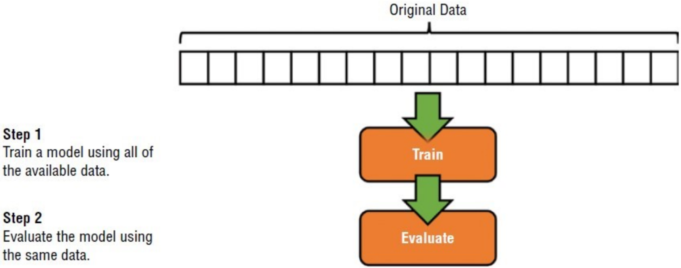
Machine learning

Estimating future performance



Step 1
Train a model using all of the available data.

Step 2
Evaluate the model using the same data.

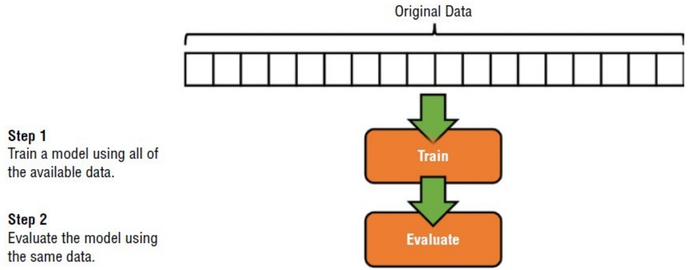
Estimating future performance

During the model building process, the goal is to use the observed data to develop a model that best estimates the relationship between **a set of predictor variables** x and **corresponding response values** y **.**

The degree to which the model explains the relationship between x and y is known as **goodness-of-fit**.

To evaluate how well the model fits against the data, we quantify the difference between the model's predicted response values y and the observed response values $\hat{y}$. The difference between the model's predicted response and the observed response values for the data from which a model is built is known as the **resubstitution error.**

Estimating future performance



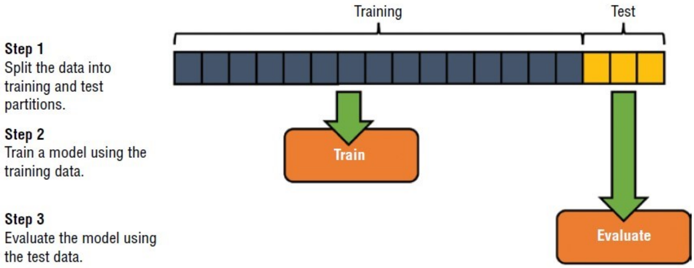
Step 1
Train a model using all of the available data.

Step 2
Evaluate the model using the same data.

While **the resubstitution error provides an assessment** of how well a model estimates the relationship between the predictors and response variables within a dataset, it does not provide useful insight into **how well the model will perform in the future against new data.**

The problem is that we're testing the model on data that it has already seen!!!.

Holdout method



Step 1
Split the data into training and test partitions.

Step 2
Train a model using the training data.

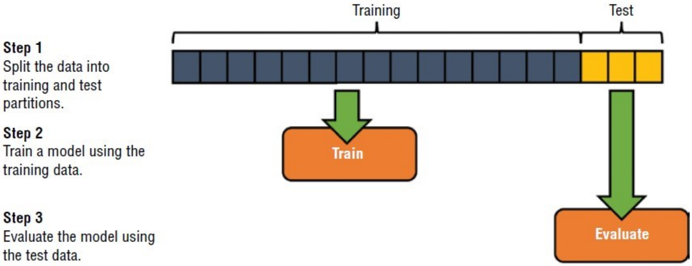
Step 3
Evaluate the model using the test data.

Instead of using our entire dataset to train and evaluate our model, we split the original data into two partitions

- we use one partition **(training data)** to build our model
- we use the other partition **(test data)** to evaluate how well our model will perform against previously unseen data.

This approach is known as the holdout method.

Holdout method



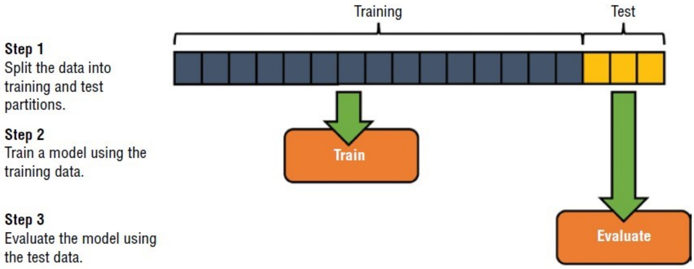
Step 1
Split the data into training and test partitions.

Step 2
Train a model using the training data.

Step 3
Evaluate the model using the test data.

The holdout method, 30% to 70% of the original data is held out for testing, while the remainder is used to train the model. However, depending on how much data is available, these proportions may vary.

Holdout method



Step 1
Split the data into training and test partitions.

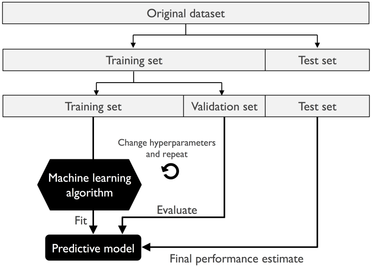
Step 2
Train a model using the training data.

Step 3
Evaluate the model using the test data.

There are two important principles to keep in mind with the holdout method.

- when creating the training and test partitions, it is important that both datasets be independent of each other;
- they must be representative samples of the original data.

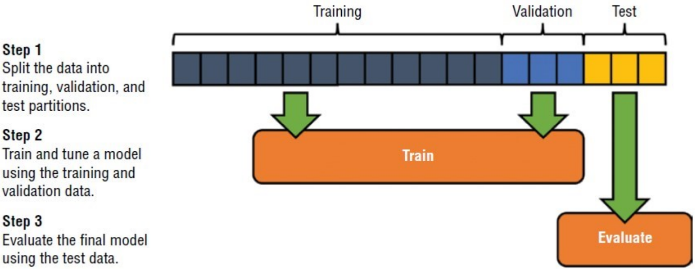
Model selection,



- In machine learning applications, we are also interested in **tuning hyperparameters** and **comparing** different parameter settings to further improve the performance for making predictions on unseen data.
- This process is called **model selection**.

Model tuning is finding the optimal hyperparameter values.

Validation set



Step 1
Split the data into training, validation, and test partitions.

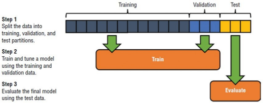
Step 2
Train and tune a model using the training and validation data.

Step 3
Evaluate the final model using the test data.

If in the process of building a model, we need to tune the model parameters many times, it is better to separate a part of the data, which we call **the validation set.**

In practice, it is common for the split between the training, validation, and test sets to be 50:25:25, respectively, and that each partition be independent of each other.

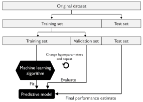
Validation set



Step 1
Split the data into training, validation, and test partitions.

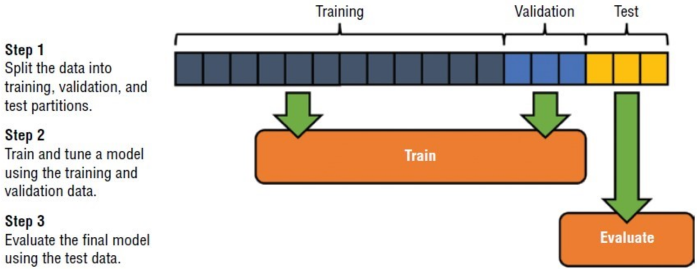
Step 2
Train and tune a model using the training and validation data.

Step 3
Evaluate the final model using the test data.



- In situations where we have a lot of available data, we can use half of the original data (50 percent) to train a model.
- We then use a separate 25 percent of the original data to evaluate the performance of the model.
- We repeat the process of training and validation several times with the same training and validation datasets to create several models based on different parameters.
- Once we decide on a final model, we then use the remaining 25 percent of the original data (the test data) that our model has not yet seen to estimate the future performance of the model.

Validation set


Training
Validation
Test
Step 1
Split the data into training, validation, and test partitions.
Step 2
Train and tune a model using the training and validation data.
Step 3
Evaluate the final model using the test data.
Train
Evaluate

The problem with this approach is that when we don't have a large amount of data to work with, all or some of our data partitions may not be adequately representative of the original dataset.

Resampling

Resampling is a technique involves repeatedly using different samples of the original data to train and validate a model.

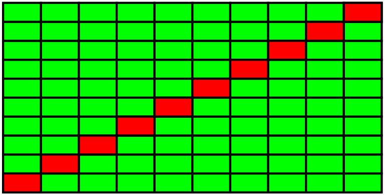
At the end of the process, the performance of the model across the different iterations is averaged to yield an overall performance estimate for the model.

The most common approaches to this resampling technique is known as **cross-validation**.

k-Fold Cross-Validation

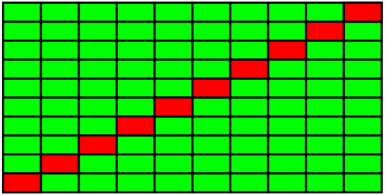
The most commonly used of all the approaches to cross-validation, is **k-fold cross-validation.**

- after the test data has been sequestered, the remaining data is divided into k completely separate random partitions of approximately equal size.
- these partitions are known as **folds** $S_1, S_2, ..., S_k$ (usually k =5,10);



k-Fold Cross-Validation

- for the first iteration, all instances labeled as **fold1** are held out, while the remainder of the data is used to train the model. The performance of the model is then evaluated against the unseen data (fold1).
- for the second iteration, the instances labeled as **fold2** are held out as the validation data, while the remaining instances are used to train the model.
- this process is then repeated three more times using each of the remaining folds.
- during each of the k iterations, a different validation set is used, and by the end of the 10th iteration, all of the instances in the dataset will have been used for both training and validation.



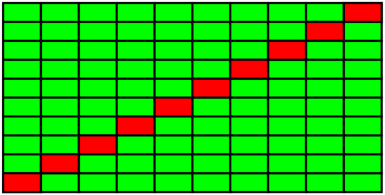
k-Fold Cross-Validation

- this process results in k estimates of the model's performance. The k-fold cross-validation estimate is computed as **the average of the k estimates**.

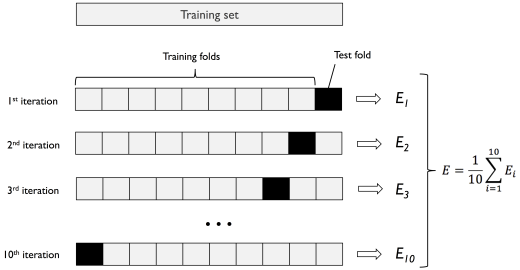
$$E = \frac{1}{k}\sum_{i=1}^{k} E_i$$

where

  - k number of folds,
  - E_i, $i = 1, ..., k$ parameter.

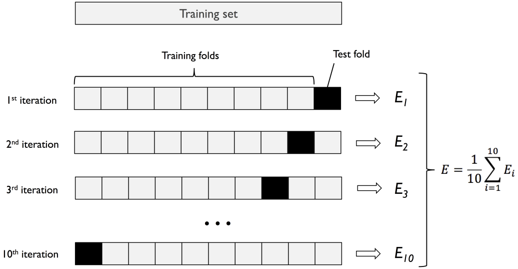


k-Fold Cross-Validation



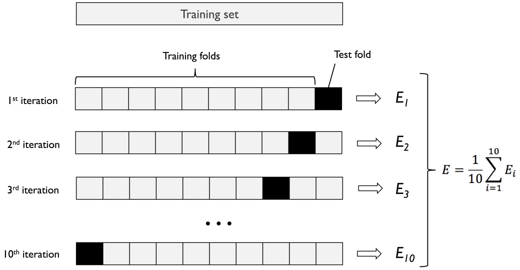
- The obtained estimated average performance of the models, based on the different, independent test folds, **is less sensitive to the sub-partitioning of the training data compared to the holdout method.**
- We use k-fold cross-validation for model tuning that yield a satisfying generalization performance, which is estimated from evaluating the model performance on the test folds.

k-Fold Cross-Validation



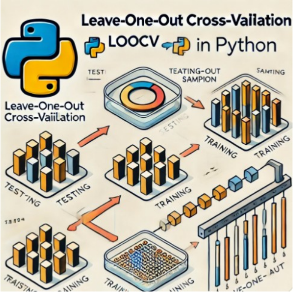
- Once we have found satisfactory hyperparameter values, we can retrain the model on the complete **training dataset** and obtain a final performance estimate using the independent test dataset.
- The rationale behind fitting a model to the whole training dataset after k-fold cross-validation is that
  - we are typically interested in a single, final model (versus k individual models),
  - providing more training examples to a learning algorithm usually results in a more accurate and robust model.

k-Fold Cross-Validation



- Since k-fold cross-validation is a resampling technique without replacement, the advantage of this approach is that **in each iteration, each example will be used exactly once, and the training and test folds are disjoint.** Furthermore, all test folds are disjoint; that is, **there is no overlap between the test folds.**

Leave-one-out cross-validation method (LOOCV)



- A special case of k-fold cross-validation is the **leave-one-out cross-validation (LOOCV)** method.
- In LOOCV, we set the number of folds equal to the number of training examples ($k = n$) so that only one training example is used for testing during each iteration.

Leave-one-out cross-validation method (LOOCV) - advantages

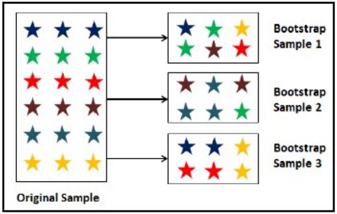
- The greatest amount of data is used each time we train the model - this helps with the accuracy of the model.
- The approach is deterministic. This means that the performance of the model will be the same every time the process is executed.
- We are training the model on every possible combination of observations.

Leave-one-out cross-validation method (LOOCV) - disadvantage

This approach is the high computational cost. Since the approach requires that a model be trained and validated n times, this can become rather expensive or infeasible with complex models and large datasets.

Bootstrap Sampling

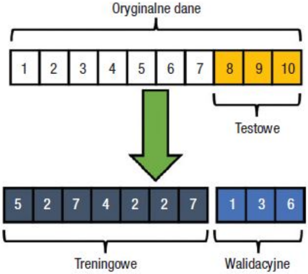
The second resampling technique is known as bootstrap sampling or bootstrapping.



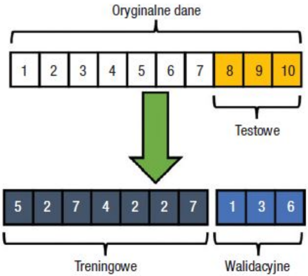
- It performs a sampling iteratively from the dataset with replacement.
- Sampling with replacement will make random selections.
- It requires the size of the sample and the number of iterations.
- In each iteration, it uniformly selects the observations.
- Each record has equal chances of being selected again.
- The samples that are not selected are known as "out-of-bag" samples.

Bootstrap Sampling

- The basic idea behind bootstrap sampling is to create a training dataset from the original data using a random sampling with replacement approach.
- A version of this technique is known as the **0.632 bootstrap**.
- It involves random sampling a dataset with n instances, n different times with replacement, to create another dataset also with n instances.
- This new dataset is used for training, while the instances from the original data, which were not selected as part of the training data, are used for validation.



Bootstrap Sampling



The 0.632 bootstrap gets its name from the fact that when sampling with replacement, the probability that a particular example will be selected as part of the training set is 63.2 percent.

Measuring performance for classification

		predicted	
		1	0
observed	1	**TP**	**FN**
	0	**FP**	**TN**

Some examples of positive classes

- spam, in anti-spam lists;
- the selected character, in OCR;
- presence of the virus, in medical tests.

The use of the terms positive and negative is not intended to imply any value judgment (that is, good versus bad), nor does it necessarily suggest that the outcome is present or absent (such as birth defect versus none). The choice of the positive outcome can even be arbitrary, as in cases where a model is predicting categories such as sunny versus rainy, or dog versus cat.

Binary confusion matrix

		actual	
		1	0
predicted	1	**TP**	**FP**
	0	**FN**	**TN**

- **TP (True Positives)** – are when you predict an observation belongs to a class and it actually does belong to that class.;
- **FN (False Negatives)** – occur when you predict an observation does not belong to a class when in fact it does;
- **FP (False Positives)** – occur when you predict an observation belongs to a class when in reality it does not;
- **TN (True Negatives)** – are when you predict an observation does not belong to a class and it actually does not belong to that class.

Evaluation metric

		actual	
		1	0
predicted	1	**TP**	**FP**
	0	**FN**	**TN**

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.

For binary clas.

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

Accuracy alone is not too informative regarding a model's effectiveness.

Evaluation metric

		actual	
		1	0
predicted	1	**TP**	**FP**
	0	**FN**	**TN**

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

For binary clas.

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

Model evaluation - measures of model quality

		predicted	
		1	0
observed	1	**TP**	**FN**
	0	**FP**	**TN**

Other measures of classifier quality can also be calculated from the confusion matrix:

- **measures of precision (PPV and NPV)**; (related to the classifier);
- **coverage (sensitivity and specificity)**; (related to the test set);
- F1 score (also known as F-measure, or balanced F-score).

The above measures apply to well-set classes.

Model evaluation - measures of model quality

		predicted	
		sick	healthy
observed	sick	**TP**	**FN**
	healthy	**FP**	**TN**

- **positive predictive value (precision)** - measures the proportion of correct positive classifications relative to all positive classifications, i.e. it answers the question: **How many of the positively classified cases were well classified?** or the indicator answers the question: **If a test result is positive, what is the probability that the person tested is ill?**

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

- **negative predictive value (precision)** - the indicator answers the question: **If the test result is negative, what is the probability that the person tested is healthy?**

$$NPV = \frac{TN}{TN + FN}.$$

Model evaluation - measures of the quality of the model

		predicted	
		sick	healthy
observed	sick	**TP**	**FN**
	healthy	**FP**	**TN**

- **recall** - measures the proportion of correct classifications relative to all positive cases, i.e. answers the question: **What proportion of positive results did the classifier detect?** or the indicator answers the question: **What is the probability that a test performed for an ill person will show that he or she is ill?**

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

- **specifity (recall)** measures the proportion of correct negative classifications relative to all negative cases, i.e. answers the question: **What proportion of negative results did the classifier detect?** or the indicator answers the question: **What is the probability that for a healthy person the test will not detect the disease?**

$$\text{specifity}\, TNR = \frac{TN}{TN + FP}.$$

Model evaluation - measures of the quality of the model - F-score

$$\begin{array}{cccc} & & \multicolumn{2}{c}{\textit{predicted}} \\ & & 1 & 0 \\ \textit{observed} & 1 & \mathbf{TP} & \mathbf{FN} \\ & 0 & \mathbf{FP} & \mathbf{TN} \end{array}, \; PPV = \frac{TP}{TP+FP}, \; TPV = \frac{TP}{TP+FN}$$

- **F-Score** allows two classifiers to be compared. Calculating all the measures presented, for two classifiers, may not give an answer as to which is better, as it is extremely difficult to compare several measures between each other at the same time.
- The harmonic mean is used to average the above parameters (it is better than the arithmetic mean).
- The range of F-measure values is [0, 1], where 1 denotes the perfect model and a classifier with a larger F value is better.

Model evaluation - measures of the quality of the model - F-score

$$\begin{array}{cccc} & & \multicolumn{2}{c}{\textit{predicted}} \\ & & 1 & 0 \\ \textit{observed} & 1 & \mathbf{TP} & \mathbf{FN} \\ & 0 & \mathbf{FP} & \mathbf{TN} \end{array}, \; PPV = \frac{TP}{TP+FP}, \; TPV = \frac{TP}{TP+FN}$$

- **F-Score** is the harmonic mean of the precision and sensitivity

$$\begin{aligned} F1 &= \left(\frac{\text{precision} PPV^{-1} + \text{recall} \, TPR^{-1}}{2} \right)^{-1} = 2 \frac{PPV * TPR}{PPV + TPR} \\ &= \frac{2TP}{2TP + FP + FN} \end{aligned}$$

The F-measure evenly considers precision and recall.

Class Imbalance Problem

Lots of classification problems where the classes are skewed (more records from one class than another)

- credit card fraud;
- intrusion detection;
- detection of sick people in the general human population.

The more imbalanced the data is, the more likely that such a classifier would have high accuracy by simply guessing the label of the majority class most of the time.

The kappa statistic

The kappa statistic adjusts accuracy by accounting for the possibility of a correct prediction by chance alone.

Kappa values range from zero to a maximum of one, which indicates perfect agreement between the model's predictions and the true values.

- Poor agreement = less than 0.20
- Fair agreement = 0.20 to 0.40
- Moderate agreement = 0.40 to 0.60
- Good agreement = 0.60 to 0.80
- Very good agreement = 0.80 to 1.00

The kappa statistic

Kappa can be thought of as an adjustment to predictive accuracy by accounting for the possibility of a correct prediction by chance alone.

$$\kappa = \frac{ACC - p_e}{1 - p_e}$$

- p_e - probability of expected or chance agreement between the predicted values and the actual values under the assumption that the predictions were made at random.

$$\kappa = \frac{(FN + TN)(FP + TN)}{all^2} + \frac{(FN + TP)(FP + TP)}{all^2}$$

- ACC - accuracy of the model.

Accuracy and Kappa

Accuracy and Kappa are the default metrics used to evaluate algorithms on binary and multiclass classification datasets.

Accuracy is the percentage of correctly classified instances out of all instances.

Kappa is like classification accuracy, except that it is normalized at the baseline of random chance on dataset. It is a more useful measure to use on problems that have an imbalance in the classes (e.g. a 70% to 30% split for classes 0 and 1)

Propensities

- During the classification process, algorithms actually estimate **the probability that an individual instance belongs to a particular class.**
- These probabilities are also known as **propensities.**
- The propensity of an instance belonging to a particular class is compared against **a threshold or cutoff value**, which was set either by the algorithm or by a user.
- If the probability of belonging to the class in question is higher than the cutoff value, then the instance is assigned to that class. For most classification algorithms, the default two-class cutoff is 0.5. It is possible to use a cutoff value that is either greater than or less than 0.5.

Visualizing model performance

The **ROC (Receiver Operating Characteristics)** curve s commonly used to visually represent the relationship between a model's true positive rate (TPR) and false positive rate (FPR) for all possible cutoff values.

$$1 - TNR = 1 - \frac{TN}{TN + FP} = \frac{FP}{TN + FP} = FPR \text{ - } \textbf{false positive rate}$$

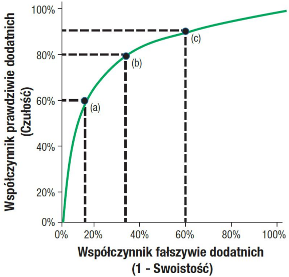
$$\begin{array}{lccc} & & \multicolumn{2}{c}{\textit{predicted}} \\ & & 1 & 0 \\ \textit{observed} & 1 & \mathbf{TP} & \mathbf{FN} \\ & 0 & \mathbf{FP} & \mathbf{TN} \end{array}$$

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

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

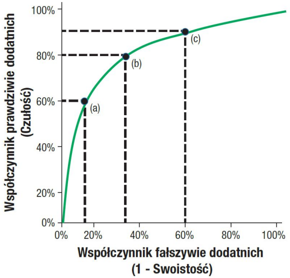
ROC curve

The ROC curve provides us with insight into the classifier's performance at various cutoff thresholds.



- at threshold (a), we see that the classifier's TPR is at 60 percent, while its FPR is at 15 percent. This means that at this threshold, the model is able to correctly classify 60 percent of the observations (the positive class) while misclassifying 15 percent of the observations (the negative class).

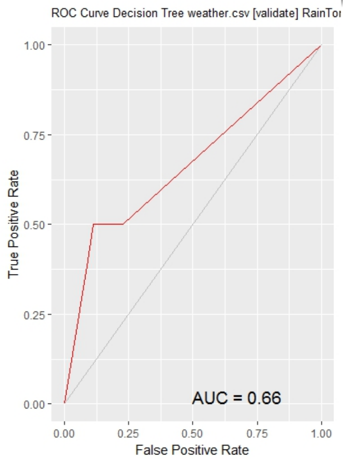
ROC curve



- as we progress up the curve to thresholds (b) and (c), we see that as the classifier's ability to correctly identify the positive class improves, so does its misclassification rate for the negative class.
- at threshold (c), the TPR for the classifier is now at 90 percent, and the FPR has also increased to 60 percent.

This illustrates **the inherent trade-off** that exists between a classifier's ability to correctly identify the positive classes (sensitivity) while also correctly identifying the negative classes (specificity).

The shape of an ROC curve



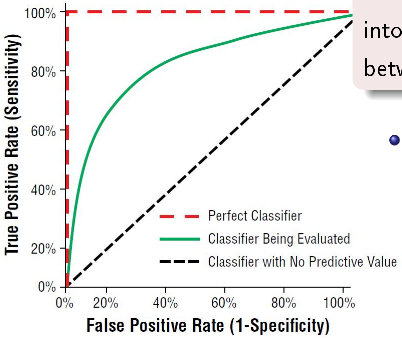
The following points are characteristic of ROC space

- **(0, 0)** – declare everything to be negative class, FP=0 and TP=0;
- **(1, 1)** – declare everything to be positive class FN=0 and TN=0;
- **(0, 1)** – ideal FP=0 i FN=0;
- on diagonal **FPR=TPR** – stands for a random guessing strategy (classifier with no predictive value).

		predicted	
		1	0
observed	1	**TP**	**FN**
	0	**FP**	**TN**

$$FPR = \frac{FP}{TN + FP} \qquad TPR = \frac{TP}{TP + FN}$$

The shape of an ROC curve



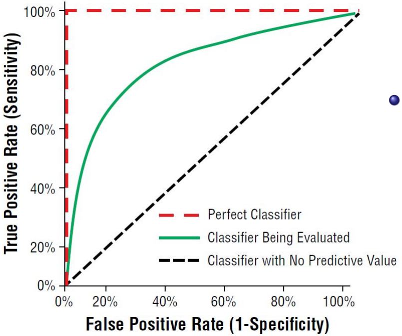
The shape of an ROC curve provides insight into a classifier's ability to discriminate between the positive and negative classes.

- The classifier represented by the black dotted line is a classifier with no predictive value. This classifier identifies positive and negative examples within the evaluation dataset at the same rate regardless of the cutoff threshold. It performs no better than chance.

		predicted	
		1	0
observed	1	TP	FN
	0	FP	TN

$$FPR = \frac{FP}{TN + FP} \qquad TPR = \frac{TP}{TP + FN}$$

The shape of an ROC curve

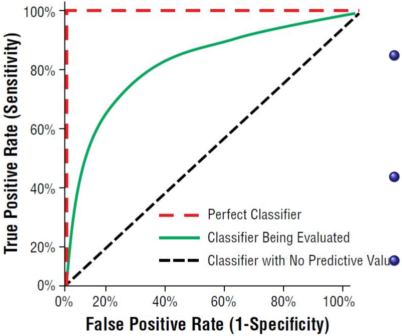


- The classifier represented by the red dotted line is an ideal classifier. It is able to identify all of the positive examples while not misclassifying any of the negative examples.

		predicted	
		1	0
observed	1	**TP**	**FN**
	0	**FP**	**TN**

$$FPR = \frac{FP}{TN + FP} \qquad TPR = \frac{TP}{TP + FN}$$

The shape of an ROC curve

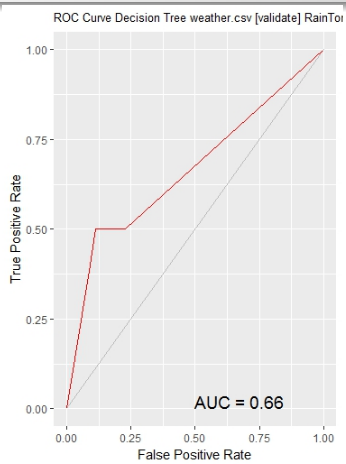


- In practice, most classifiers fall somewhere between both extremes, as represented by the **green ROC curve**.
- The closer a classifier's ROC curve is to the red line, the better it is.
- The closer a classifier's ROC curve is to the **black line**, the worse it is.

		predicted	
		1	0
observed	1	**TP**	**FN**
	0	**FP**	**TN**

$$FPR = \frac{FP}{TN + FP} \qquad TPR = \frac{TP}{TP + FN}$$

AUC



AUC (Area Under ROC Curve) measures the area under the ROC curve. The AUC treats the ROC diagram as a two-dimensional square and measures the total area under the ROC curve.

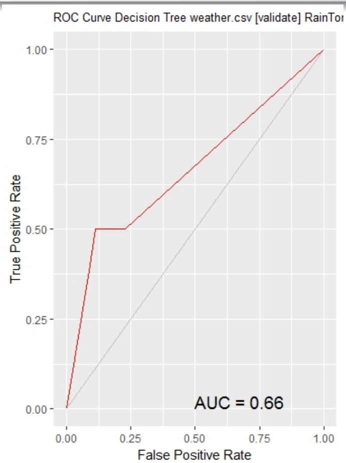
- **AUC=1** - a perfect classifier;
- **AUC<0.5** - a classifier with no predictive value;

		predicted	
		1	0
observed	1	**TP**	**FN**
	0	**FP**	**TN**

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

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

AUC



- **AUC≥0.9** - a perfect classifier;
- **0.9>AUC≥0.8** - a excellent/good classifier;
- **0.8>AUC≥0.7** - a acceptable/fair classifier;
- **0.7>AUC≥0.6** - a poor classifier;
- **0.6>AUC≥0.5** - no discrimination;

		predicted	
		1	0
observed	1	**TP**	**FN**
	0	**FP**	**TN**

$$FPR = \frac{FP}{TN + FP} \qquad TPR = \frac{TP}{TP + FN}$$

Thank you for your attention!!!