EVALUATING MODEL PERFORMANCE

Module 5

VISUALIZING PERFORMANCE TRADE-OFFS

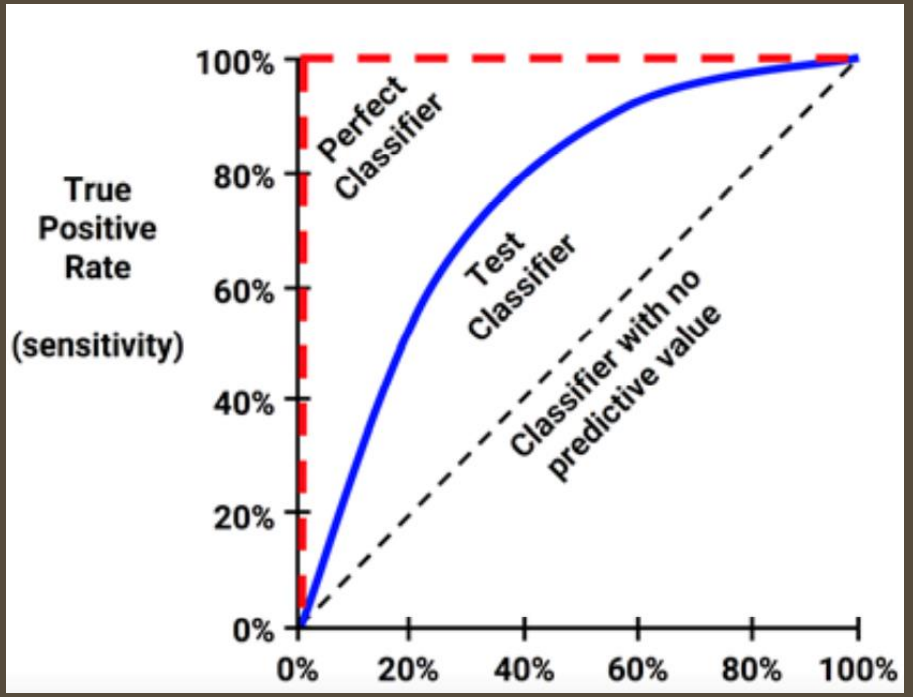
- Visualizations are helpful to understand the performance of machine learning algorithms in greater detail.
- It helps to depict how a learner performs across a wide range of conditions.

ROC curves

- The Receiver Operating Characteristic (ROC) curve is commonly used to examine the trade-off between the detection of true positives and false positives.

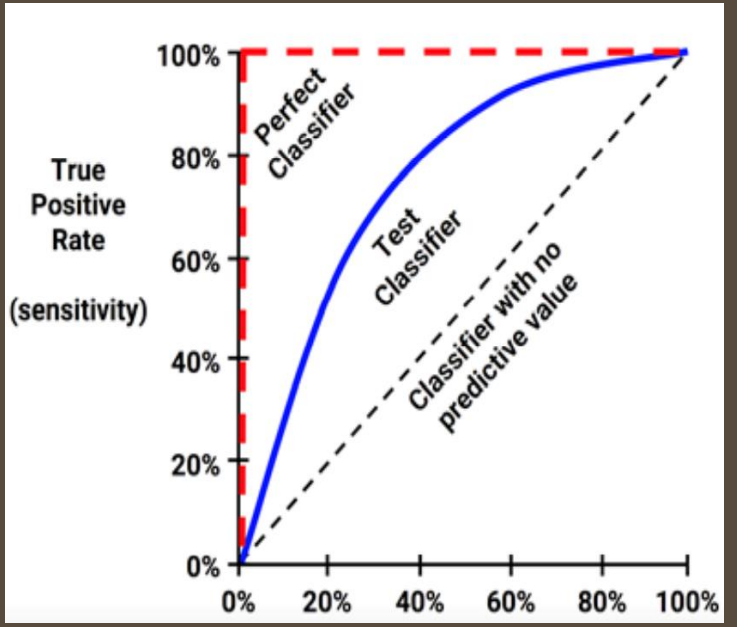
ROC curves

- The characteristics of a typical ROC diagram are depicted in the following plot.
- Curves are defined on a plot with the proportion of true positives on the vertical axis and the proportion of false positives on the horizontal axis.
  - These values are equivalent to sensitivity and (1 – specificity) respectively, and hence the diagram is also known as a sensitivity/specificity plot.



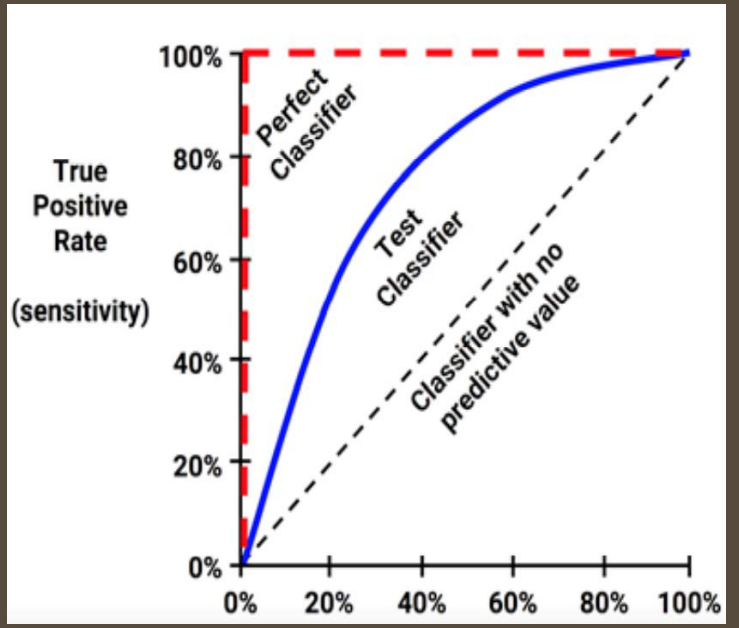
ROC curves

- The evaluation is done in the following manner:
  - The diagonal line from the bottom-left to the top-right corner of the diagram represents a **classifier with no predictive value**. This type of classifier detects true positives and false positives at exactly the same rate, implying that the classifier cannot discriminate between the two. This is the baseline by which other classifiers may be judged. **ROC curves falling close to this line indicate models that are not very useful.**
  - The **perfect classifier** has a curve that passes through the point at a 100 percent true positive rate and 0 percent false positive rate. It is able to correctly identify all of the positives and no incorrect classification on negative data.
  - Most real-world classifiers fall somewhere in the zone between perfect and useless.

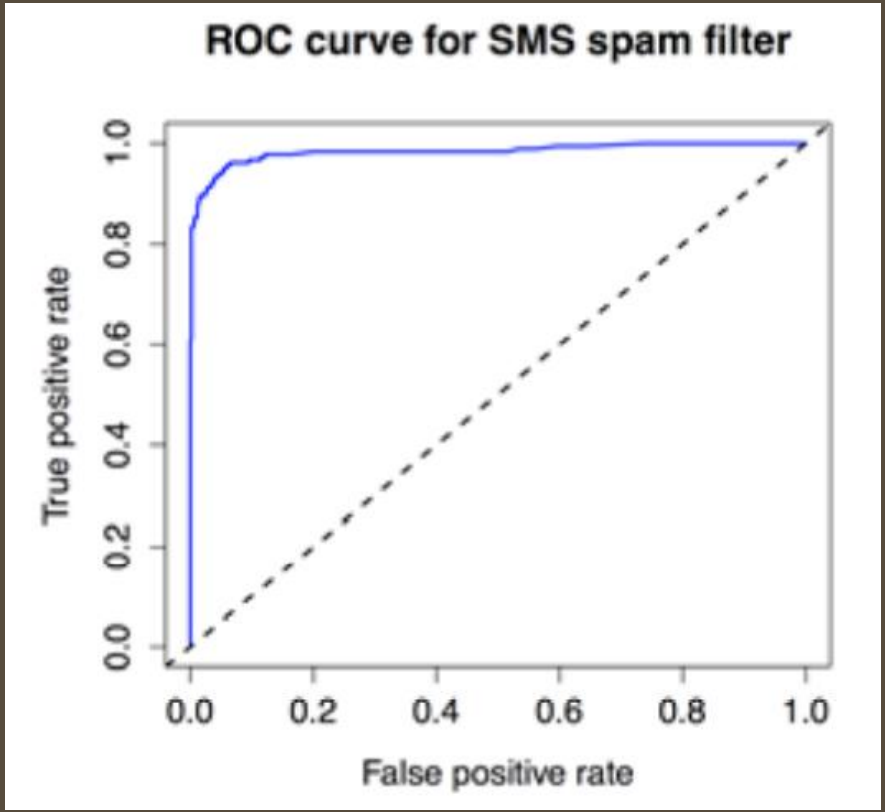


ROC curves

- The closer the curve is to the perfect classifier, the better it is at identifying positive values. This can be measured using a statistic known as the **area under the ROC curve** (abbreviated AUC).
- The AUC treats the ROC diagram as a two-dimensional square and measures the total area under the ROC curve. **AUC ranges from 0.5** (for a classifier with no predictive value) **to 1.0** (for a perfect classifier).
- A convention to interpret AUC scores uses a system similar to academic letter grades:
  - A: Outstanding = 0.9 to 1.0
  - B: Excellent/good = 0.8 to 0.9
  - C: Acceptable/fair = 0.7 to 0.8
  - D: Poor = 0.6 to 0.7
  - E: No discrimination = 0.5 to 0.6



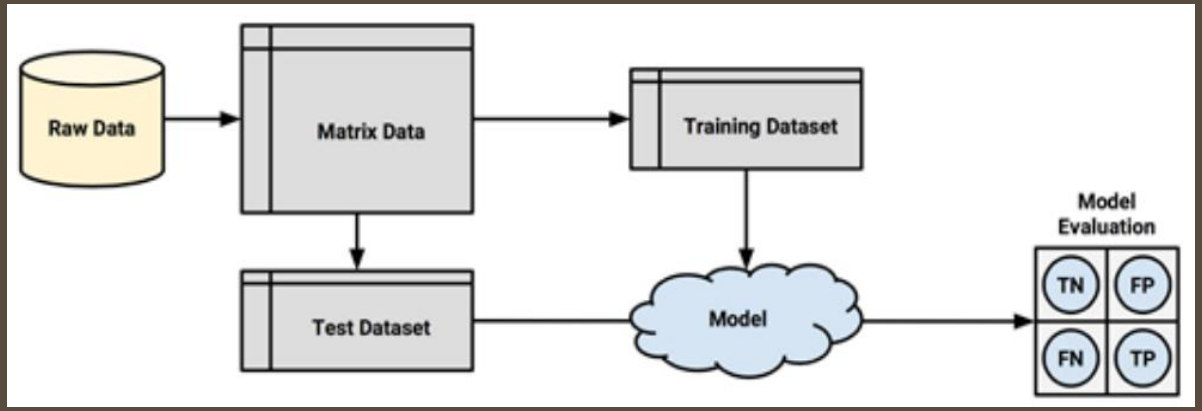
ROC curve evaluation- an Example



- we can see that this ROC curve appears to occupy the space at the top-left corner of the diagram, which suggests that it is closer to a perfect classifier than the dashed line representing a useless classifier.
- Here the AUC value is **0.984**, hence having outstanding performance.

Hold Out method

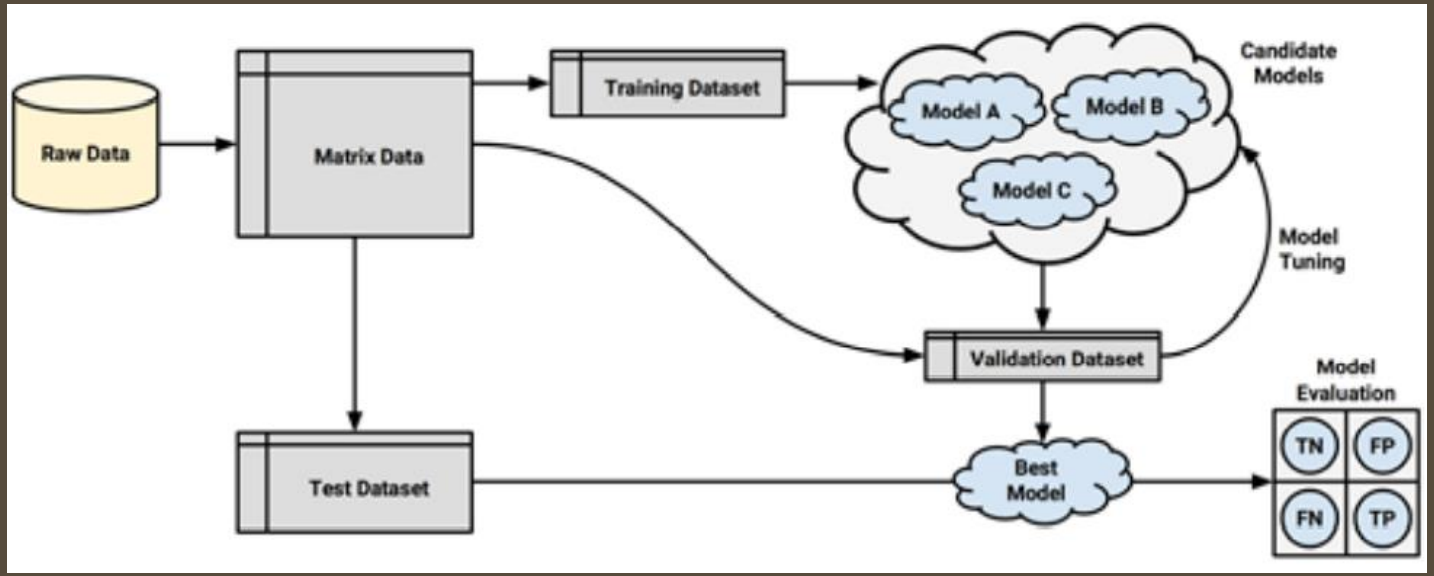
- The procedure of partitioning data into training and test datasets is known as the holdout method.
- The training dataset is used to generate the model, which is then applied to the test dataset to generate predictions for evaluation. Typically, about one-third of the data is held out for testing, and two-thirds is used for training. This proportion can vary depending on the amount of available data.



Hold Out method…

- We built several models on the training data, and selected the one with the highest accuracy on the test data.
- The test performance is a biased measure of the performance on unseen data.
- It is better to divide the original data into the training datasets, the test datasets, and a validation dataset.
- The validation dataset would be used for iterating and refining the model or models chosen, leaving the test dataset to be used only once as a final step to report an estimated error rate for future predictions.
- A typical split between training, test, and validation would be 50 percent, 25 percent, and 25 percent, respectively.

Hold Out method….



Hold Out method-Methods to generate splits

- A simple method to create holdout samples uses **random number generators** to assign records to partitions.
- Suppose we have a data frame with 1000 rows of data. We can divide it into three partitions as follows.
  - First, we create a vector of randomly ordered row IDs from 1 to 1000 using a random number generator function, which generates a specified number of random values between 0 and 1.
  - Next, we can use the resulting random IDs to divide the data frame into 500, 250, and 250 records comprising the training, validation, and test datasets.
  -

Hold Out method- Random Generation

- One problem with holdout sampling is that each partition may have a larger or smaller proportion of some classes.
- In certain cases, particularly those in which a class is a very small proportion of the dataset, this can lead a class to be omitted from the training dataset.
- This is a significant problem, because the model will not be able to learn this class.

Hold Out method- Stratified Random Sampling

- To avoid such kind of occurring, a technique called **stratified random sampling** can be used.
- Stratified random sampling guarantees that the random partitions have nearly the same proportion of each class as the full dataset, even when some classes are small.
- A technique called **repeated holdout** is sometimes used to solve the problems of randomly composed training datasets.
- The repeated holdout method is a special case of the holdout method that uses the average result from several random holdout samples to evaluate a model's performance. As multiple holdout samples are used, it is less likely that the model is trained or tested on non representative data.

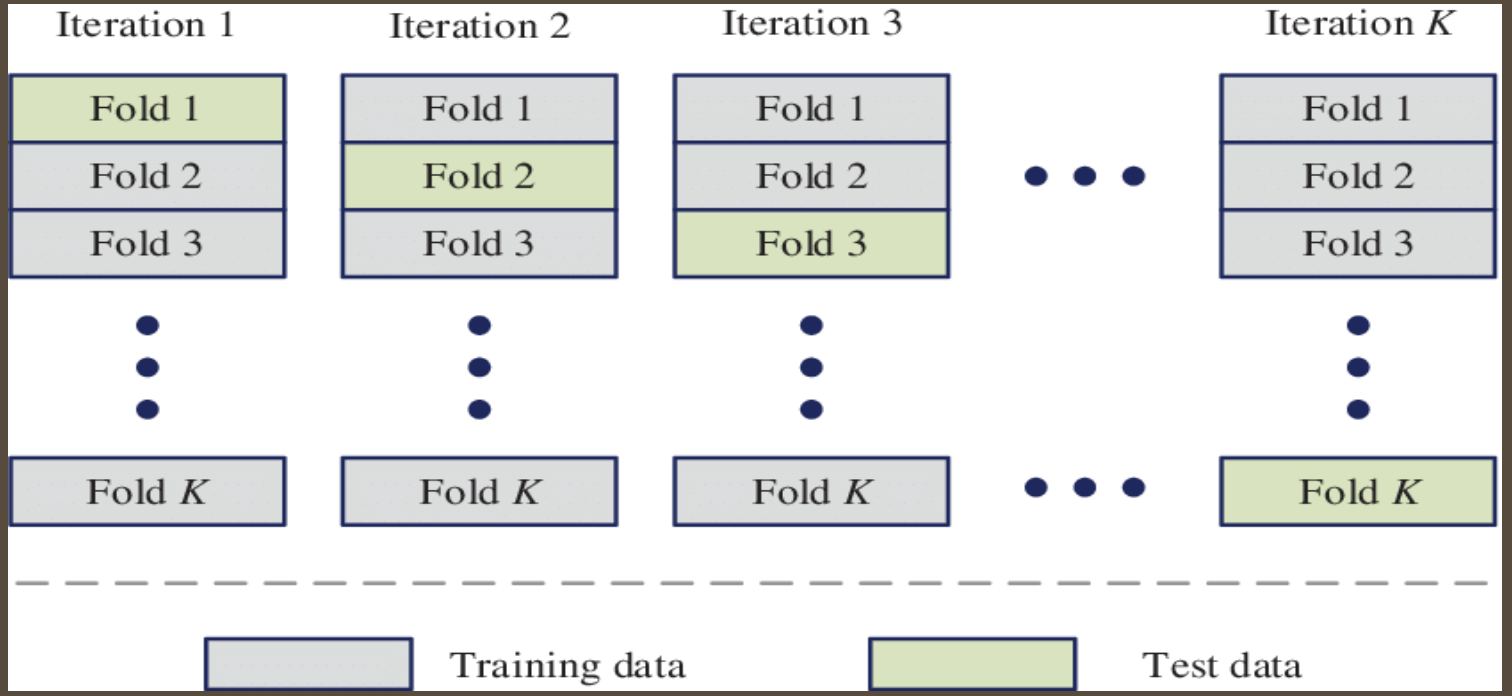
Cross-validation

- The repeated holdout is the basis of a technique known as **k-fold cross-validation** (or k-fold CV)
- It has become the industry standard for estimating model performance.
- In this method, rather than taking repeated random samples that could potentially use the same record more than once, k-fold CV randomly divides the data into k to completely separate random partitions called **folds**.

K-fold Cross Validation

- ‘k’ can be set to any number, the most common convention is to use 10-fold cross-validation (10-fold CV).
- For this, 10 folds are there. Each comprising 10 percent of the total data.
- For each of the 10 folds, a machine learning model is built on the remaining 90 percent of data. The fold's matching 10 percent sample is then used as test datafor model evaluation.
- After the process of training and evaluating the model has occurred for 10 times (with 10 different training/testing combinations), the average performance across all the folds is reported.

K-fold Cross Validation



K-fold Cross Validation- Example

- List of 10 folds
  - $ **Fold01**: int [1:100] 1 5 12 13 19 21 25 32 36 38 ...
  - $ **Fold02**: int [1:100] 16 49 78 81 84 93 105 108 128 134 ...
  - $ **Fold03**: int [1:100] 15 48 60 67 76 91 102 109 117 123 ...
  - $ **Fold04**: int [1:100] 24 28 59 64 75 85 95 97 99 104 ...
  - $ **Fold05**: int [1:100] 9 10 23 27 29 34 37 39 53 61 ...
  - $ **Fold06**: int [1:100] 4 8 41 55 58 103 118 121 144 146 ...
  - $ **Fold07**: int [1:100] 2 3 7 11 14 33 40 45 51 57 ...
  - $ **Fold08**: int [1:100] 17 30 35 52 70 107 113 129 133 137 ...
  - $ **Fold09**: int [1:100] 6 20 26 31 42 44 46 63 79 101 ...
  - **$ Fold10**: int [1:100] 18 22 43 50 68 77 80 88 106 111 ...

- Cross Validation Results
  - $ Fold01: 0.343
  - $ Fold02: 0.255
  - $ Fold03: 0.109
  - $ Fold04: 0.107
  - $ Fold05: 0.338
  - $ Fold06: 0.474
  - $ Fold07: 0.245
  - $ Fold08: 0.0365
  - $ Fold09: 0.425
  - $ Fold10: 0.505
  - mean(cv_results)→ 0.283796

Bootstrap sampling

- This refers to statistical methods of using random samples of data to estimate the properties of a larger set.
- When this principle is applied to machine learning model performance, it implies the creation of several randomly selected training and test datasets, which are then used to estimate performance statistics.
- The results from the various random datasets are then averaged to obtain a final estimate of future performance.

Bootstrap sampling

- How this procedure different from k-fold CV?
- The cross-validation divides the data into separate partitions in which each example can appear only once.
- The bootstrap allows examples to be selected multiple times through a process of sampling with replacement. This means that from the original dataset of n examples, the bootstrap procedure will create one or more new training datasets that will also contain n examples, some of which are repeated. The corresponding test datasets are then constructed from the set of examples that were not selected for the respective training datasets.

Bootstrap sampling

- One advantage of bootstrap over cross-validation is that it tends to work better with very small datasets.
- Using sampling with replacement as described previously, the probability that any given instance is included in the training dataset is 63.2 percent. Consequently, the probability of any instance being in the test dataset is 36.8 percent.
- A special case of bootstrapping known as the 0.632 bootstrap accounts for this by calculating the final performance measure as a function of performance on both the training data and the test data.
- The final error rate is then estimated as:

$$\text{error} = 0.632 \times \text{error}_{\text{test}} + 0.368 \times \text{error}_{\text{train}}$$

Bootstrap sampling