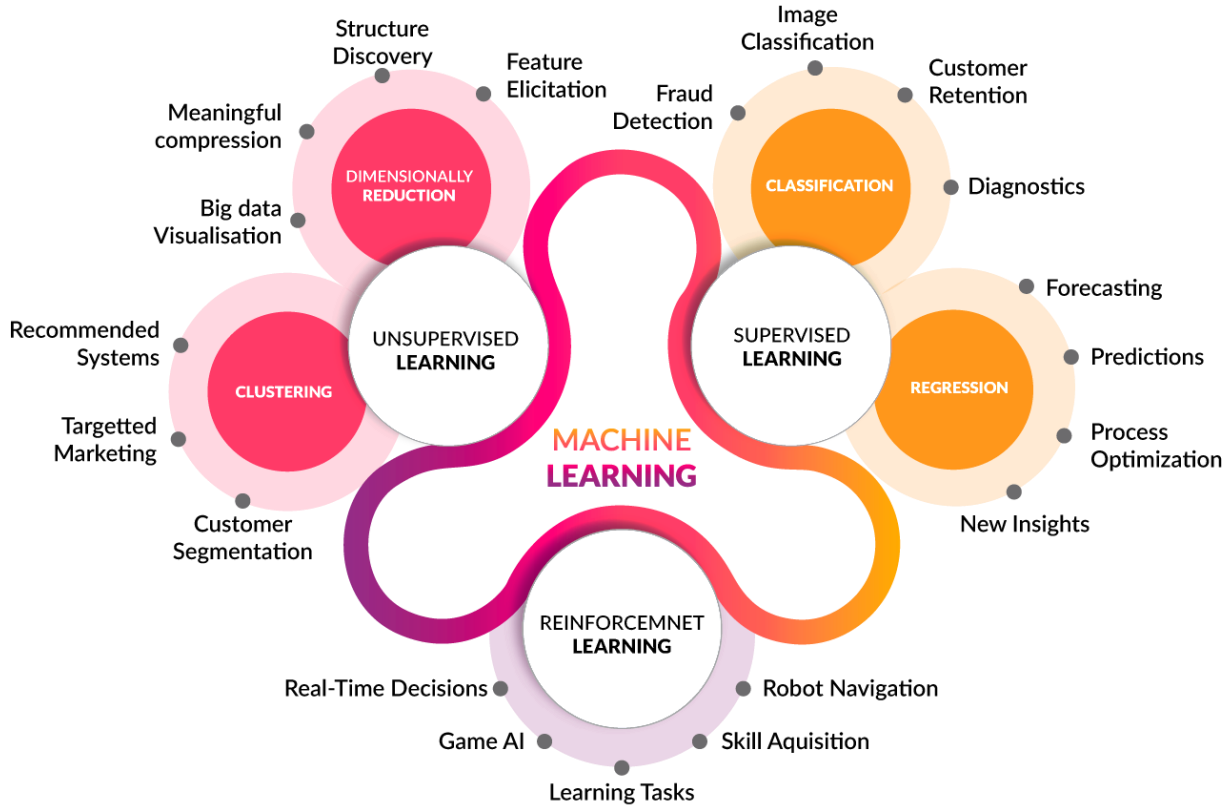


Model Evaluation and Selections

Unit 4

Model Evaluation

- Model evaluation is the technique of assessing the correctness of model on unseen data.
- Data for machine learning is divided into training and test data.

- Training data are seen by model during training and the pattern from the training data are learned during the training process.
- However test data are not seen by model i.e. they are not used by model to learn the patterns of data.
- Thus, test data are useful to evaluate the machine learning model whether they are able to learn from the training or not.

***Note**: It is common practice to split given data into 80% training (or 70% training) and 20% test (or 30% test) data. But it is not mandatory. Depending on volume of data it can be anything >= 70% or more.*

Model Evaluation

- Models can be evaluated using multiple metrics as supported by algorithms and data.
- Thus, the right choice of an evaluation metric is determined depending upon problem we are solving.
- A clear understanding of wide range of metrics can help data scientist/machine learning engineer to experiment and evaluate through various technique.
- For example, we are to evaluate the regression problem and we have various choices of evaluation metrics such as MSE, RMSE, MAE or MAPE etc.
- Regression and classification techniques can both be evaluated using various metrics which we are supposed to deal here.

Regression Evaluation Metrics

- Regression models produce continuous value as prediction. Thus our evaluation metrics take into account of continuous value.
- Some of the widely used regression metrics we will discuss here are:
  1. Mean Absolute Error (MAE)
  2. Mean Squared Error (MSE)
  3. Root Mean Squared Error (RMSE)
  4. R-Squared Error (R^2)

Regression Evaluation Metrics (contd.)

Error in regression problem is given as:

$$Error\ (e) = Actual\ value\ (y) - predicted\ value\ (\hat{y})$$

Then, for m training data, Total Error is given as:

$$Total\ Error\ (E) = \sum_{i=1}^{m} (y_i - \hat{y}_i)$$

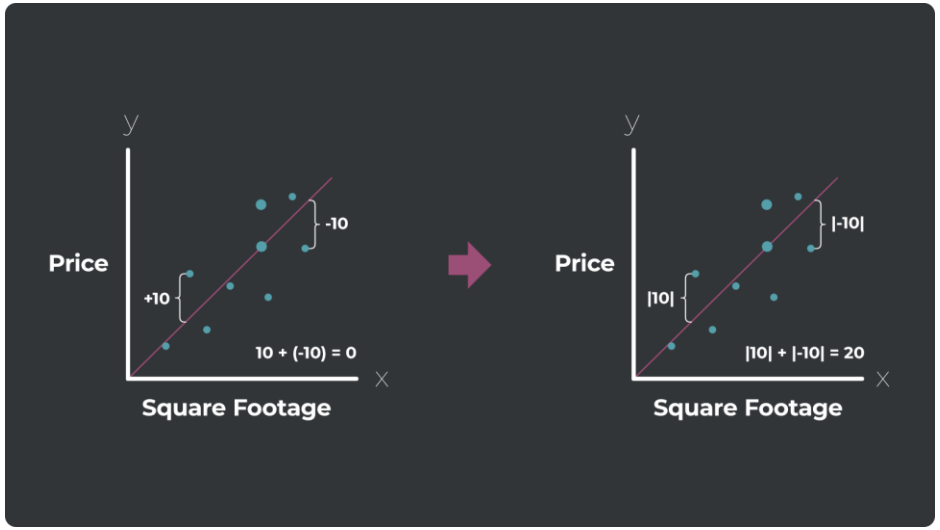
And, the mean error is given as:

$$Mean\ Error(e)\ = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i$$

Regression Evaluation Metrics (contd.)

However, these doesn't give us the right measure of error as there are three possibilities:

1. Error may be skewed towards max –ve value.
2. Or, error may be skewed towards max +ve value.
3. Or, error may settle around 0 mean value.



Regression Evaluation Metrics (contd.)

Case I:

Thus instead of this, we square each error value and get new set of metrics to use such as total squared error is given as:

$$Total\ Squared\ Error\ (E) = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{m} e_i^2$$

And, Mean Squared Error (MSE) is given as:

$$Mean\ Squared\ Error\ (MSE) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

Regression Evaluation Metrics (contd.)

Like As mean, MSE are also are sensitive towards outliers and will show very high error even if few outliers are present in the otherwise well-fitted model predictions.

Computing the Square root of MSE gives rise to Root Mean Squared Error (RMSE). This is particularly useful to scale down the errors closer to the actual values i.e. Square root tentatively minimizes the squared portion. It is given as:

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

Thus, Mean Squared Error (MSE) is a regression metric that calculates the sum of square of differences between actual value and the predicted value divided by total number of observations.

Regression Evaluation Metrics (contd.)

Another commonly used metric in regression is R-Squared. R-Square measures the proportion of variance of the dependent variable explained by the independent variable.

$$R^2 = 1 - \frac{variance(model)}{variance(average)}$$

Where,

$$variance(model) = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

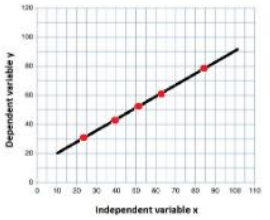
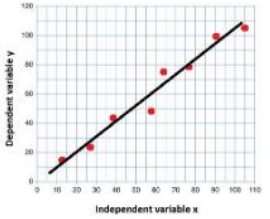
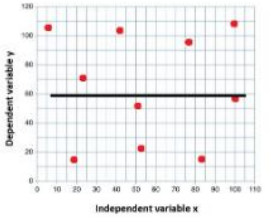
And,

$$variance(average) \sum_{i=1}^{m} (y_i - \bar{y}_i)^2$$

Regression Evaluation Metrics (contd.)

This gives us the comparison of how must regression line is better than a mean line. Thus, considering the average prediction would be 50% correct mostly, we always tend to prefer $R^2 > 0.5$ for any regression problem as value of R^2 ranges from 0 to 1.

Closer the value of R^2 to 1, the better the regression model is.

R^2 Values	Interpretation	Graph
$R^2 = 1$	All the variation in the y values is accounted for by the x values	
$R^2 = 0.83$	83% of the variation in the y values is accounted for by the x values	
$R^2 = 0$	None of the variation in the y values is accounted for by the x values	

Regression Evaluation Metrics (contd.)

Case II:

Instead of squaring the error, we can also take absolute value of error and this is called Absolute error. Taking sum of absolute error avoids the problem of negative error and zero error. Then, Mean Absolute Error is given as:

$$Mean\ Absolute\ Error\ (MAE) = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|$$

Also, we can also compute Mean Absolute Percentage Error (MAPE) as:

$$Mean\ Absolute\ Percentage\ Error\ (MAPE) = \frac{1}{m}\sum_{i=1}^{m}\frac{|y_i - \hat{y}_i|}{y_i}$$

Mean absolute percentage error measures the average magnitude of error produced by a model, or how far off predictions are on average.

Regression Evaluation Metrics (contd.)

A MAPE value of 20% means that the average absolute percentage difference between the predictions and the actuals is 20%. In other words, the model's predictions are, on average, off by 20% from the real values.

A lower MAPE value indicates a more accurate prediction – an MAPE of 0% means the prediction is the same as the actual, while a higher MAPE value indicates a less accurate prediction.

Classification Evaluation Metrics

- Likewise to regression tasks, classification tasks can also be evaluated using couple of metrices:
  1. Accuracy
  2. Recall
  3. Precision
  4. F1-score
  5. RUC and AUC curve etc.
- To compute all these metrices, we first need to understand the concept of confusion metrics.
- Confusion matrix refers to the matrix like structure formed from the prediction of classification tasks on unseen data sets.
- This is the basis for the evaluation of classification algorithms.

Confusion Metrics

For any binary classification job, we either predict True (1) or False (0) and also the actual value is either True(1) or False (0).

		Actual	
		True	**False**
Predicted	**True**	*True Positive (TP)*	*False Positive (FP)*
	False	*False Negative (FN)*	*True Negative (TN)*

- **True Positive**: Actual True classified as True
- **False Positive**: Actual False but classified as True
- **False Negative**: Actually True but classified as False
- **True Negative**: Actually False and Classified as False

Accuracy

Accuracy is one of the common measure to evaluate the performance of classification algorithms.

Accuracy can, now, be easily computed from the confusion metrics:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Thus, accuracy is the ratio of total correct instances to the total instances.

$$Accuracy = \frac{Total\ Correctly\ classified\ instances}{TOtal\ no.of\ instances}$$

Accuracy value falls between 0 and 1 but measured in terms of %. Thus, accuracy of 0.862 will be read as 86.2% accurate model.

Accuracy

An algorithm trained in a disease prediction tasks, following data was observed.

		Actual	
		Ill	**Not Ill**
Predicted	**Ill**	*48*	*2*
	Not Ill	*5*	*45*

Computing Accuracy,

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{48 + 45}{48 + 2 + 45 + 5} = \frac{93}{100} = 0.93$$

This means model shows 93% accuracy on test data.

Is it a good model or bad model?

Accuracy (contd.)

While on its surface, the concept of model accuracy might seem like the perfect measure for understanding a model's reliability, but it doesn't tell the whole story.

Let's understand from the above scenarios:

- Beside the correct prediction i.e. Actual ill as ill and actually not ill as not ill, model also has predicted **not ill as ill** (*not too dangerous – may have to go for more round of medical tests*) and **ill as not ill** (*dangerous as this could lead to death of the patient*)
- And in our case, there are 5 such cases where ill patient are not recognized and identified as not ill.
- Thus, though the measure is simple, it may not be good measure despite of high accuracy value.
- We now need additional set of measures and they are precision and recall.

Precision and Recall

Precision specifically calculates the percentage of correct positive predictions and given as:

$$Precision = \frac{TP}{TP + FP}$$

Recall, by contrast, calculates the percentage of actual positives correctly identified by the model.

$$Recall = \frac{TP}{TP + FN}$$

Both of these measures have value between 0 and 1. Precision is usually considered relating to the measure of quality and recall as measure of quantity.

Higher precision means that an algorithm returns more relevant results than irrelevant ones, and high recall means that an algorithm returns most of the relevant results (whether or not irrelevant ones are also returned).

Precision and Recall

Fitting in the above numerical,

$$Precision = \frac{TP}{TP + FP} = \frac{48}{48 + 2} = 0.96$$

And,

$$Recall = \frac{TP}{TP + FN} = \frac{48}{48 + 5} = 0.90$$

Interpretation:

Precision: Precision refers to the proportion of "true positives" (correctly identified instances) among all the cases that the model identified as positive. In other words, precision measures how accurate the model's positive predictions are.. Higher the precision value is considered a better one.

Recall: Conversely, recall refers to the proportion of true positives that the model correctly identified among all the actual positive instances in the data. In other words, recall is the measure of how well the model can identify all the positive instances in the data. Higher the recall value is considered the better one.

Trade-off between Recall and Precision

- We wish to get highest precision and recall as far as possible but the trade-off between precision and recall occurs because improving one usually comes at the expense of the other.
- If you increase precision, it will reduce recall and vice versa. This is called the precision/recall tradeoff.
- For example, for our dataset, we can consider that achieving a high recall is more important than getting a high precision – we would like to detect as many heart patients as possible.
- And, for some other models, like classifying whether or not a bank customer is a loan defaulter, it is desirable to have high precision since the bank wouldn't want to lose customers who were denied a loan based on the model's prediction that they would be defaulters.
- Hence there are numerous other situations, where either precision or recall is important. Or sometime both are equally important.
- For example, for our model, if the doctor informs us that the patients who were incorrectly classified as ill are equally important since they could be indicative of some other ailment, then we would aim for not only a high recall but a high precision as well.

F1 Score

F1-Score is another evaluation metrics used in classification which combines precision and recall.

Mathematically,

$$F1\ Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2\ *Precision\ *Recall}{Precision + Recall}$$

This is the harmonic mean of Precision and Recall because harmonic mean encourages similar values for precision and recall. That is, the more the precision and recall scores deviate from each other, the worse the harmonic mean would be.

Positive and Negative Rates

True Positive Rate (TPR) is a synonym for recall (aka. **Sensitivity**) and is therefore defined as follows:

$$TPR = \frac{TP}{TP + FN}$$

True Negative Rate (TNR) (aka. **Specificity**) tells us what proportion of the negative class got correctly classified.

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

False Positive Rate (FPR) tells us what proportion of the negative class got incorrectly classified by the classifier.

$$FPR = \frac{FP}{FP + TN} = 1 - Specificity$$

False Negative Rate (FNR) tells us what proportion of the positive class got incorrectly classified by the classifier. :

$$FNR = \frac{FN}{TP + FN} = 1 \ - Sensitivity$$

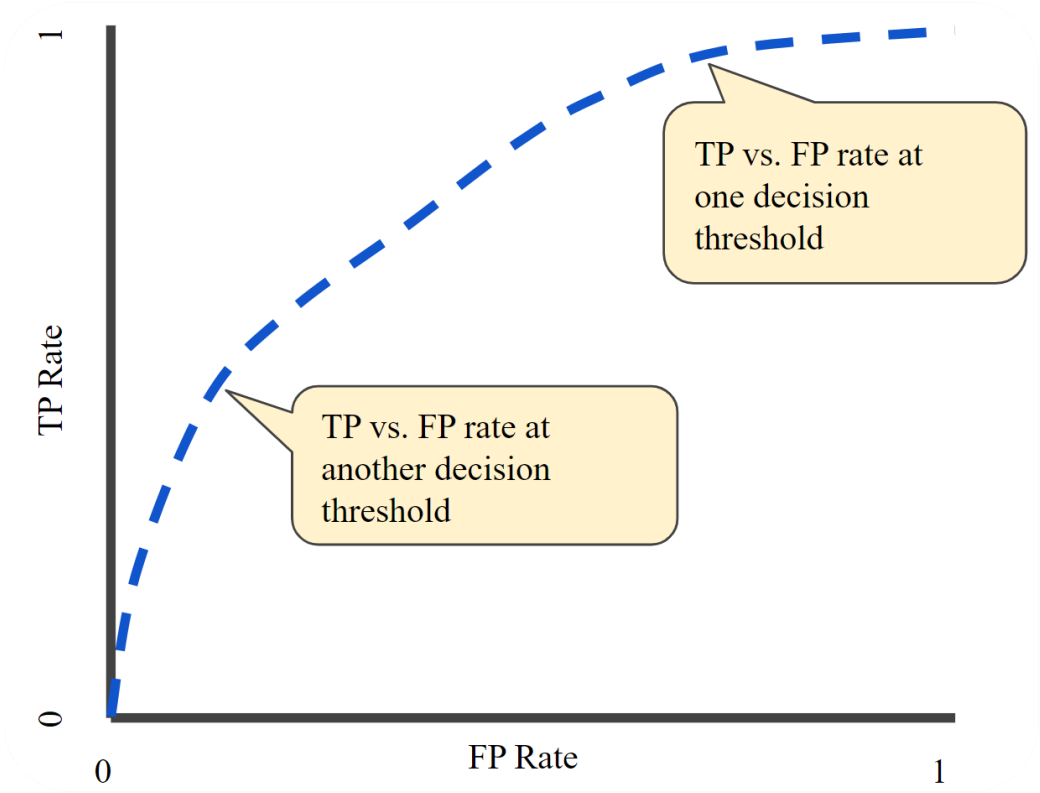
ROC Curve

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

It helps us see how the model makes decisions at different levels of certainty.

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.



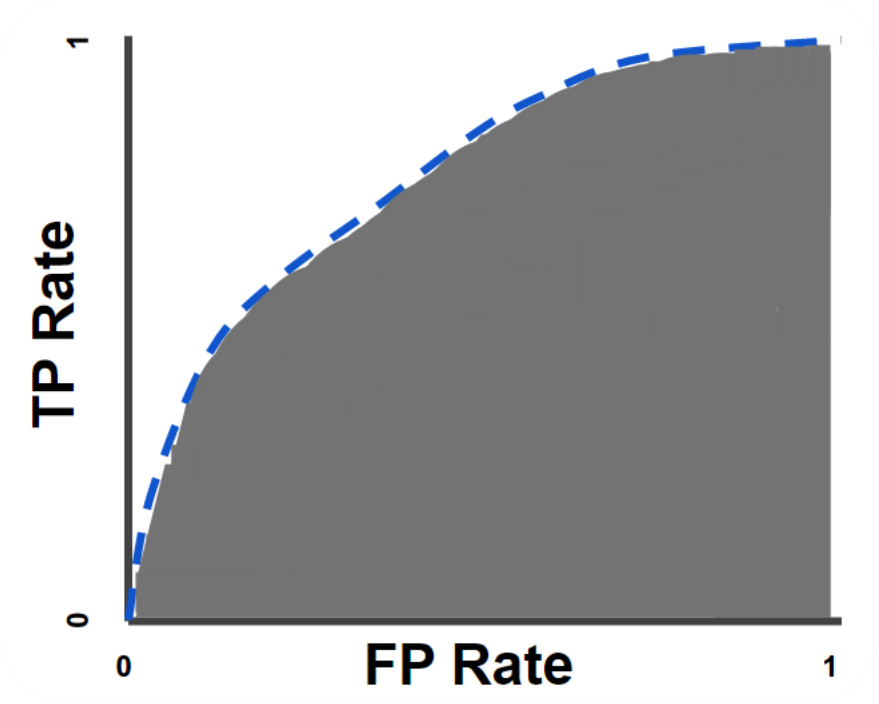
Classification: ROC Curve and AUC | Machine Learning | Google for Developers

ROC Curve

- If area under the ROC curve is higher, the better is the model performance.
- If the curve is somewhere near the 50% diagonal line, it suggests that the model randomly predicts the output variable.
- To compute the points in an ROC curve, we could evaluate a logistic regression model many times with different classification thresholds, but this would be inefficient. Fortunately, there's an efficient, sorting-based algorithm that can provide this information for us, called AUC.

Area under the Curve (AUC)

The Area Under the Curve (AUC) is the measure of the ability of a binary classifier to distinguish between classes and is used as a summary of the ROC curve.



The higher the AUC, the better the model's performance at distinguishing between the positive and negative classes. When AUC = 1, the classifier can correctly distinguish between all the Positive and the Negative class points. If, however, the AUC had been 0, then the classifier would predict all Negatives as Positives and all Positives as Negatives.

When 0.5<AUC<1, there is a high chance that the classifier will be able to distinguish the positive class values from the negative ones. This is so because the classifier is able to detect more numbers of True positives and True negatives than False negatives and False positives.

So, the higher the AUC value for a classifier, the better its ability to distinguish between positive and negative classes.

Multi-class Evaluation

Before we understand the multi-class evaluation, we need to understand the multiclass classification. Logistic Regression we discussed early is only for the binary classification i.e. fits for the cases when there are only two classes.

Combining multiple Logistic Regression, we can achieve the multi-class classification but this is not the efficient approach.

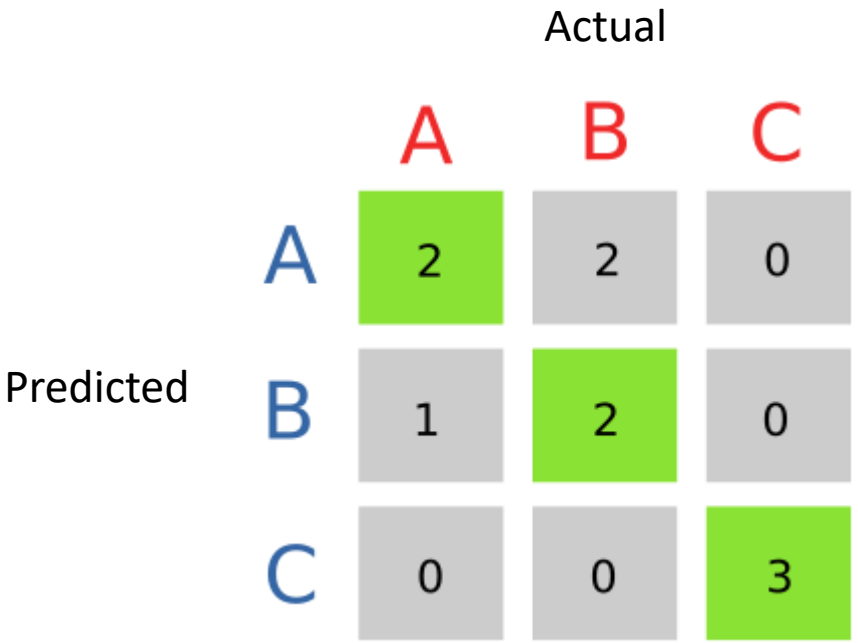
Instead, we have **Softmax Regression** which performs multi-class classification. Softmax Regression is given as:

$$h(x) = g(z_j) = \frac{\exp(z_j)}{\sum \exp(z_j)}$$

Multi-class Evaluation (contd.)

Now, if there are C number of classes, then its obvious that the confusion matrix also has shape $C \times C$.

Say, we have 3 classes for the prediction. Thus we get confusion matrix of 3×3 as below:



Then, we need to compute TP, FP, TN and FN for each class separately such that

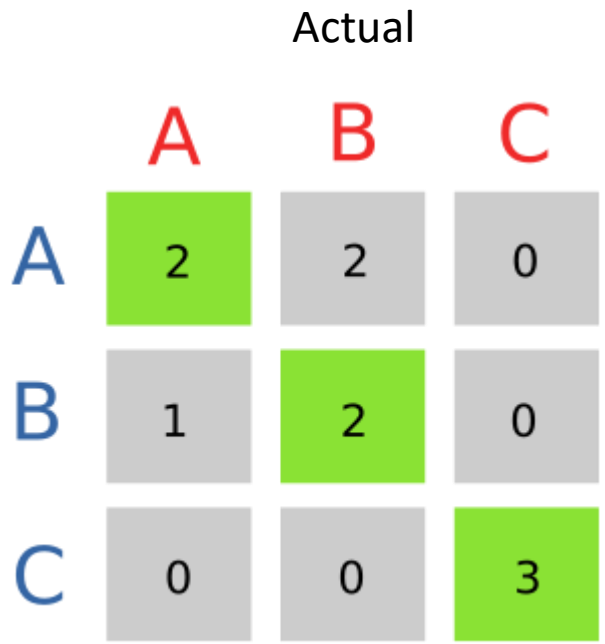
True Positive (TP): Both actual and predicted are same.

False Positive (FP): Sum of all entries in the row except TP.

True Negative (TN): Sum of all entries in rows and columns except row and columns of calculating class.

False Negative (FN): Sum of all entries in the column except TP.

Multi-class Evaluation (contd.)



From the given scenario, for the class A

TP = 2 (*Predicted A, Actual A*)

FP = 2 (*Predicted A, Actual B + Actual C*)

FN = 1 (*Predicted B + Predicted C, Actual A*)

TN = 5 (*Predicted B + Predicted C, Actual B + Actual C*)

Can you calculate for the class B and Class C?

Similarly, for Class B: TP = 2. FP = 1, FN = 2, TN =5

And for Class C: TP = 3, FP = 0, FN = 0, TN = 7

Multi-class Evaluation (contd.)

- Now, we can compute accuracy, precision, recall etc.
- Remember, in multi-class classification we calculate accuracy of each class separately and then we calculate weighted average accuracy.
- For ease, we assign equal weight to each class i.e. $w_k = \frac{1}{|C|}$
- Then, we have two approach to calculate Precision/Recall:
  1. **Macro Averaged:** This is the average of individual classes score.
  2. **Micro Averaged:** This is calculated from the individual classes' true positives (TPs), true negatives (TNs), false positives (FPs), and false negatives (FNs) of the model.

Multi-class Evaluation (contd.)

$$Macro\ Averaged\ Precision = \frac{Prec_1 + Prec_2 + \cdots + Prec_{|C|}}{|C|}$$

$$Micro\ Averaged\ Precision = \frac{TP_1 + TP_2 + \cdots + TP_{|C|}}{TP_1 + TP_2 + \cdots + TP_{|C|} + FP_1 + FP_2 + \cdots + FP_{|C|}}$$

Similarly,

$$Macro\ Averaged\ Recall = \frac{Rec_1 + Rec_2 + \cdots + Rec_{|C|}}{|C|}$$

$$Micro\ Averaged\ Precision = \frac{(TP_1 + TP_2 + \cdots + TP_{|C|})}{(TP_1 + TP_2 + \cdots + TP_{|C|}) + (FP_1 + FP_2 + \cdots + FP_{|C|})}$$

Multi-class Evaluation (contd.)

Similarly, we can have two approach for F1 Score:

1. **Macro Averaged:** Compute F1 Score of every class and take average of them.
2. **Micro Averaged:** Calculate Micro-averaged precision and recall score and take their harmonic mean.

$$Macro\ Average\ F1\ Score = \frac{F1_1 + F1_2 + \cdots + F1_{|C|}}{|C|}$$

$$Micro\ Average\ F1\ Score = \frac{2\ *\ Micro\ Precision\ *Micro\ Recall}{Micro\ Precision + Micro\ Recall}$$

Multi-class Evaluation (contd.)

When to use micro-averaging and macro-averaging scores?

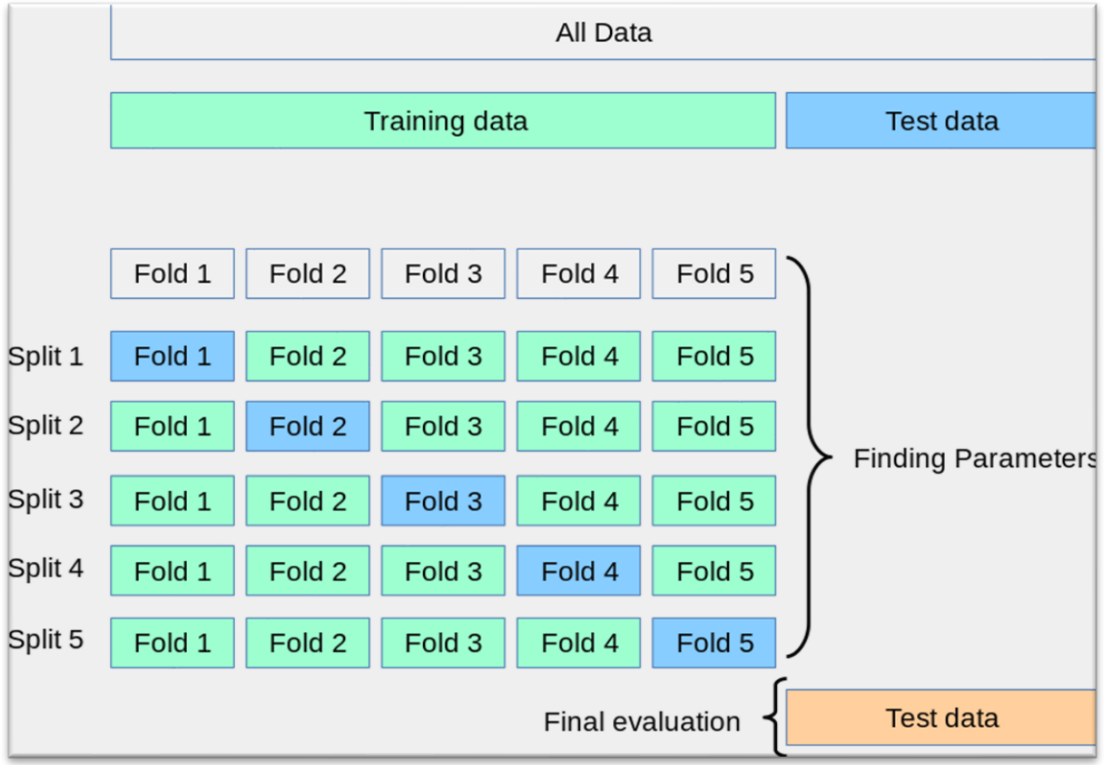
- Use micro-averaging score when there is a need to weight each instance or prediction equally.
- Use macro-averaging score when all classes need to be treated equally to evaluate the overall performance of the classifier with regard to the most frequent class labels.
- Use weighted macro-averaging score in case of class imbalances (different number of instances related to different class labels). The weighted macro-average is calculated by weighting the score of each class label by the number of true instances when calculating the average.

Model Selection

- Model Selection is the process of selecting one among many candidate algorithm to deploy to production or real world application.
- For selecting the best model among multiple candidate models, we need multiple strategies:
  1. **Hold Out Set:** We need to divide data into train set, validation set and test sets for model evaluation. Validation dataset can be used to access model performance through hyper –parameter tuning.
  2. **K-Fold Cross Validation:** We can evaluate performance of models using K-fold cross validation.

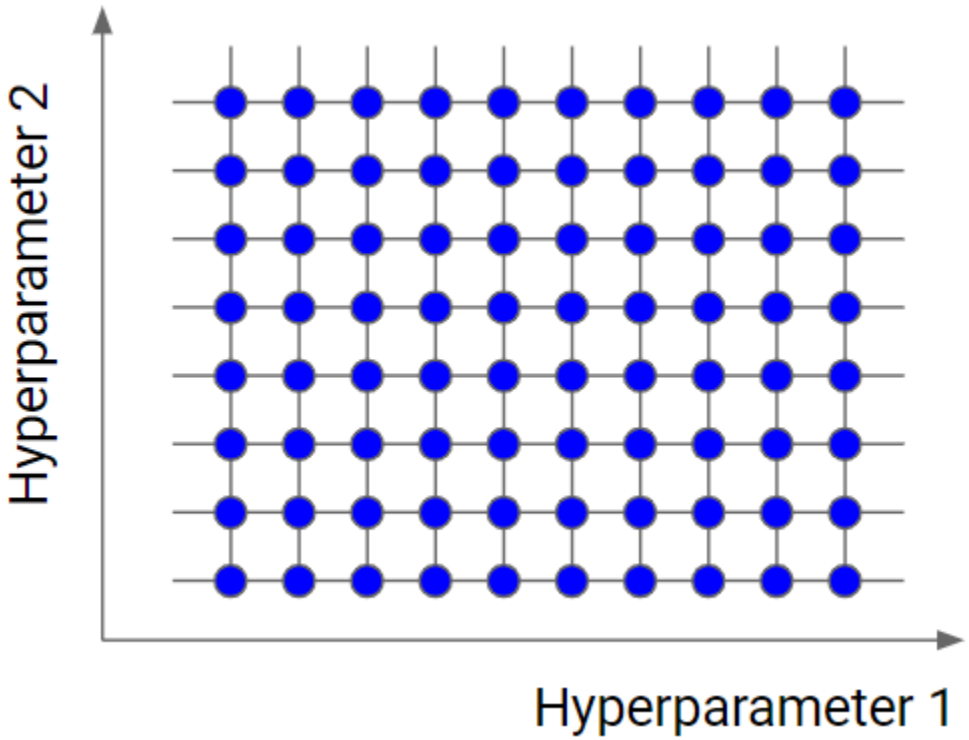
Model Selection (contd.)

A model is trained using k-1 of the folds as training data; the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).



Model Selection (contd.)

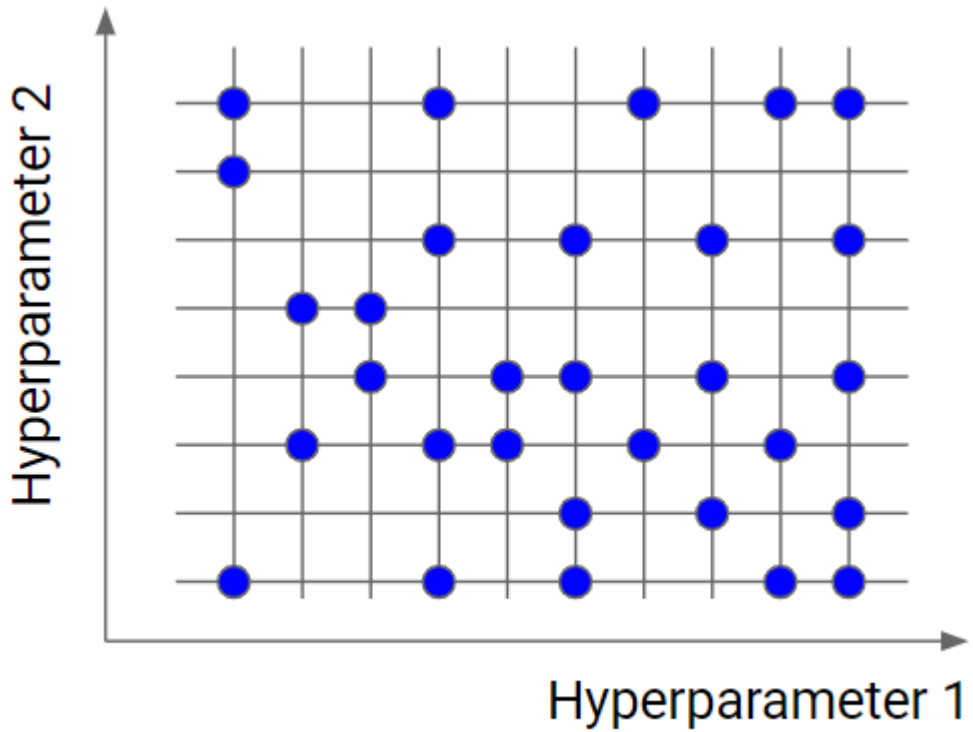
3. **Hyperparameter Optimization:**
   Hyperparameter Optimization is the process of finding the best possible values for the hyperparameters to optimize performance metric. We can use multiple technique to find best values for hyperparameter such as Grid Search , Random Search etc.
   a) **Grid search** is an exhaustive search algorithm that looks out for all the possible combinations in order to find the best point in the domain. As it experiments, all the possibilities, it is slower in nature.



Grid Search

Model Selection (contd.)

b) **Random Search** is similar to grid search, but instead of using all the points in the grid, it tests only a randomly selected subset of these points. The smaller this subset, the faster but less accurate the optimization. The larger this dataset, the more accurate the optimization but the closer to a grid search.



Random Search

End of the Chapter