METRICS

Mean squared Error(MSE)

MSE basically measures average squared error of our predictions. For each point, it calculates square difference between the predictions and the target and then average those values.

The higher this value, the worse the model is. It is never negative, since we're squaring the individual prediction-wise errors before summing them, but would be zero for a perfect model.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where y_i is the actual expected output and $\hat{y}_i$ is the model's prediction.

Mean Absolute Error (MAE)

In MAE the error is calculated as an average of absolute differences between the target values and the predictions.

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Root Mean Squared Error (RMSE)

RMSE is just the square root of MSE. The square root is introduced to make scale of the errors to be the same as the scale of targets.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} = \sqrt{\mathrm{MSE}}$$

R squared and Adjusted R squared

$$R^2 = \frac{\text{Variance explained by the model}}{\text{Total variance}}$$

$$R^2_{adj} = 1 - \left[\frac{(1 - R^2)(n-1)}{n-k-1} \right]$$

N - the number of points in your data sample

K - the number of independent regressor i.e. the number of variables in your model

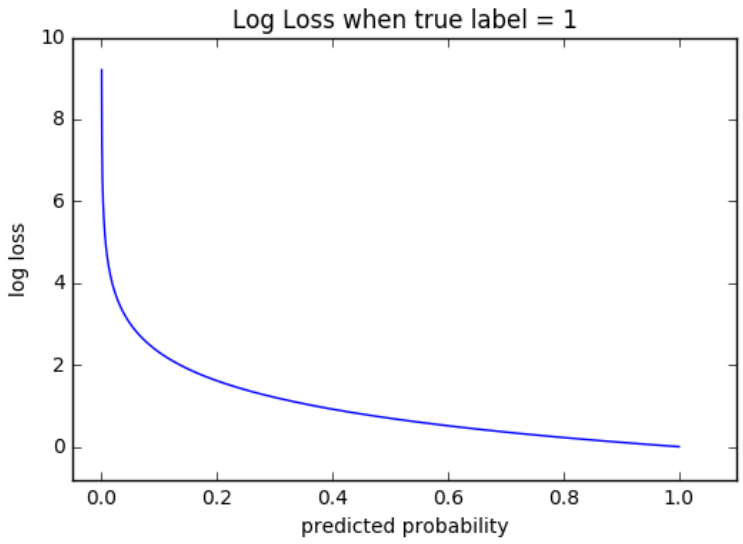
Difference between R squared and Adjusted R squared

R squared assumes that every single variable explains the variation in the dependent variable. The adjusted R squared tells you the percentage of variation explained by only the independent variables that actually affect the dependent variable.

R squared will always increase as you add more features to the model, even if they are unrelated to the response. Every predictor added to a model increases R-squared and never decreases it while the adjusted R-squared compensates for the addition of variables and only increases if the new term enhances the model above what would be obtained by probability and decreases when a predictor enhances the model less than what is predicted by chance. If you add more and more useless variables to a model, adjusted r-squared will decrease. If you add more useful variables, adjusted r-squared will increase. Adjusted r-squared will always be less than or equal to R2.

Cross Entropy

Cross-entropy loss, or log loss measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. A perfect model would have a log loss of 0. Cross-entropy and log loss are slightly different depending on context, but in machine learning when calculating error rates between 0 and 1 they resolve to the same thing.



The graph above shows the range of possible loss values given a true observation. As the predicted probability approaches 1, log loss slowly decreases.

Where does the cross entropy function fit in my deep learning pipeline?

It sits right after the Softmax function and it takes in the input from the Softmax function output and the true label. Remember the goal for cross entropy loss is to compare the how well the probability distribution output by Softmax matches the one-hot-encoded ground truth label of the data.

Binary cross entropy

Remember there can only be two state of the world in binary classification, either y the ground truth is one or zero.

$$J(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\right]$$

If y=0, the first term is zero, so we only calculate (1- y) * log(1-y_hat). (1-y) is 1! If y=1, the second term is zero, only the first term take effect.

Multiclass cross entropy

$$J = -\frac{1}{N}\left(\sum_{i=1}^{N} \mathbf{y_i} \cdot \log(\hat{\mathbf{y}}_\mathbf{i})\right)$$

So if correct = (0, 1, 0) and predicted = (0.1, 0.7, 0.2) then:

loss = - [0 * log (0.1) + 1 * log (0.7) + 0 * log (0.2)]

= - [0 + (1 * -0.36) + 0]

= 0.36

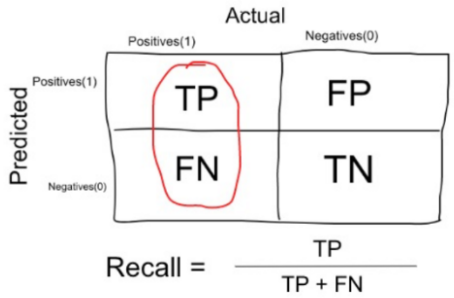
ROC-AUC

One of the problems with the confusion matrix is that all the values that are populated is based on an arbitrary choice of the threshold. AUC - ROC curve is a performance measurement for classification problem at various thresholds settings.

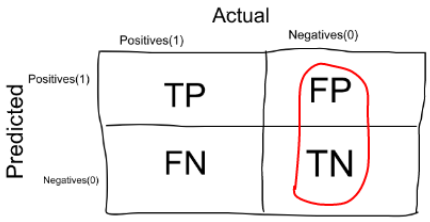
AUC (Area Under the Curve) ROC (Receiver Operating Characteristics) is one of the most important evaluation metrics for checking any classification model's performance. It is also written as AUROC (Area Under the Receiver Operating Characteristics). ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s i.e. model is better at distinguishing.

The ROC curve is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis.

True positive rate = Recall = Sensitivity = true positive / (true positive + false negative)

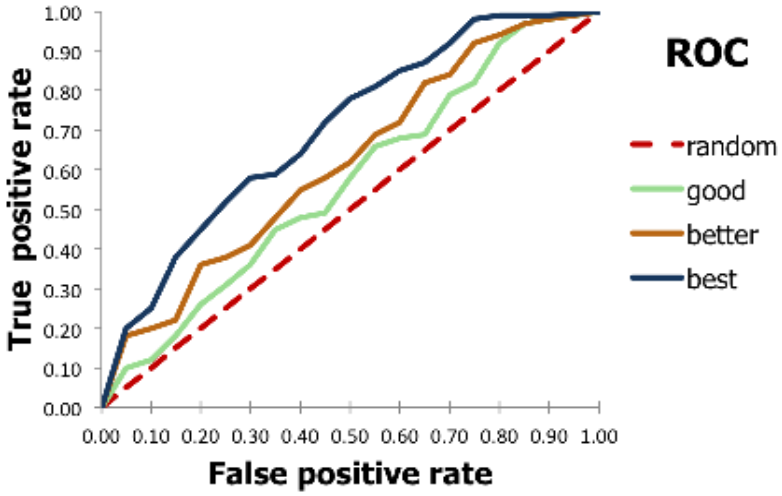


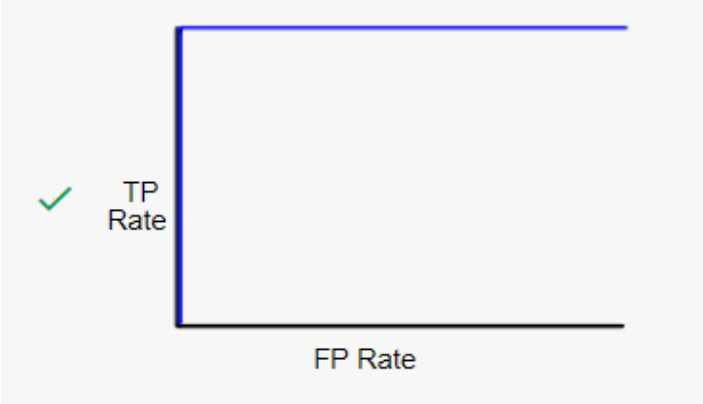
False positive rate = false positive / (false positive + true negative)



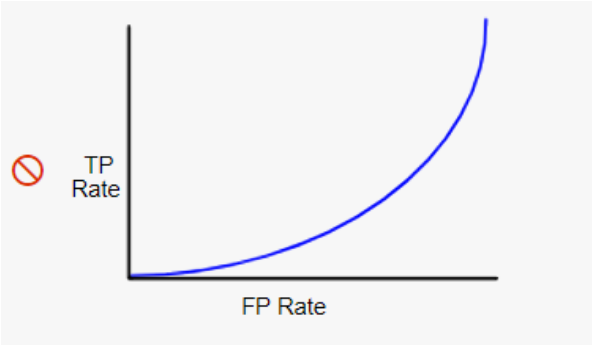
$$\text{Specificity} = \frac{TN}{TN + FP}$$

An excellent model has AUC near to the 1 which means it has good measure of separability. A poor model has AUC near to the 0 which means it has worst measure of separability. In fact, it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s. And when AUC is 0.5, it means model has no class separation capacity whatsoever.

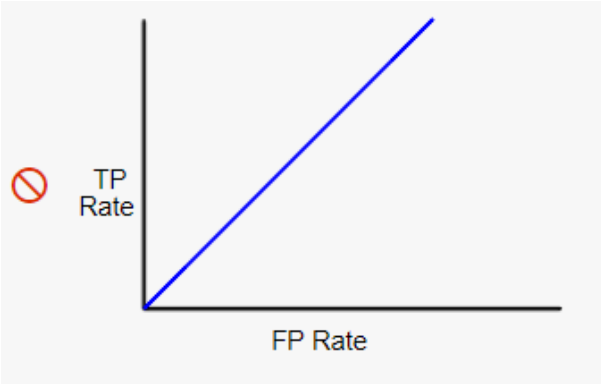




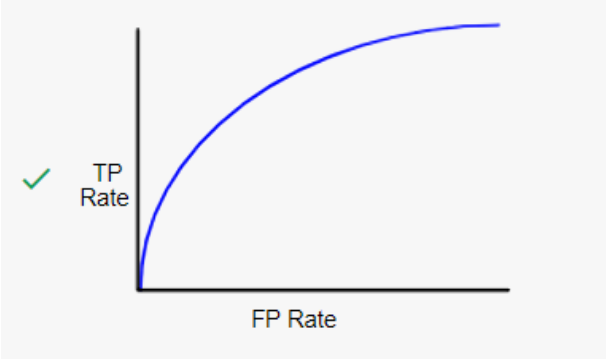
This is the best possible ROC curve, as it ranks all positives above all negatives. It has an AUC of 1.0.



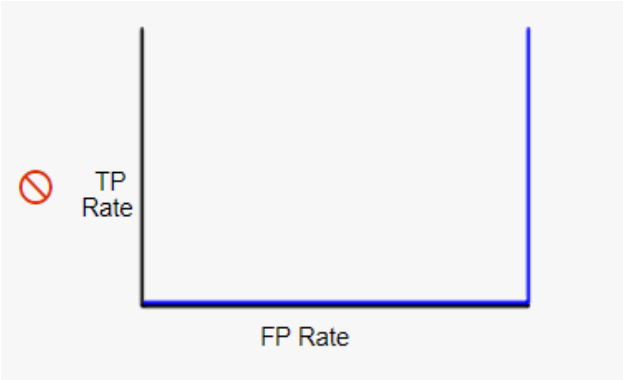
This ROC curve has an AUC between 0 and 0.5, meaning it ranks a random positive example higher than a random negative example less than 50% of the time. The corresponding model actually performs worse than random guessing! If you see an ROC curve like this, it likely indicates there's a bug in your data.



This ROC curve has an AUC of 0.5, meaning it ranks a random positive example higher than a random negative example 50% of the time. Thus, the corresponding classification model is basically worthless, as its predictive ability is no better than random guessing.



This ROC curve has an AUC between 0.5 and 1.0, meaning it ranks a random positive example higher than a random negative example more than 50% of the time. Real-world binary classification AUC values generally fall into this range.



This is the worst possible ROC curve; it ranks all negatives above all positives, and has an AUC of 0.0. If you were to reverse every prediction (flip negatives to positives and positives to negatives), you'd actually have a perfect classifier!