used in (multinomial) logistic regression and extensions of it such as neural networks, defined as the negative log-likelihood of the true labels given a probabilistic classifier's predictions.
- For any given problem, a "closer to zero" log-loss value means better predictions.
- For a single sample with true label yt in {0,1} and estimated probability yp that yt = 1, the log loss is

  -log P(yt|yp) = -(yt log(yp) + (1 - yt) log(1 - yp))

$$-\frac{1}{N}\sum_{i=1}^{N}[y_i \log p_i + (1-y_i)\log(1-p_i)].$$

Sane's STATS Academy of Statistics

Classification Metrics

- Accuracy Score
- Precision
- Recall
- F1-Score
- ROC AUC (require probabilities)
- Log Loss (require probabilities)

For Numeric / Continuous Response

- For numeric or continuous response variables we have following measures:
  - Mean Absolute Error
  - Mean Square Error
  - R^2

Model Evaluation: MAE

- The Mean Absolute Error (or MAE) is the sum of the absolute differences between predictions and actual values.
- Lesser the MAE, better is the model

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \widehat{y_i}|$$

where

y_i = Observed Values
$\widehat{y_i}$ = Predicted Values
n = No. of observations

MAE in Python

```
In [60]: y_pred = np.array([13.4,45.4,89.3,90.4,87.3,45.9,16.5])
    ...: y_true = np.array([12.3,46.4,90,100.4,86.3,46,17])
    ...: from sklearn.metrics import mean_absolute_error
    ...: mean_absolute_error(y_true, y_pred)
Out[60]: 2.0571428571428
```

Sane's STATS Academy of Statistics

Model Evaluation: MSE

- The Mean Squared Error (or MSE) is mean of squared error
- Lesser the MSE, better is the model

$$\text{MSE} = \frac{\sum(y_i - \widehat{y_i})^2}{n}$$

Where

y_i = Observed Values

$\widehat{y_i}$ = Predicted Values

n = No. of observations

MSE in Python

```
In [59]: y_pred = np.array([13.4,45.4,89.3,90.4,87.3,45.9,16.5])
    ...: y_true = np.array([12.3,46.4,90,100.4,86.3,46,17])
    ...: from sklearn.metrics import mean_squared_error
    ...: mean_squared_error(y_true, y_pred)
Out[59]: 14.851428571428572
```

Model Evaluation: R^2

- It is measure of the variation explained by the model.
- Bigger the R^2 better is the model

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where

y_i = Observed Values
$\widehat{y_i}$ = Predicted Values
$\bar{y}$ = Mean of Response Variable Values
n = No. of observations

R^2 in Python

```
In [61]: y_pred = np.array([13.4,45.4,89.3,90.4,87.3,45.9,16.5])
    ...: y_true = np.array([12.3,46.4,90,100.4,86.3,46,17])
    ...: from sklearn.metrics import r2_score
    ...: r2_score(y_true, y_pred)
Out[61]: 0.9864325353663524
```

Questions?