Model Evaluation – Overfitting

-

Overview

- Overfitting vs Underfitting
- Underfitting to Overfitting in Prediction(Regression)
- Underfitting to Overfitting in Classification
- Impact of Overfitting on Performance
- Approaches to reduce Overfitting
- Summary

What is Overfitting

- ML models can generalize and produce highly complex explanations of relationships (models) between predictor variables and response variable where the 'fit' maybe excellent.

- However, when used with new unseen data, models of great complexity may not do so well!

- Overfitting happens when a model learns the details and also noise in the training data to the extent that it negatively impacts the performance of the model on new data. (excellent performance on training, but poor generalization on new data).

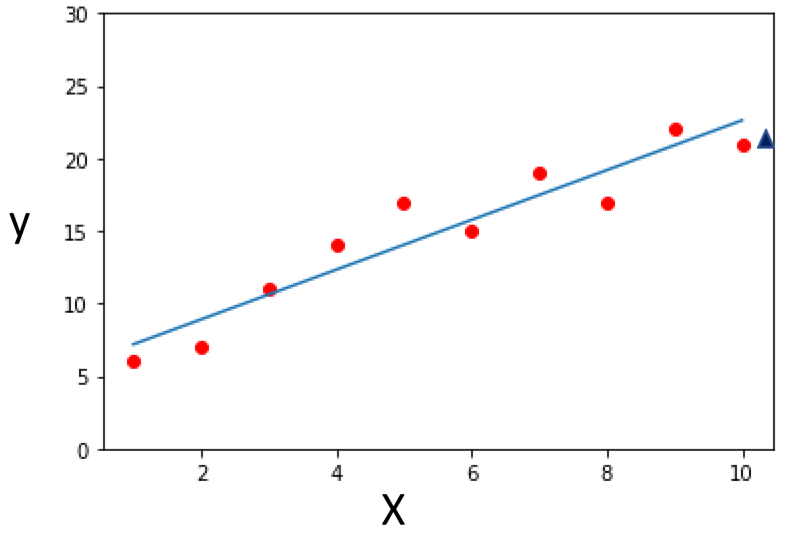
- Overfitting produces poor predictive performance – past a certain point, the error rate on new data starts to increase

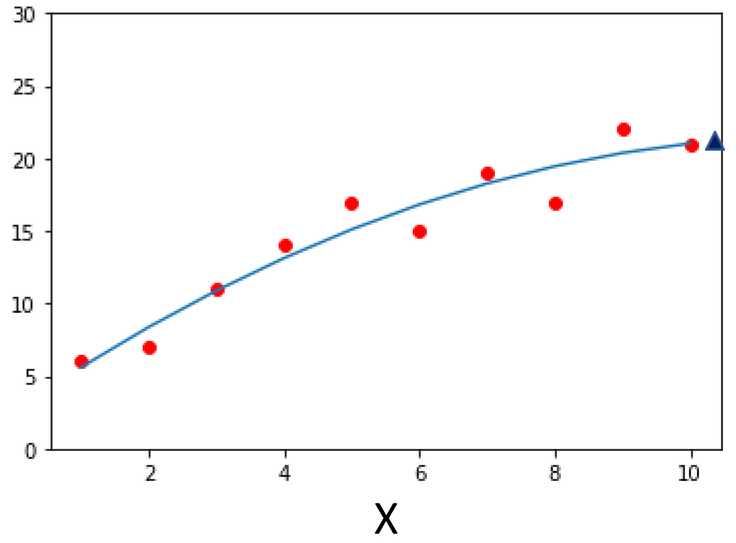
- Conversely, Underfitting occurs when the model fails to capture or generalize the underlying pattern in the data, mostly due to over-simplicity of the model. (poor performance on training as well as poor generalization on new data)

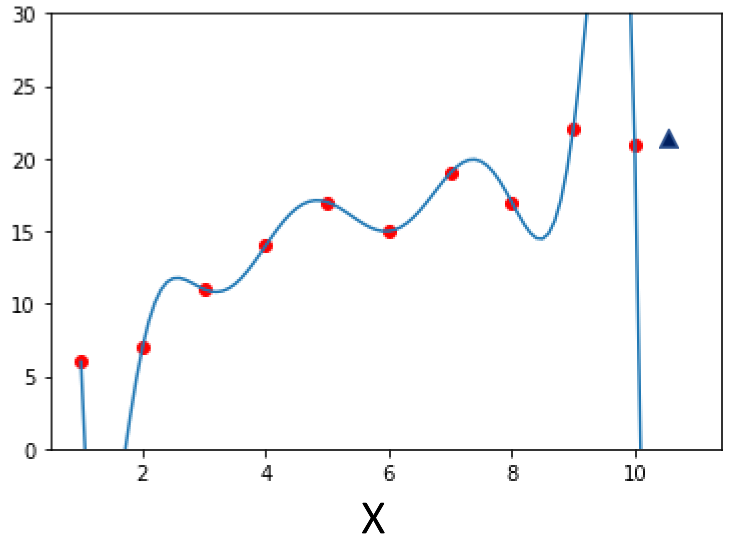
Underfitting to Overfitting – Prediction (Regression)

Given a dataset, where

X = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] and
y = [6, 7, 11, 14, 17, 15, 19, 17, 22, 21]







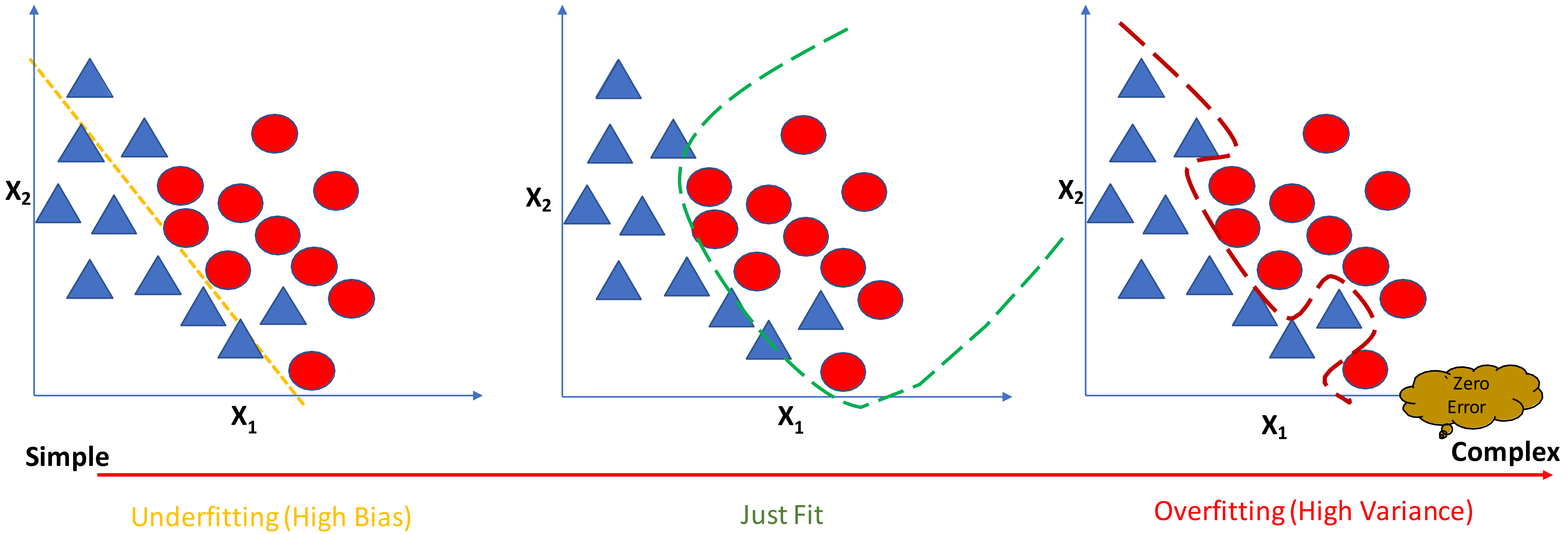
Linear Model	Quadratic Model	Polynomial Model
$y = aX + c$	$y = a_1X + a_2X^2 + c$	$y = a_1X+a_2X^2+a_3X^3+a_4X^4+a_5X^5+a_6X^6+a_7X^7+a_8X^8+ a_8X^8+c$
Accuracy: 89.5 %	Accuracy: 93%	Accuracy: 99.99%

Zero Error

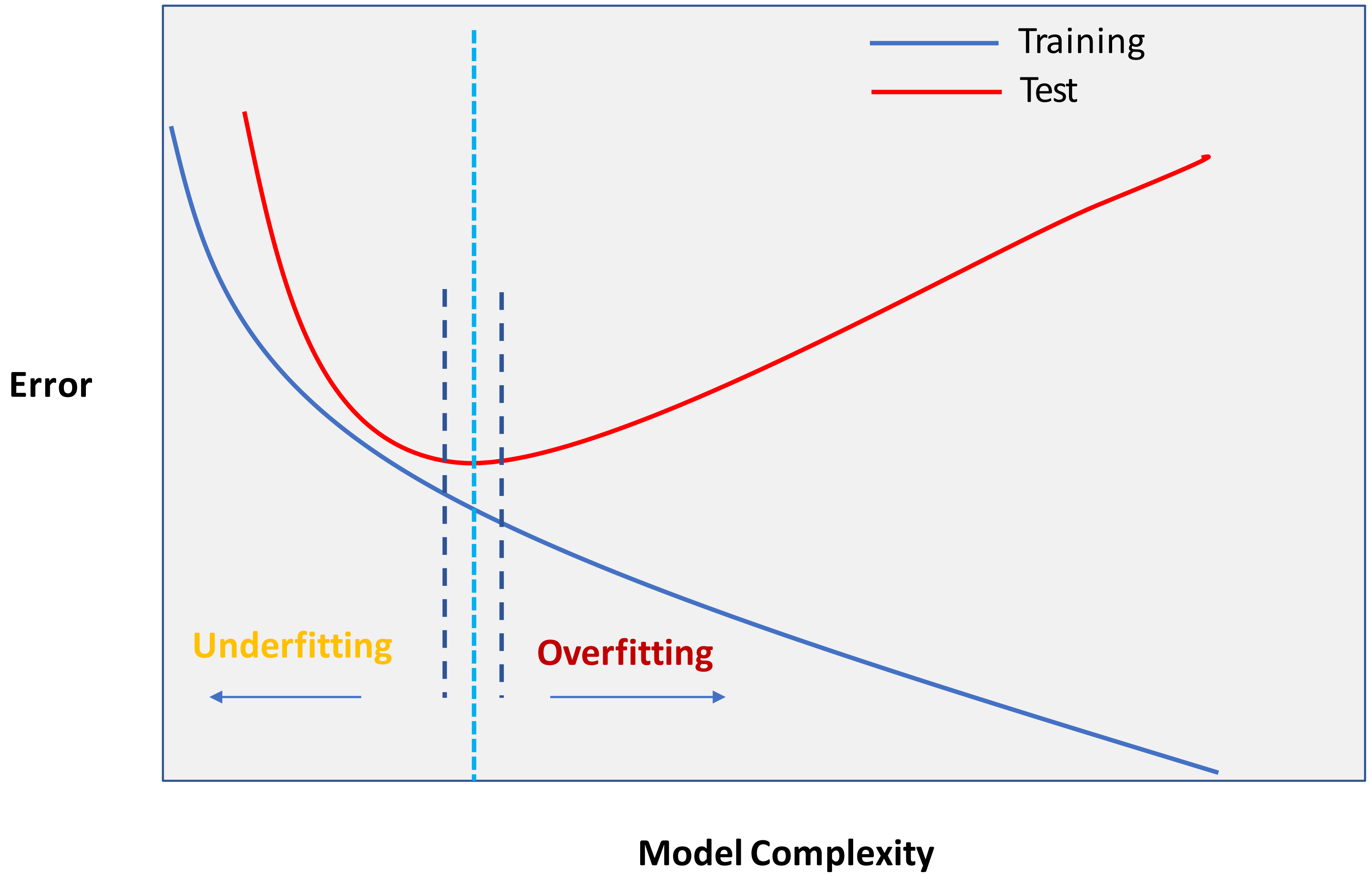
Simple → **Complex**

Underfitting (High Bias)	Just Fit	Overfitting (High Variance)

Underfitting to Overfitting – Classification



Impact of Overfitting on Performance

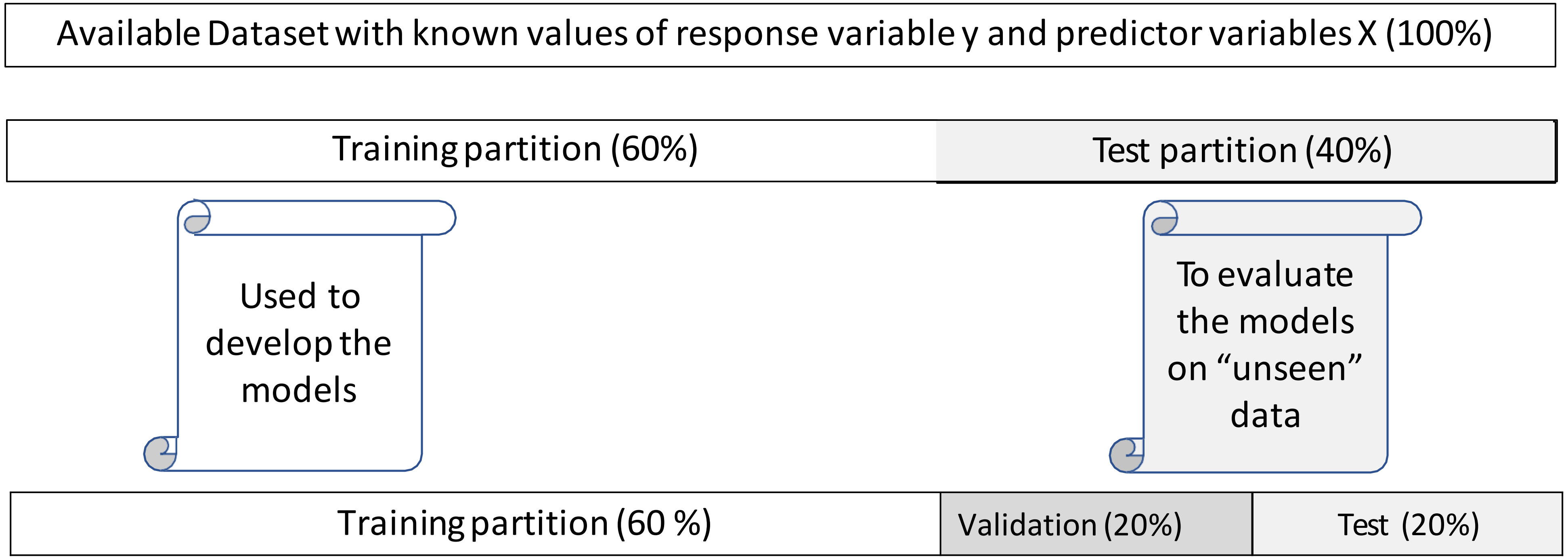


Approaches to reduce Overfitting

- Partitioning the available data into Training – Validation – Test partitions and performing Cross Validation
- Reducing the number of features used in building the model
- Regularization

Partitioning: Train-Test split

- How well will our model perform on fresh data, the model has not seen before and how to reduce the scope for overfitting?
  - ➢ Partition the available data into minimum 2 parts or ideally 3 parts


Available Dataset with known values of response variable y and predictor variables X (100%)
Training partition (60%)
Test partition (40%)
Used to develop the models
To evaluate the models on “unseen” data
Training partition (60 %)
Validation (20%)
Test (20%)

Need for a 3rd Partition

- When a model is developed using training data, it can overfit the training data and hence the need to assess the performance on ‘unseen’ data, a.k.a. validation partition

- Assessing multiple models on the same ‘unseen’ data, can again lead to overfitting on this data (i.e. the validation partition)

- Hence the final selected model is applied to the third partition (Test partition) to give an unbiased estimate of its performance on ‘new’ data. Test partition also referred to as ‘Holdout’ set

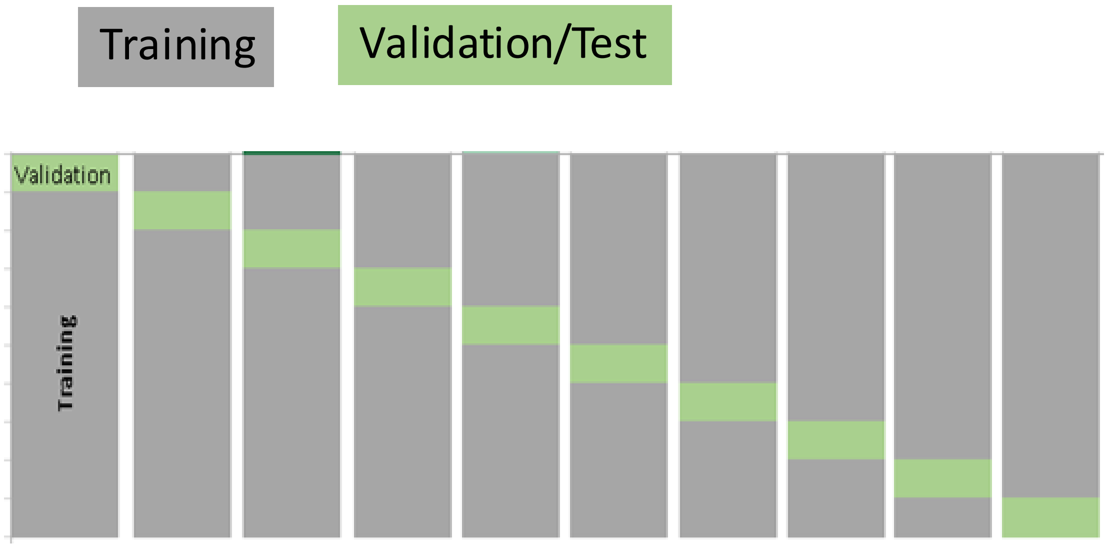
Note - ML literature typically refers to the 2 partition scenario as Training/Test, whereas some literature refer to them as Training/Validation.

Shortcomings of training, validation and test split

- Given the split into only one set of training, validation, test partitions, the performance of our models are highly dependent on the nature of data in those partitions
- Thus performance might vary if the split points were different and training and evaluation were on different subsets of data
- What if we could do this split into training and validation set multiple times, each time on different subsets of the same data, and then train and evaluate our models each time to look at the average performance of the models across multiple evaluations?

k-fold Cross Validation

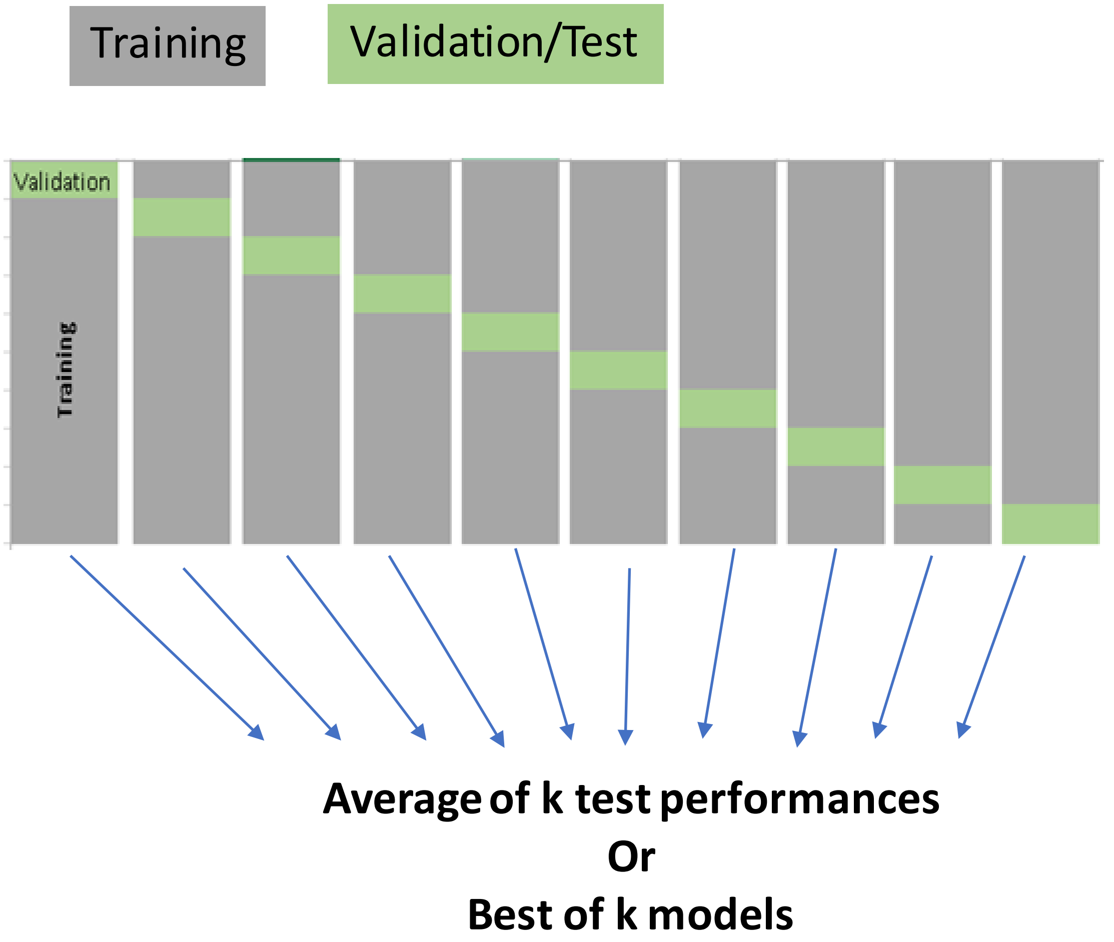
- An alternative to data partitioning, esp. when the number of records in the dataset is small
- Partitioning the data into “folds,” or non-overlapping (disjoint) sub-samples.
- If we choose *k* = 10 folds, meaning that the data are randomly partitioned into 10 equal parts, where each fold has 10% of the observations.



Ref: https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/

k-fold Cross Validation

- Randomly split your entire dataset of n instances into k 'folds' (each of size n/k)
- For each k-fold in your dataset, build a model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for kth fold
- Record the error you see on each of the predictions
- Repeat this until each of the k-folds has served as the test set
- The average of your k recorded errors is called the cross validation error and will serve as your performance metric for the model



Ref: https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/

How to choose value of k in ‘k-fold cross validation’

- On small datasets, lower value of k could increase bias and hence not desirable (similar to the use of traditional partitions – train/validation/test)

- Higher values of k reduces bias, but increases scope for variability

- Special case of k = n, referred to as ‘Leave one out cross validation (LOOCV)’ or n-fold cross validation (where n = number of training samples)
  - Reserve only one data point from the available dataset, and train the model on the rest of the data. This process iterates for each data point and hence k=n.
  - Computationally expensive

Ref: https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/

k-fold Cross-Validation for Parameter Tuning

- Sometimes cross validation is built into ML algorithm implementation, with the results of the cross validation used for choosing the algorithm's parameters

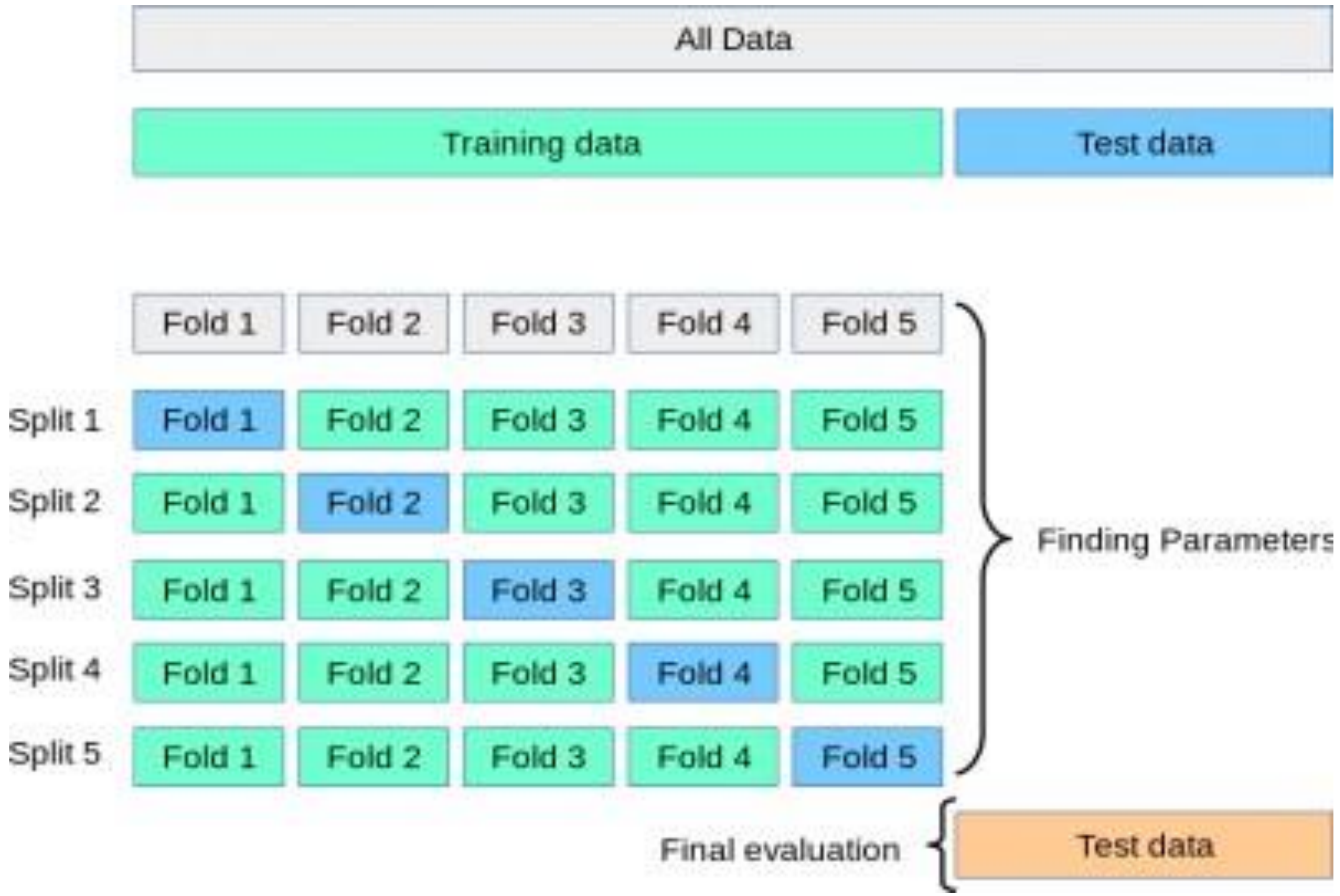


Image Source: https://towardsdatascience.com/cross-validation-and-hyperparameter-tuning-how-to-optimise-your-machine-learning-model-13f005af9d7d

Variations of Cross validation

- **Train/Test Split**: k=2 such that a single train/test split is created to evaluate the model.
- **LOOCV**: Leave-one-out cross-validation, k=n
- **Stratified**: Each fold has the same proportion of observations with a given categorical value (Genuine/Fraudulent), such as the class outcome value.
- **Repeated**: k-fold cross-validation procedure is repeated n times, the data sample is shuffled prior to each repetition, which results in a different splits
- **Nested**: k-fold cross-validation performed within each fold of cross-validation, often to perform hyperparameter tuning during model evaluation. Also called double cross-validation.

Ref: https://machinelearningmastery.com/k-fold-cross-validation/

Why do we need to evaluate models?

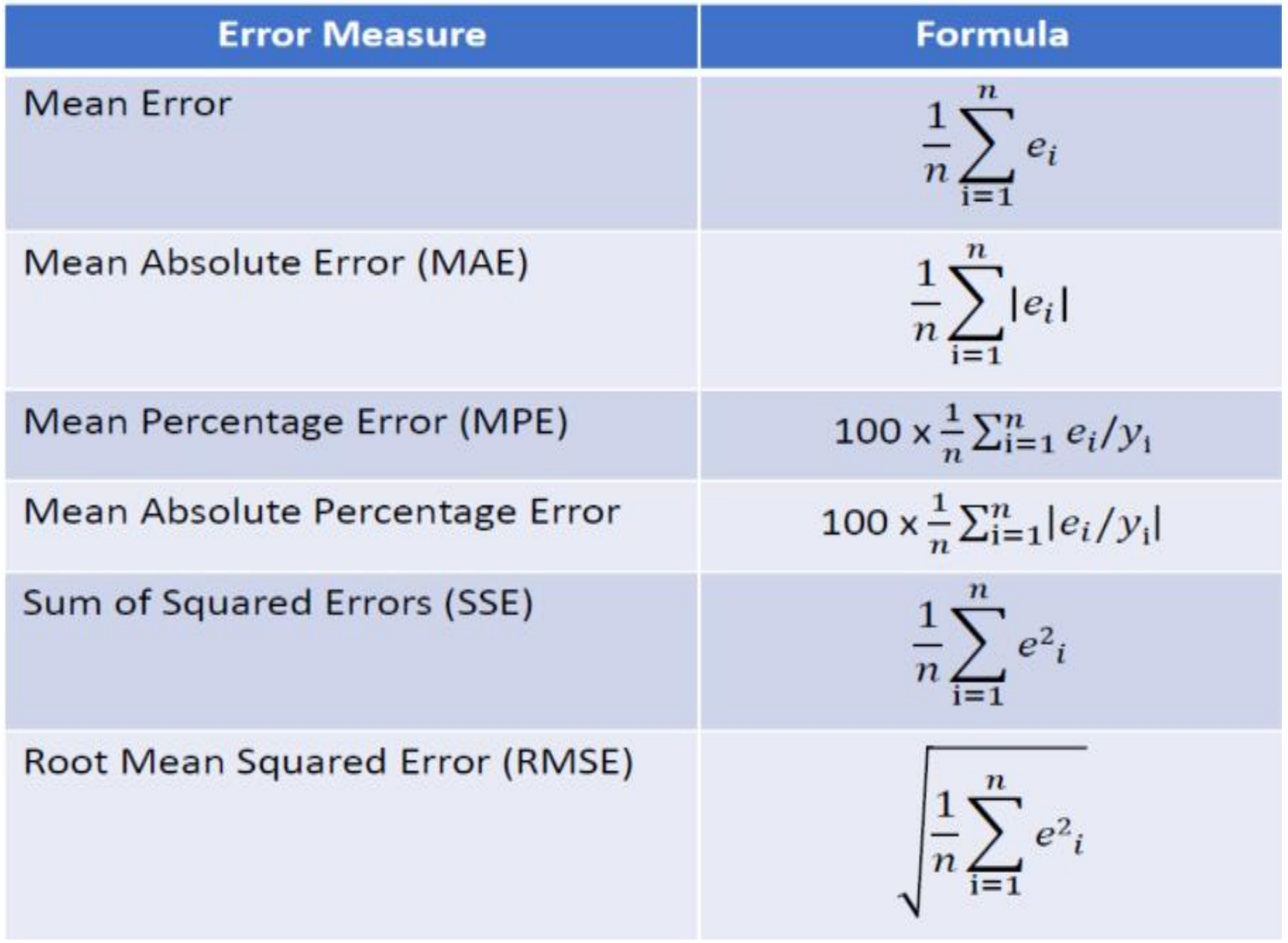
- Multiple ML algorithms applicable to classification/prediction
- Wide choice of parameter and/or hyperparameter settings possible in these algorithms
- Hence the need to evaluate each model's performance
- In all cases, performance to be evaluated on validation/test data (to avoid wrong interpretations from overfitting on training data)

Evaluating performance in Prediction

- In such scenarios, we need to evaluate how the model predicts **new data**, not how well it fits the data it was trained with (goodness-of-fit)

- Key component of most performance measures is the difference between actual *y* and predicted y' which is referred to as the 'error' :

  $e = y - y'$



Error Measure	Formula
Mean Error	$\frac{1}{n}\sum_{i=1}^{n} e_i$
Mean Absolute Error (MAE)	$\frac{1}{n}\sum_{i=1}^{n} \lvert e_i \rvert$
Mean Percentage Error (MPE)	$100 \times \frac{1}{n}\sum_{i=1}^{n} e_i/y_i$
Mean Absolute Percentage Error	$100 \times \frac{1}{n}\sum_{i=1}^{n} \lvert e_i/y_i \rvert$
Sum of Squared Errors (SSE)	$\frac{1}{n}\sum_{i=1}^{n} e^2{}_i$
Root Mean Squared Error (RMSE)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n} e^2{}_i}$

Evaluating performance in Classification

Most Classification algorithms classify via a 2-step process:

For each record,

1. Compute **probability of belonging to class '1'**
2. Compare to cutoff value, and classify accordingly

(Default cutoff value is 0.50, If >= 0.50, classify as "1",   If < 0.50, classify as "0")

- Can use different cutoff values and accordingly the classification output varies

- Error = classifying a record as belonging to one class when it actually belongs to another class.

- Error rate = percent of misclassified records out of the total records in the validation/test data

Confusion Matrix

		Actual Class	
		C_1	C_2
Predicted Class	C_1	$n_{1,1}$ = number of C_1 records classified correctly as C_1	$n_{2,1}$ = number of C_2 records classified incorrectly as C_1
	C_2	$n_{1,2}$ = number of C_1 records classified incorrectly as C_2	$n_{2,2}$ = number of C_2 records classified correctly as C_2

$$\text{err} = \frac{n_{1,2} + n_{2,1}}{n},$$

When One Class is More Important & misclassification costs are asymmetrical

- **In most cases it is more important to identify members of one class**
  - Diagnosing illness (Illness)
  - Detecting SPAM mail (Spam mails)
  - Credit default (Potential Defaulter Class)
  - Tax fraud (Fraudulent Tax Class)
  - Response to promotional offer (Respondent Class)
  - Detecting electronic network intrusion (Malicious Packet class)
  - Predicting delayed flights (Delayed flights)

- **In such cases, we are willing to tolerate greater overall error, in return for better identifying the important class for further attention**

- The cost of making a misclassification error may be higher for one class than the other(s)

Asymmetrical Costs – Response to Promotional Offer

Suppose we send an offer to 1000 people, with 1% average response rate ("1" = response, "0" = nonresponse)

- "Naïve rule" (classify everyone as '0') has error rate of 1% (seems good)
- Let's assume that by using some ML model
  - ➢ We can correctly classify eight 1's as 1's
  - ➢ It comes at the cost of misclassifying twenty 0's as 1's and two 1's as 0's.
  - ➢ Error rate = (2+20) = 2.2% (higher than naïve rate)

	Actual 1	Actual 0
Predicted 1	8	20
Predicted 0	2	970

Suppose: Profit from a **'1'** is $10 & Cost of sending offer is $1

- Under naïve rule, all are classified as "0", so no offers are sent: no cost, no profit
- Under ML predictions, 28 offers are sent.
  8 respond with profit of $10 each 20 fail to respond, cost $1 each
  972 receive nothing (no cost, no profit)

	Actual 1	Actual 0
Predicted 1	80$	-20$
Predicted 0	0	0

Net profit = $60

Thus, we need to look beyond the traditional error/accuracy metrics in classification scenarios

Ref: Shmueli et al,Data Mining for Business Analytics: Concepts, Techniques and Applications in Python, Wiley, 2019

Basic terms

True positives (TP): Predicted positive and are actually positive.

False positives (FP): Predicted positive and are actually negative.

True negatives (TN): Predicted negative and are actually negative.

False negatives (FN): Predicted negative and are actually positive.

Alternate Accuracy Measures

		Actual Class	
		C_1	C_2
Predicted Class	C_1	$n_{1,1}$ = number of C_1 records classified correctly as C_1 **True Positive (TP)**	$n_{2,1}$ = number of C_2 records classified incorrectly as C_1 **False Positive (FP)**
	C_2	$n_{1,2}$ = number of C_1 records classified incorrectly as C_2 **False Negative (FN)**	$n_{2,2}$ = number of C_2 records classified correctly as C_2 **True Negative (TN)**

If "C_1" is the important class,

- **Sensitivity (also called "recall)** = % of actual C_1 class correctly classified

$$n_{1,1} / (n_{1,1} + n_{1,2})$$

- Ability of the classifier to detect the important class members correctly.

- Also referred to as **True Positive Rate, TPR** = TP/ (TP + FN)

Alternate Accuracy Measures

		Actual Class	
		C_1	C_2
Predicted Class	C_1	**$n_{1,1}$ = number of C_1 records classified correctly as C_1** **True Positive (TP)**	**$n_{2,1}$ = number of C_2 records classified incorrectly as C_1** **False Positive (FP)**
	C_2	**$n_{1,2}$ = number of C_1 records classified incorrectly as C_2** **False Negative (FN)**	**$n_{2,2}$ = number of C_2 records classified correctly as C_2** **True Negative (TN)**

- If "C_1" is the important class,
- **Specificity** = % of actual C_2 class correctly classified

$$n_{2,2} / (n_{2,1} + n_{2,2})$$

- Ability of the classifier to rule out the other class members (C_2) correctly.
- Also referred to as **True Negative Rate, TNR** = TN / (FP + TN)
- **False Positive Rate (FPR)** = 1- Specificity

$$FPR = FP / (FP + TN)$$

Alternate Accuracy Measures

		Actual Class	
		C_1	C_2
Predicted Class	C_1	$n_{1,1}$ = number of C_1 records classified correctly as C_1 **True Positive (TP)**	$n_{2,1}$ = number of C_2 records classified incorrectly as C_1 **False Positive (FP)**
	C_2	$n_{1,2}$ = number of C_1 records classified incorrectly as C_2 **False Negative (FN)**	$n_{2,2}$ = number of C_2 records classified correctly as C_2 **True Negative (TN)**

If "C_1" is the important class,

- **Precision**= % of predicted C_1 that are actually C_1

  $n_{1,1}$ / ($n_{1,1}$ + $n_{2,1}$)

  TP/ (TP + FP)

- **Recall (also called "sensitivity")** = % of actual C_1 class correctly classified

  $n_{1,1}$ / ($n_{1,1}$ + $n_{1,2}$)

  TP/ (TP + FN)

- F-Measure provides a way to combine both precision and recall into a single measure that captures both properties. Also know as F-Score or F1-Score

  F1-Score= (2*Precision*Recall) /(Precision + Recall)

  - Common metric used on classification problems on imbalanced datasets.

Example 1: Binary classification

We have 2 types of fruits apples and grapes and we want our machine learning model to identify or classify the given fruit as an apple or a grape.

Total no. of samples = 15, no. of apples = 8 and no. of grapes = 7

We will represent apple as 1 and grape as 0 class.

Suppose,

Actual = [1,1,1,1,1,1,1,1,0,0,0,0,0,0,0]

Prediction = [1,0,0,0,1,1,1,1,0,0,0,0,0,1,1]

Then,

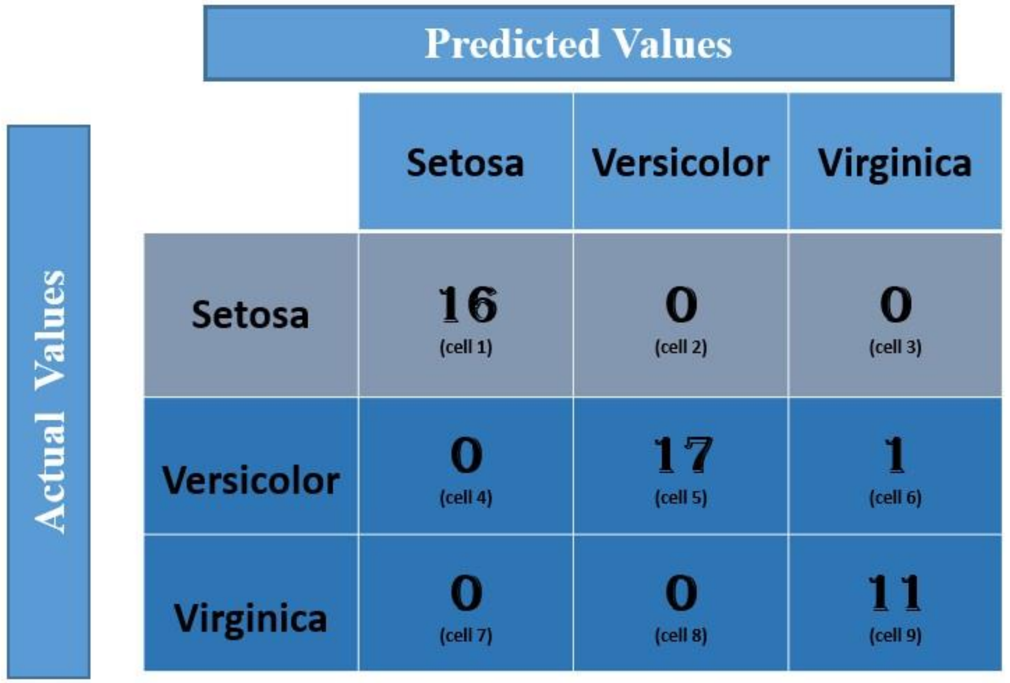
TP=5, FN=3, FP=2, TN=5

Example 2: Multi-class classification – IRIS DATASET.

In the multi-class classification problem, we won’t get TP, TN, FP, FN values directly as in the binary classification problem. We need to calculate for each class.

The dataset has 3 flowers as outputs or classes,

- Versicolor
- Virginica
- Setosa



		Predicted Values		
		Setosa	Versicolor	Virginica
Actual Values	Setosa	16 (cell 1)	0 (cell 2)	0 (cell 3)
	Versicolor	0 (cell 4)	17 (cell 5)	1 (cell 6)
	Virginica	0 (cell 7)	0 (cell 8)	11 (cell 9)

Let us calculate the TP, TN, FP, FN values for the class **Setosa**

TP: The actual value and predicted value should be the same. So concerning Setosa class, the value of cell 1 is the TP value.

FN: The sum of values of corresponding rows except the TP value.

FN = (cell 2 + cell3) = (0 + 0) = 0

FP : The sum of values of corresponding column except the TP value.

FP = (cell 4 + cell 7) = (0 + 0) = 0

TN: The sum of values of all columns and row except the values of that class that we are calculating the values for.

TN = (cell 5 + cell 6 + cell 8 + cell 9) = 17 + 1 +0 + 11 = 29

Now, let us calculate the TP, TN, FP, FN values for the class **Versicolor**

TP: The actual value and predicted value should be the same. So

TP = 17 (cell 5)

FN: The sum of values of corresponding rows except the TP value.

FN = (cell 4 + cell 6) = 0 + 1 = 1

FP: The sum of values of corresponding column except the TP value.

FP = (cell 2 + cell 8) = 0 + 0 = 0

TN: The sum of values of all columns and row except the values of that class that we are calculating the values for.

TN = (cell 1 + cell 3 + cell 7 + cell 9) = 16 + 0 + 0 + 11 = 27

Similarly we can calculate the TP, TN, FP, FN values for the class **Virginica**

TP: The actual value and predicted value should be the same. So

TP = 11 (cell 9)

FN: The sum of values of corresponding rows except the TP value.

FN = (cell 7 + cell 8) = (0 + 0) = 0

FP: The sum of values of corresponding column except the TP value.

FP = (cell 3 + cell 6) = (0 + 1) = 1

TN: The sum of values of all columns and row except the values of that class that we are calculating the values for.

TN = (cell 1 + cell 2 + cell 4 + cell 5) = 16 + 0 +0 + 17 = 33

Accuracy

Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

$$Accuracy = \frac{No. of\ correct\ predictions}{Total\ no. of\ predictions}$$

Can you calculate the accuracies for the above systems?

For binary classification,

$$Accuracy = \frac{10}{15}$$

Accuracy for a multiclass classifier is calculated as the average accuracy per class.

$$Accuracy_{Setosa} = \frac{45}{45} \qquad Accuracy_{Versicolor} = \frac{44}{45}$$

$$Accuracy_{Virginica} = \frac{44}{45}$$

$$Accuracy_{Total} = \frac{1}{3} * \left(\frac{45}{45} + \frac{44}{45} + \frac{44}{45}\right)$$

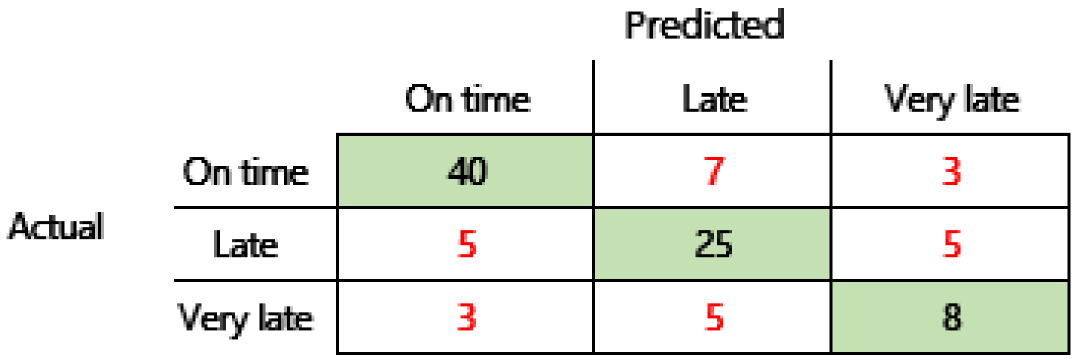
Performance evaluation

Evaluate the performance of a binary classifier from the below confusion matrix using Accuracy, Precision, Recall, Specificity and F1 Score.

	Predicted: 0	**Predicted: 1**
True: 0	**TN** = 126	**FP** = 13
True: 1	**FN** = 24	**TP** = 60

Performance evaluation

Evaluate the performance of a multiclass classifier from the below confusion matrix using Accuracy, Precision, Recall, Specificity and F1 Score.



		Predicted		
		On time	Late	Very late
Actual	On time	40	7	3
	Late	5	25	5
	Very late	3	5	8

Suppose 10,000 patients get tested for cancer; out of them, 8,200 are healthy and 1,800 are sick. For the sick people, a test was positive for 1,020 and negative for 780. For the healthy people, the same test was positive for 280 and negative for 7,920. Construct a confusion matrix for the data. Compute the precision, recall and F1 score.

Alternate Accuracy Measures

		Actual Class	
		C_1	C_2
Predicted Class	C_1	$n_{1,1}$ = number of C_1 records classified correctly as C_1 **True Positive (TP)**	$n_{2,1}$ = number of C_2 records classified incorrectly as C_1 **False Positive (FP)**
	C_2	$n_{1,2}$ = number of C_1 records classified incorrectly as C_2 **False Negative (FN)**	$n_{2,2}$ = number of C_2 records classified correctly as C_2 **True Negative (TN)**

- If "C_1" is the important class,
- **Sensitivity (also called "recall)** = % of actual C_1 class correctly classified

$$n_{1,1} / (n_{1,1} + n_{1,2})$$

True Positive Rate, TPR = TP/ (TP + FN)

- **Specificity** = % of actual C_2 class correctly classified

$$n_{2,2} / (n_{2,1} + n_{2,2})$$

True Negative Rate, TNR = TN / (FP + TN)

- **False Positive Rate (FPR)** = 1- Specificity

FPR = FP / (FP + TN)

ROC Curves (Receiver Operating Characteristic Curves) for a Perfect Classifier

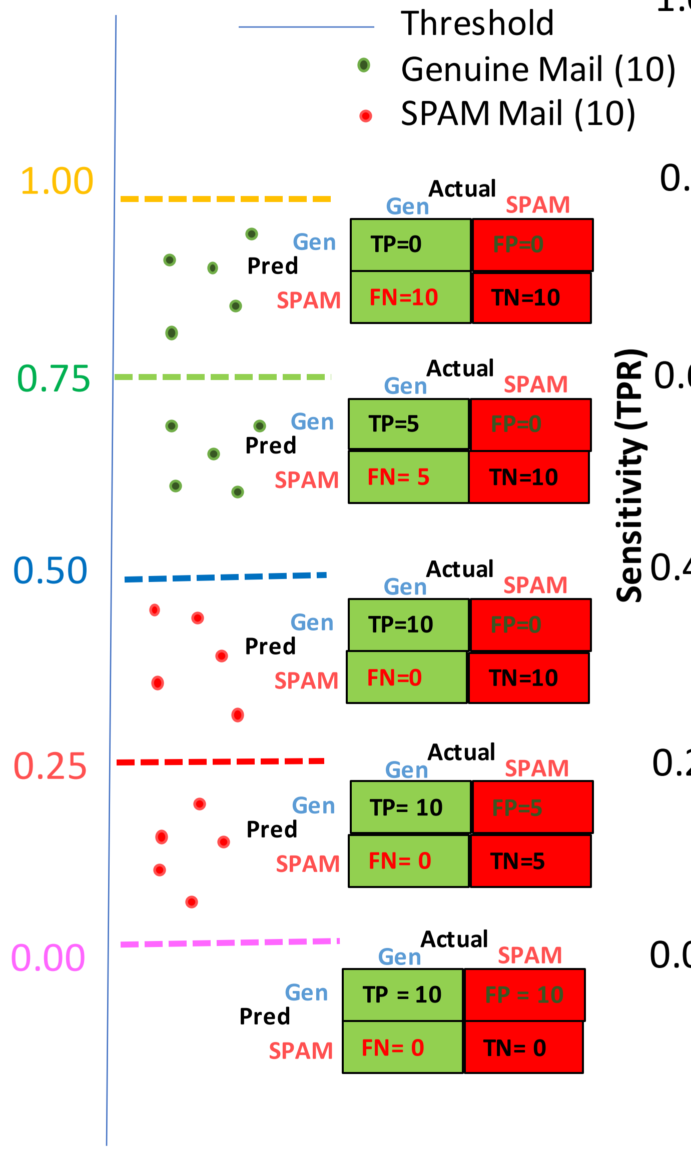
When threshold is near 1,
Sensitivity = TP/(TP+FN) = 0/10 = 0
Specificity = TN/(TN+FP) = 10/10= 1
Hence (0,0)

When threshold is near .75,
Sensitivity = TP/(TP+FN) = 5/10 = 0.5
Specificity = TN/(TN+FP) = 10/10= 1
Hence (0,0.5)

When threshold is near 0.5,
Sensitivity = TP/(TP+FN) = 10/10 = 1
Specificity = TN/(TN+FP) = 10/10= 1
Hence (0,1)

When threshold is near 0.25,
Sensitivity = TP/(TP+FN) = 10/10 = 1
Specificity = TN/(TN+FP) = 5/ 10 = 0.5
Hence (0.5,1)

When threshold is near 0.0,
Sensitivity = TP/(TP+FN) = 10/10 = 1
Specificity = TN/(TN+FP)= 0/10 = 0
Hence (1,1)



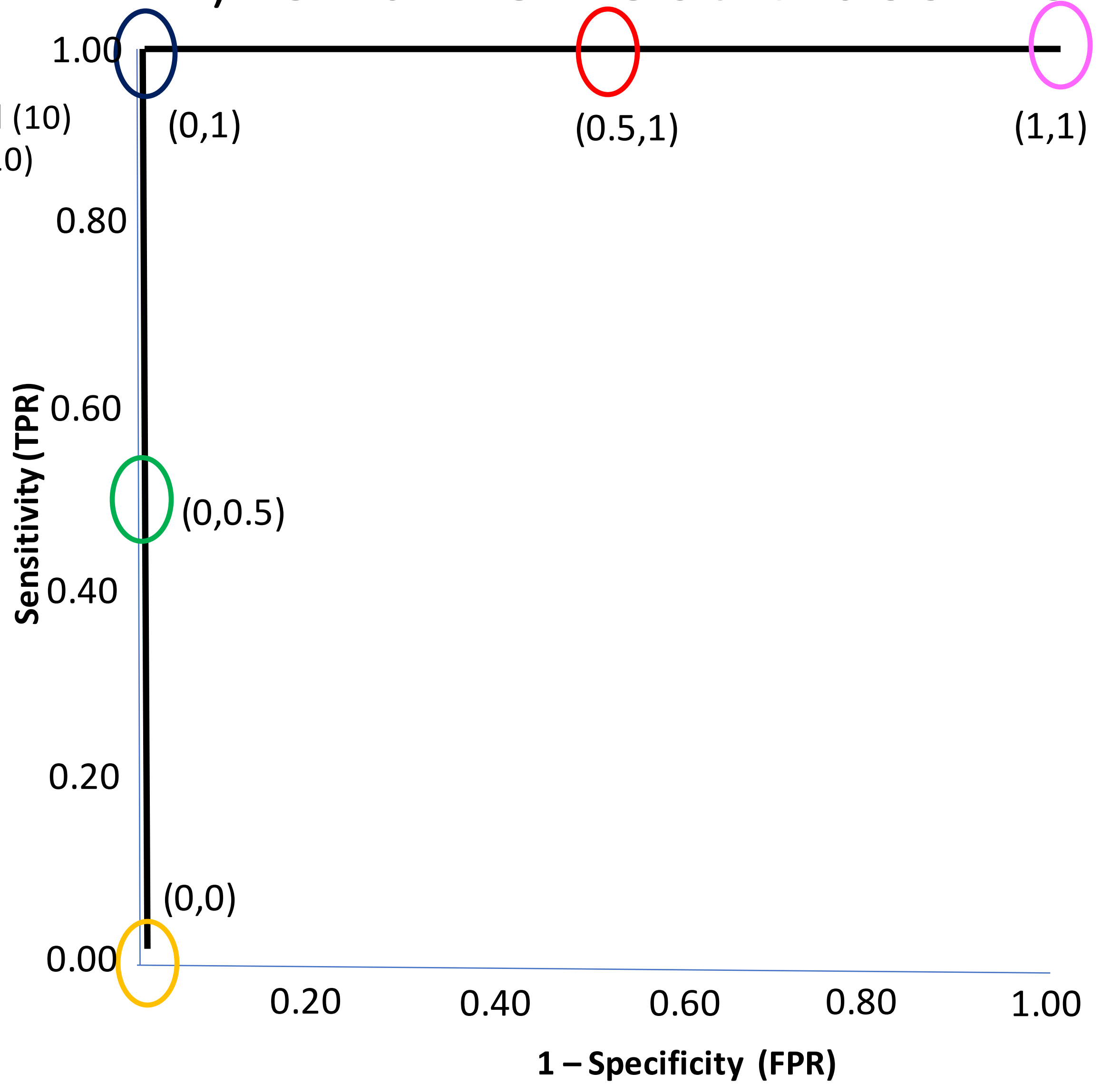
Pred \ Actual	Gen	SPAM
Gen	TP=0	FP=0
SPAM	FN=10	TN=10

Pred \ Actual	Gen	SPAM
Gen	TP=5	FP=0
SPAM	FN= 5	TN=10

Pred \ Actual	Gen	SPAM
Gen	TP=10	FP=0
SPAM	FN=0	TN=10

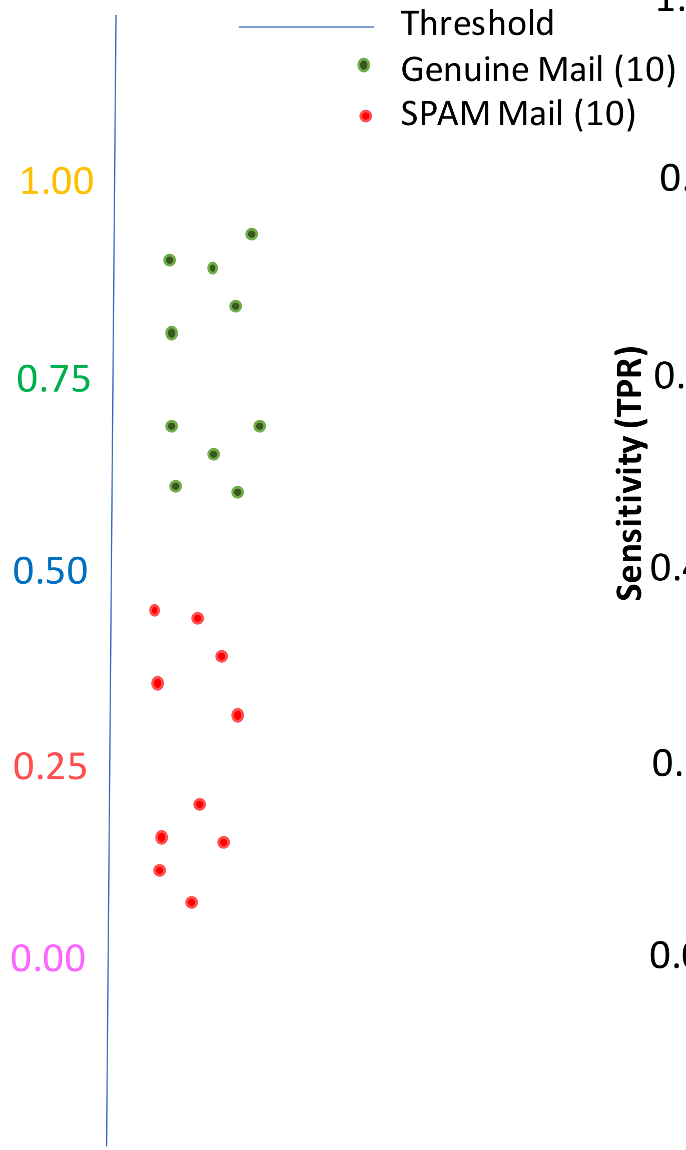
Pred \ Actual	Gen	SPAM
Gen	TP= 10	FP=5
SPAM	FN= 0	TN=5

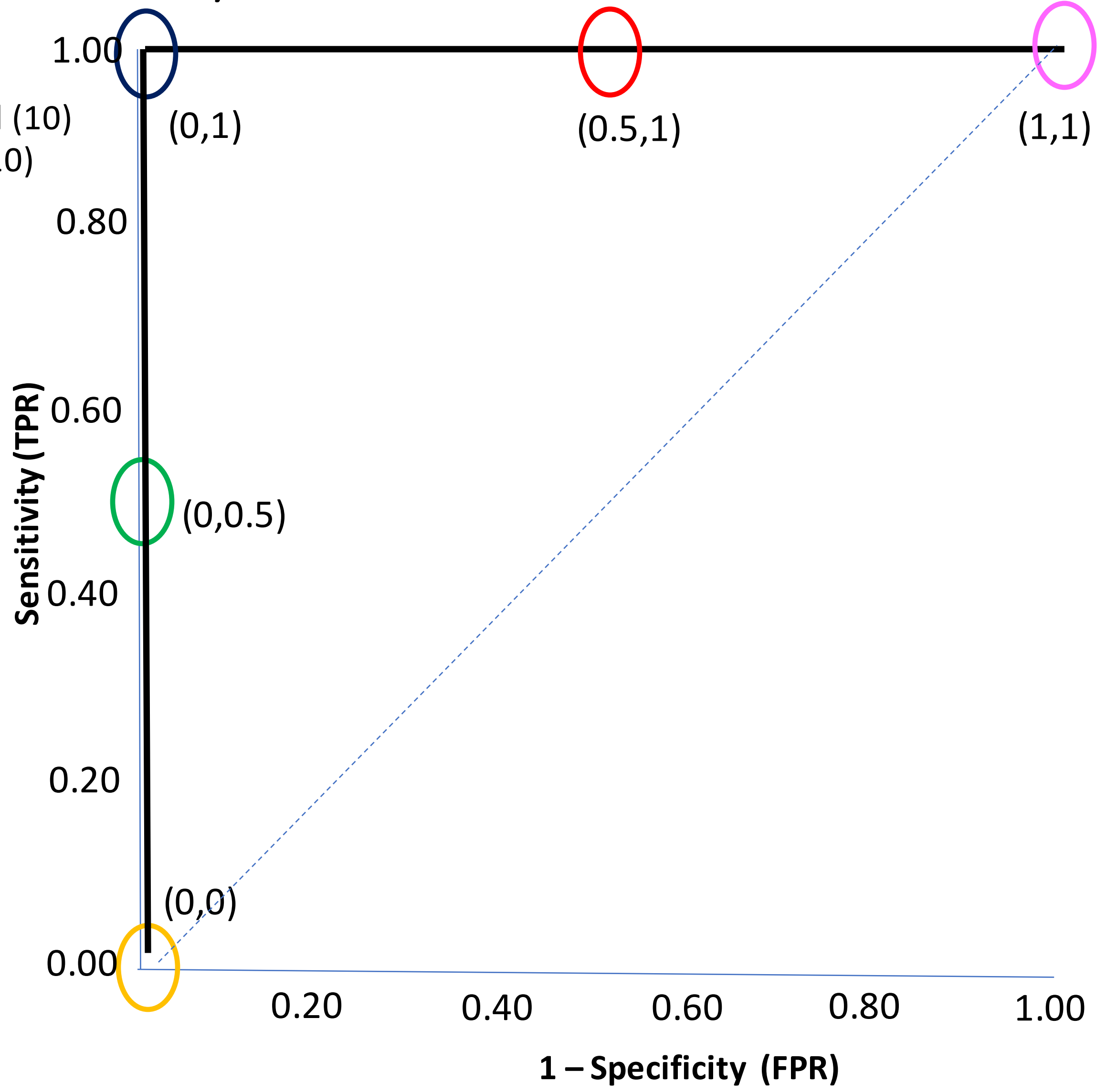
Pred \ Actual	Gen	SPAM
Gen	TP = 10	FP = 10
SPAM	FN= 0	TN= 0



ROC Curves (Receiver Operating Characteristic Curves) for a Perfect Classifier

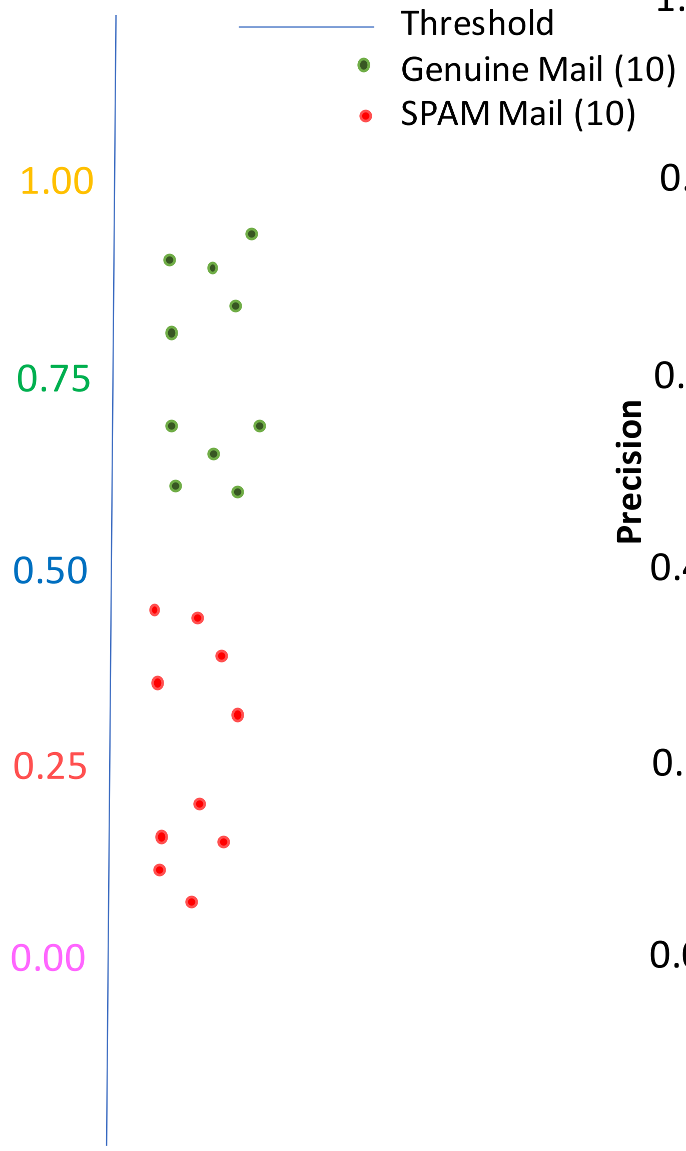
- ROC curve is an evaluation metric for binary classification problems.
- Plots the Sensitivity (**TPR**) against 1 – Specificity (**FPR**) for **varying** threshold values
- **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve
- The goal in ROC space is to be in the ***upper-left-hand corner*** (0, 1), i.e. zero FP's (*FPR=0*) ; Sensitivity(Recall)=1, i.e. all the positives classified as positives
- The diagonal (the "*curve*" for a naïve classifier), on an average, when drawing randomly scores on the unit interval or if our predictions are all 0 or all 1.

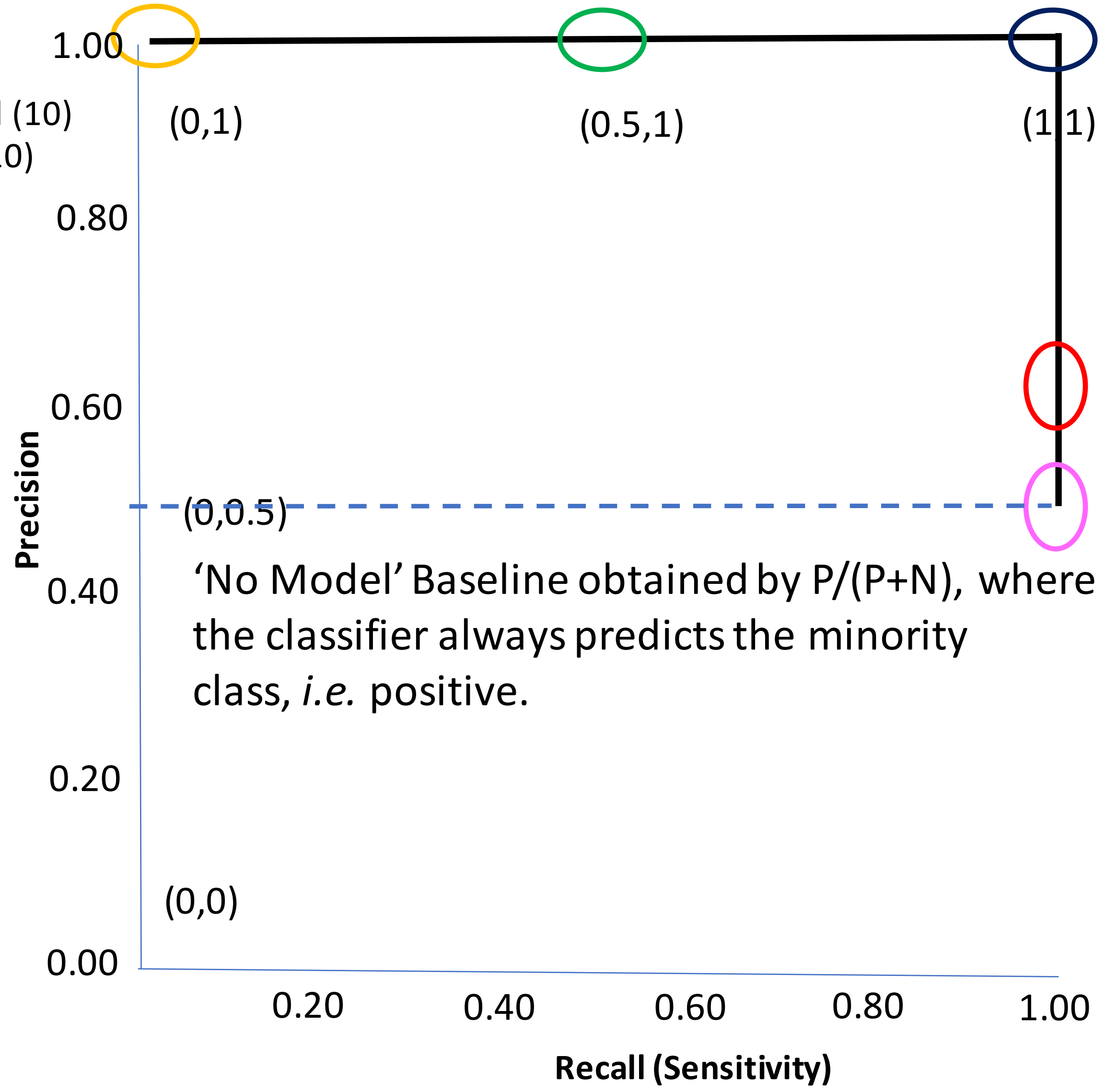




Precision Recall Curves for a Perfect Classifier

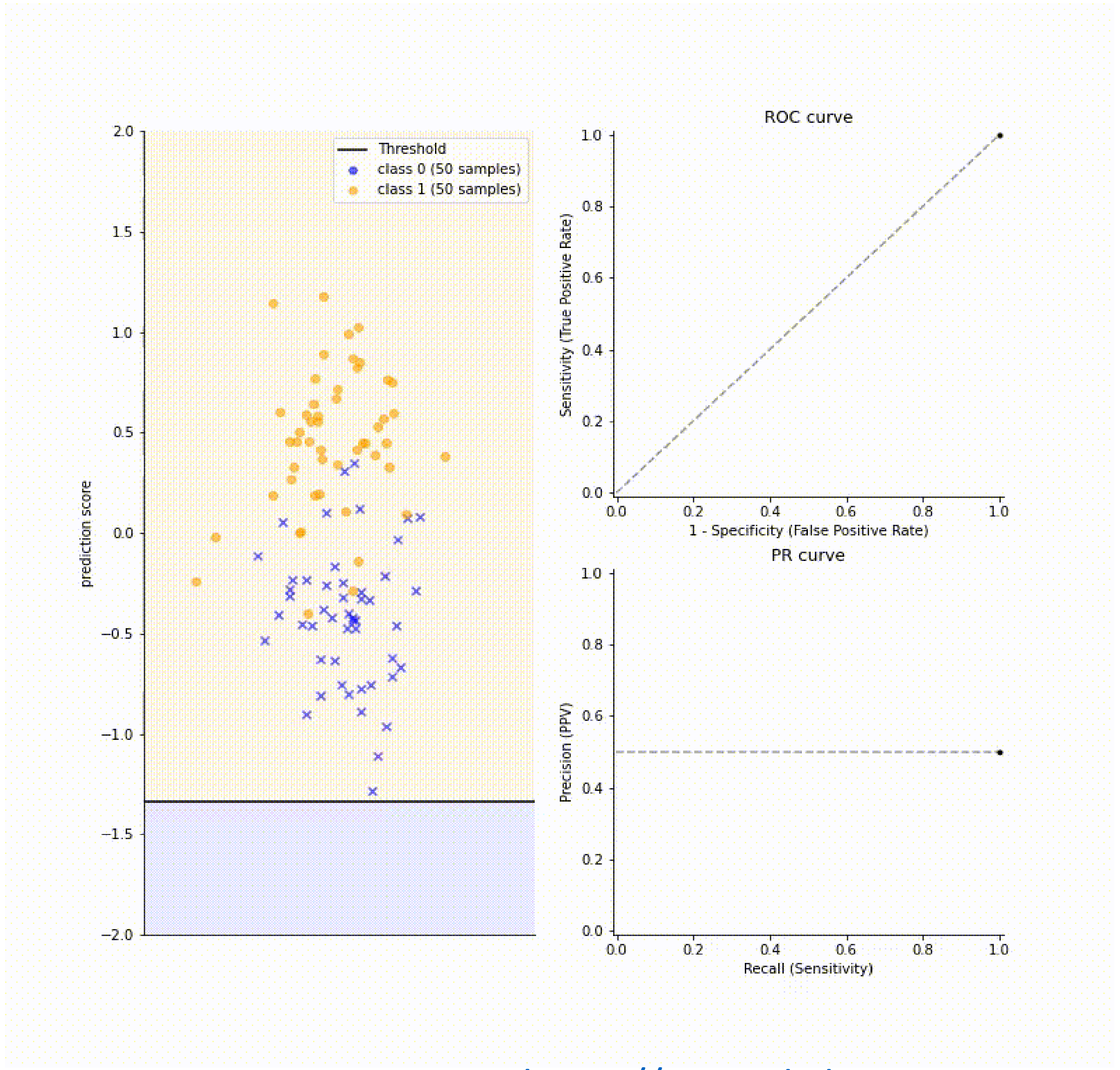
- Performance of the positive class (minority) when dealing with imbalanced classes.
- We are less interested in the skill of the model at predicting class 0 correctly, e.g. high true negatives. Key to the calculation of precision and recall is that the calculations do not make use of the true negatives. It is only concerned with the correct prediction of the minority class
- In the PR space, the goal is to be in the upper-right-hand corner — the top right corner (1, 1), all positives are classified as positive (*Recall=1*) and everything we are classifying as positive is true positive (*Precision=1*), ie zero False Positives.





'No Model' Baseline obtained by P/(P+N), where the classifier always predicts the minority class, *i.e.* positive.

ROC Curve for Imperfect classifier

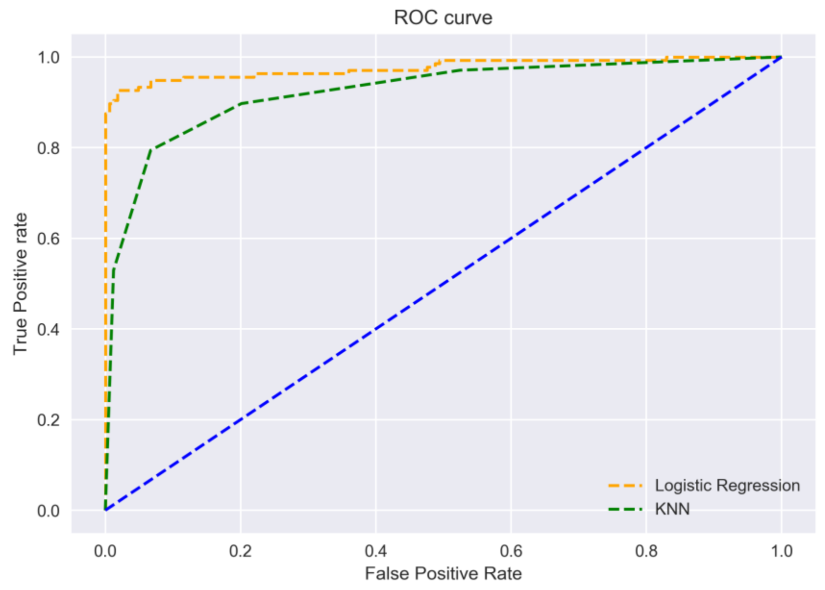


- By design the ROC curve typically develops in a U-shape
- **Step Left**: Turn a False Positive (blue cross) into a True Negative (blue dot). This behavior represents a correct decision and thus it reduces the False Positive Rate (*x-axis*) (mostly from -1.5 to -0.5)
- **Step downwards:** Turn a True Positive (orange dot) into a False Negative (orange cross). This behavior represents a wrong decision and thus it reduces the Recall (*y-axis*). (mostly from 0.5 to 1.5)

Image Source: https://towardsdatascience.com/on-roc-and-precision-recall-curves-c23e9b63820c

ROC curves for comparing models – Area Under the Curve

- The curves of different models can be compared directly in general or for different thresholds.
- As it can be challenging to compare two or more classifiers based on their curves, the Area Under the Curve (AUC) can be used as a summary of the model skill



- AUC for the Logistic Regression ROC curve is higher than that for the KNN ROC curve.
- Here Logistic regression performs better in classifying the positive class in this dataset

- PR AUC can be used for compare classification models on imbalanced datasets

Image Source: https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/