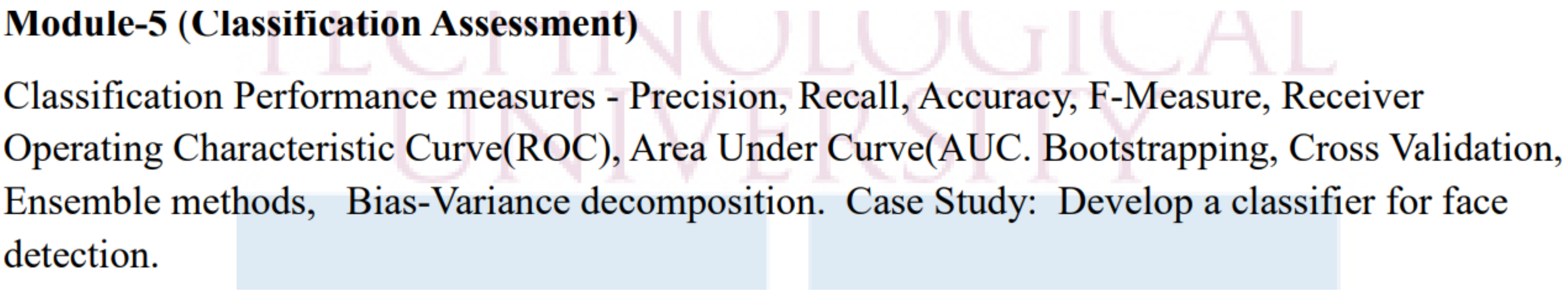
CST413 MACHINE LEARNING

Syllabus

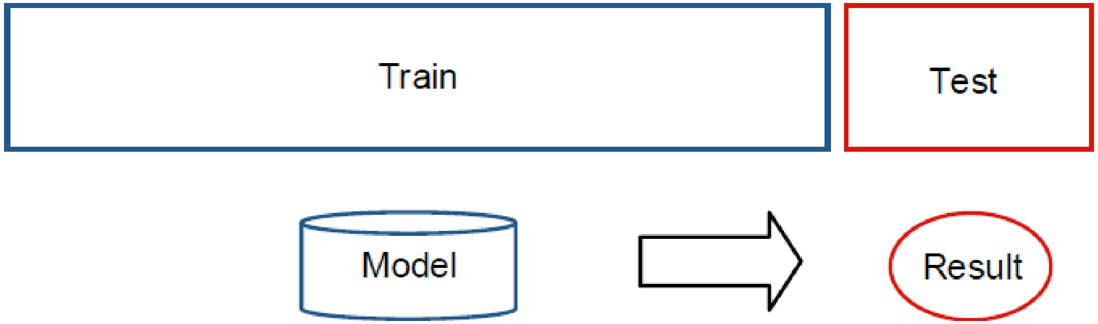


Module-5 (Classification Assessment)

Classification Performance measures - Precision, Recall, Accuracy, F-Measure, Receiver Operating Characteristic Curve(ROC), Area Under Curve(AUC. Bootstrapping, Cross Validation, Ensemble methods, Bias-Variance decomposition. Case Study: Develop a classifier for face detection.

Training and Test Set

- ❑ Datasets in machine learning are usually split into disjoint sets for **training** and **testing**
    - ❑ Training set is used to **fit and calibrate** the model parameters
    - ❑ Test set is used to measure the **predictive performance** on **unseen** data
- ❑ Each measures a different error, i. e. the **training and test error**
- ❑ Rule-of-thumb: 80% for training and 20% for testing (or 90% vs. 10%)

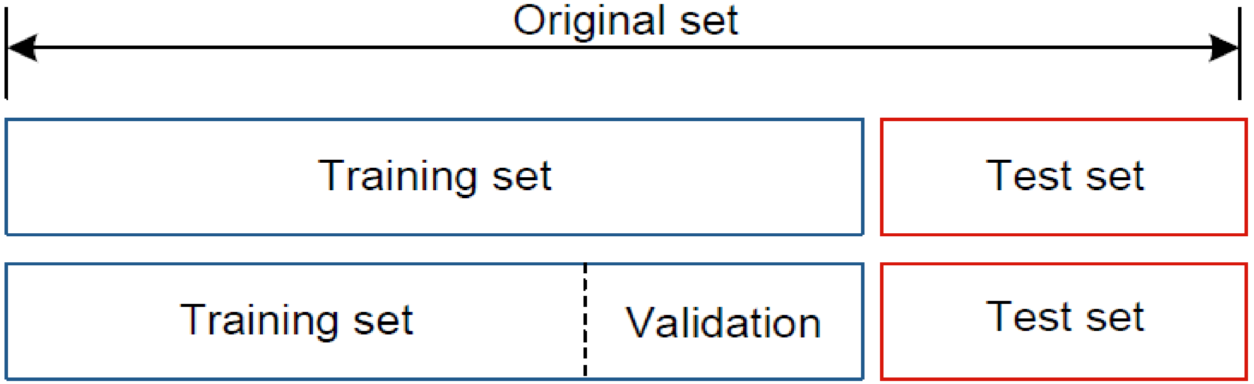


Training vs. Test Error

- ❑ **Training error** results from applying the model to the training data
- ❑ **Test error** is the average error when predicting on unseen observations
- ❑ Alternative terms refer to **in-sample** and **out-of-sample** performance

Validation

- **Validation** - The process of deciding whether the numerical results quantifying hypothesized relationships between variables, are acceptable as descriptions of the data, is known as validation
- Three-fold split into training, validation and test set
  - Fit model of different complexity to training data
  - Select model based on performance on unseen data from the validation set
  - Measure predictive performance based on the test set.



(80,20)

60,20,20

- Training dataset is used to train a few candidate models.(The actual dataset that we use to train the model)
- validation dataset is used to evaluate the candidate models.
  - *The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.*
  - *Make sure the models are not overfitting.*
- one of the candidates is chosen
  - the chosen model is trained with a new training dataset
  - the trained model is evaluated with the test dataset

Evaluation of classifiers(Model)

- Methods of evaluation
- Need for multiple validation set
  - When we apply a classification algorithm in a practical situation, we always do a validation test.
  - We keep a small sample of examples as validation set and the remaining set as the training set.
  - The classifier developed using the training set is applied to the examples in the validation set.
  - Based on the performance on the validation set, the accuracy of the classifier is assessed.
  - But, the performance measure obtained by a single validation set alone does not give a true picture of the performance of a classifier.
  - USE MULTIPLE VALIDATION SET

- Statistical distribution of errors.
  - We use a classification algorithm on a dataset and generate a classifier.
  - If we do the training once, we have one classifier and one validation error.
  - To average over randomness (in training data, initial weights, etc.),
  - we use the same algorithm and generate multiple classifiers.
  - We test these classifiers on multiple validation sets and record a sample of validation errors.
  - We base our evaluation of the classification algorithm on the statistical distribution of these validation errors.
  - We can use this distribution for assessing the expected error rate of the classification algorithm for that problem.

- No-free lunch theorem.
  - There is no such thing as the “best” learning algorithm. For any learning algorithm, there is a dataset where it is very accurate and another dataset where it is very poor. This is called the No Free Lunch Theorem.

Motivation for Resampling

- Sometimes performance is subject to the (random) split
    - If splitting is repeated randomly, there might be a **high variability** across the results.
    - Especially relevant for time-dependent or ordered data
    - Making splits **random** can be of importance here
- Model performance is often inferior the less data is used

Resampling

- Idea
- ❑ Repeatedly draw sub-samples from the given data set
- ❑ Then use these splits to fit and assess the model
- Common methods
- ❑ Cross-validation
  - ❑ Improved approach for estimating the test error
  - ❑ k-fold cross-validation
- ❑ Bootstrap
  - ❑ Quantify the uncertainty of an estimator or method
  - ❑ Returns standard errors or confidence intervals for a coefficient

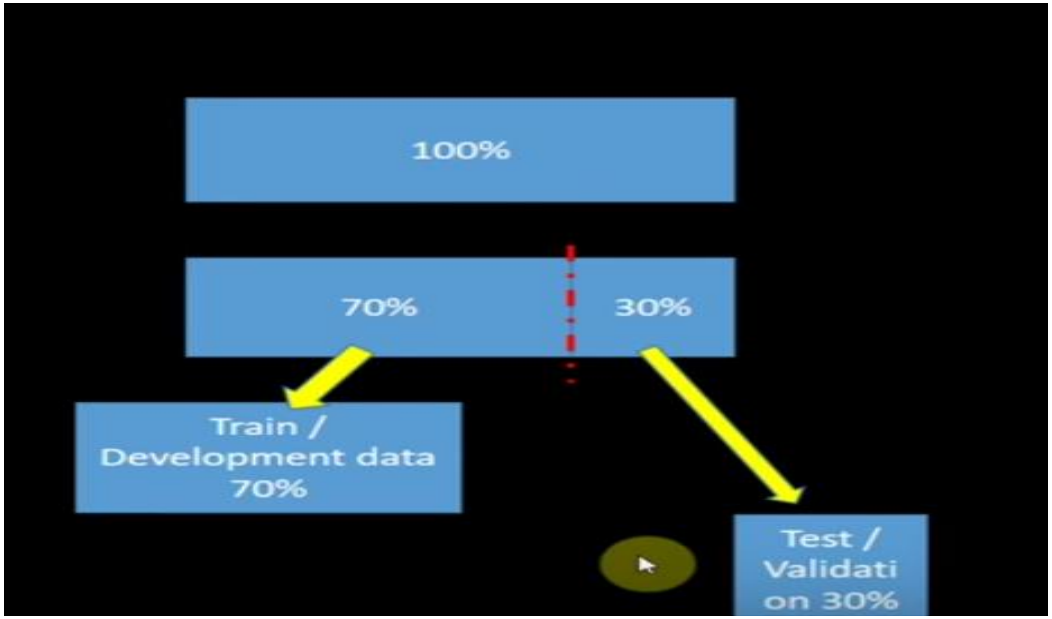
Cross-validation

- ❑ To test the performance of a classifier, we need to have a number of training/validation set pairs from a dataset X.
- ❑ To get them, if the sample X is large enough, we can randomly divide it then divide each part randomly into two and use one half for training and the other half for validation.
- ❑ Datasets are never large enough to do this. So, we use the same data split differently; this is called cross-validation.

- ❑ **Cross-validation** - technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.
  - ❑ Holdout Method
  - ❑ K-Fold Cross Validation

Holdout Method

- ❑ The simplest kind of cross validation.
- ❑ The data set is separated into two sets, One split is set aside or hold out for training the model. Another set is set aside or hold out for testing or evaluating the model.
- ❑ The algorithm fits a function using training set only.
- ❑ Then the function is used to predict the output values for the data in the testing set.
- ❑ The errors it makes are used to evaluate the model.
- ❑ In this method, we perform training on the 50- 70% of the given data-set and rest 50-30% is used for the testing purpose.
- ❑ The major drawback of this method is that we perform training on the 50% of the dataset, it may possible that the remaining 50% of the data contains some important information which we are leaving while training our model.



100%

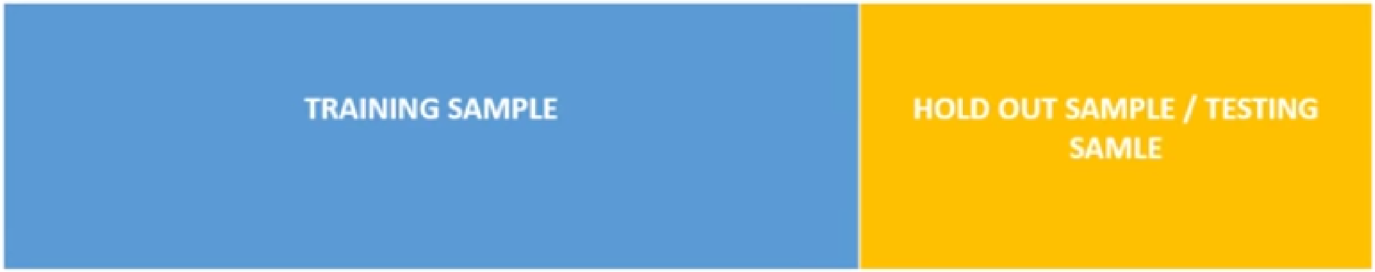
70%

30%

Train / Development data 70%

Test / Validati on 30%

Hold Out Method



TRAINING SAMPLE

HOLD OUT SAMPLE / TESTING SAMLE

50-70 %

50-30 %

- The following is the process of using hold-out method for model evaluation:
  - Split the dataset into two parts (preferably based on 70-30% split; However, the percentage split will vary)
  - Train the model on the training dataset; While training the model, some fixed set of hyper parameters is selected.
  - Test or evaluate the model on the held-out test dataset
  - Train the final model on the entire dataset to get a model which can generalize better on the unseen or future dataset.

- Hold-out method for Model Selection

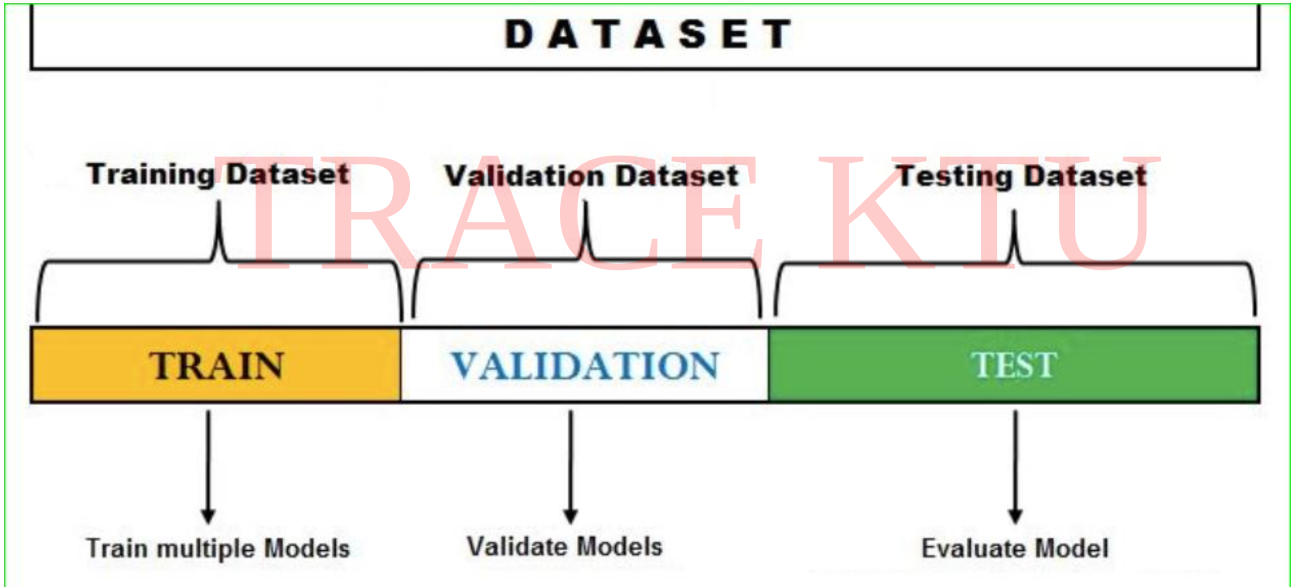


Fig 2. Hold out method – Training – Validation – Test Dataset

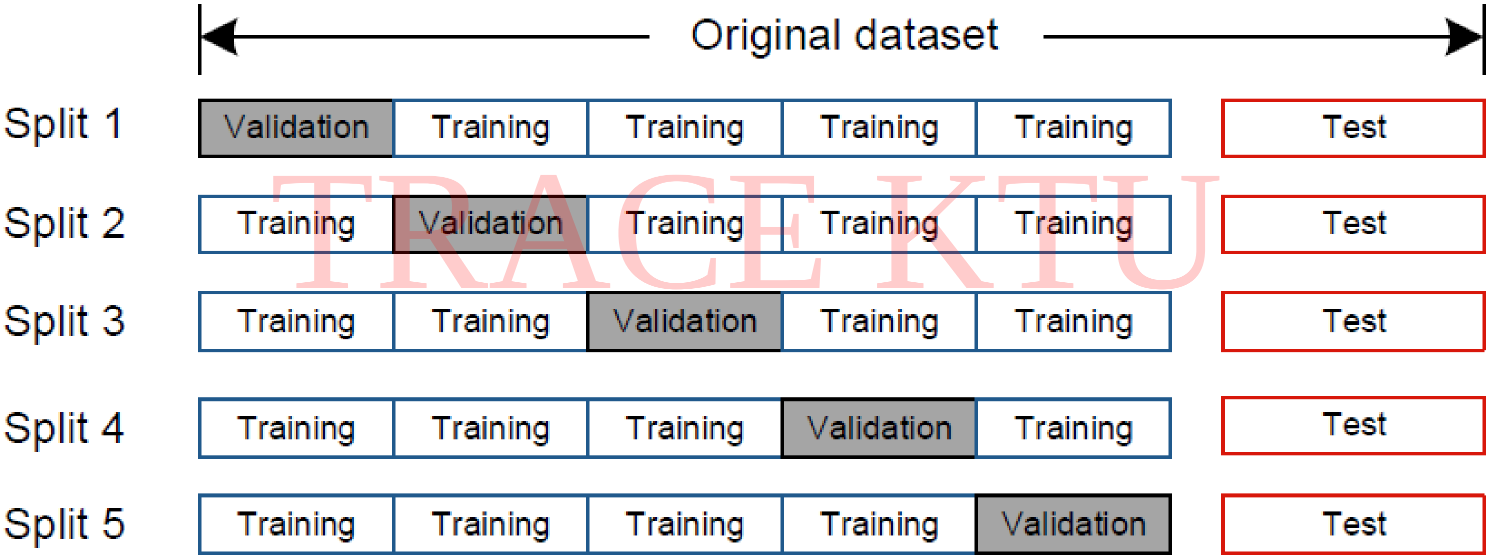
K-Fold Cross Validation

- The dataset X is divided into k equal sized parts(subsets).
- Xi, i= 1,…,K.
- To generate each pair, we keep one of the K parts out as the validation set Vi,and combine the remaining K−1 parts to form the training set Ti.
- Doing this K times, each time leaving out another one of the K parts out, we get K pairs(Vi,Ti).

$$
\begin{aligned}
V_1 &= X_1, \quad T_1 = X_2 \cup X_3 \cup \ldots \cup X_K \\
V_2 &= X_2, \quad T_2 = X_1 \cup X_3 \cup \ldots \cup X_K \\
&\cdots \\
V_K &= X_K, \quad T_K = X_1 \cup X_2 \cup \ldots \cup X_{K-1}
\end{aligned}
$$

- The error estimation is averaged over all k trials to get total effectiveness of our model.
- This significantly reduces bias as we are using most of the data for fitting, and also significantly reduces variance as most of the data is also being used in validation set.

5-Fold Cross Validation



Original dataset

Split 1	Validation	Training	Training	Training	Training	Test
Split 2	Training	Validation	Training	Training	Training	Test
Split 3	Training	Training	Validation	Training	Training	Test
Split 4	Training	Training	Training	Validation	Training	Test
Split 5	Training	Training	Training	Training	Validation	Test

1. Split the data into k equal subsets
2. Perform k rounds of learning; on each round
   - 1/k of the data is held out as a test(validation)set and
   - the remaining examples are used as training data.
3. Compute the average test/validation set score of the k rounds

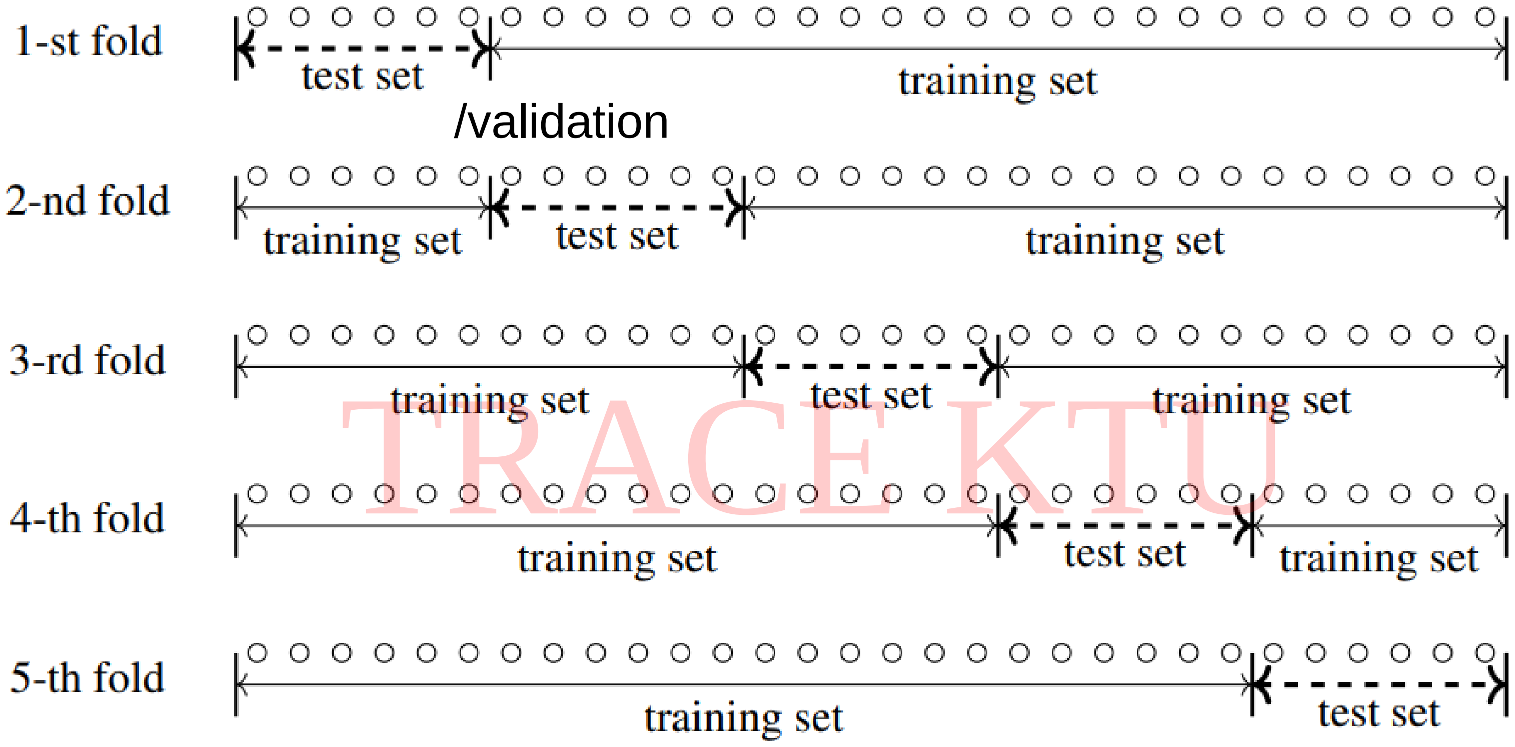


Figure 5.1: One iteration in a 5-fold cross-validation

Leave-one-out cross-validation

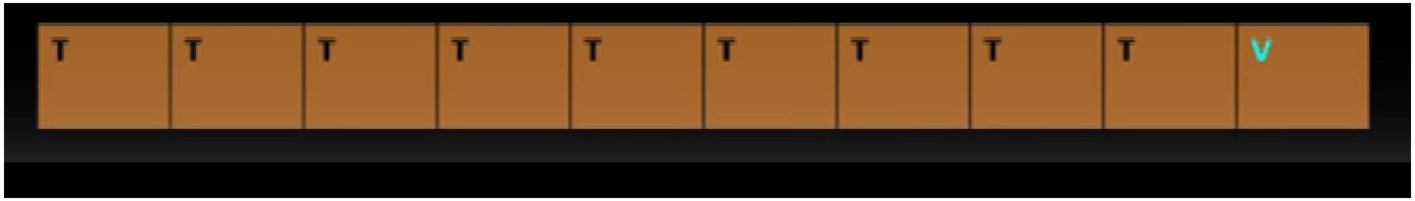
- This method is used when the data set is very small.
- It is a special case of k fold cross-validation.
- Each fold of the cross validation has only **a single test example** and all the rest of the data is used in training.
  - K-fold cross-validation is leave-one-out where given a dataset of N instances, only one instance is left out as the validation set and training uses the remaining N − 1 instances.
  - We then get N separate pairs by leaving out a different instance at each iteration.

1. Train the model on N-1 data points

2. Testing the model against that one data points which was left in the previous step

3.Calculate prediction error

4. Repeat above 3 steps until the model is not trained and tested on all data points



5×2 cross-validation

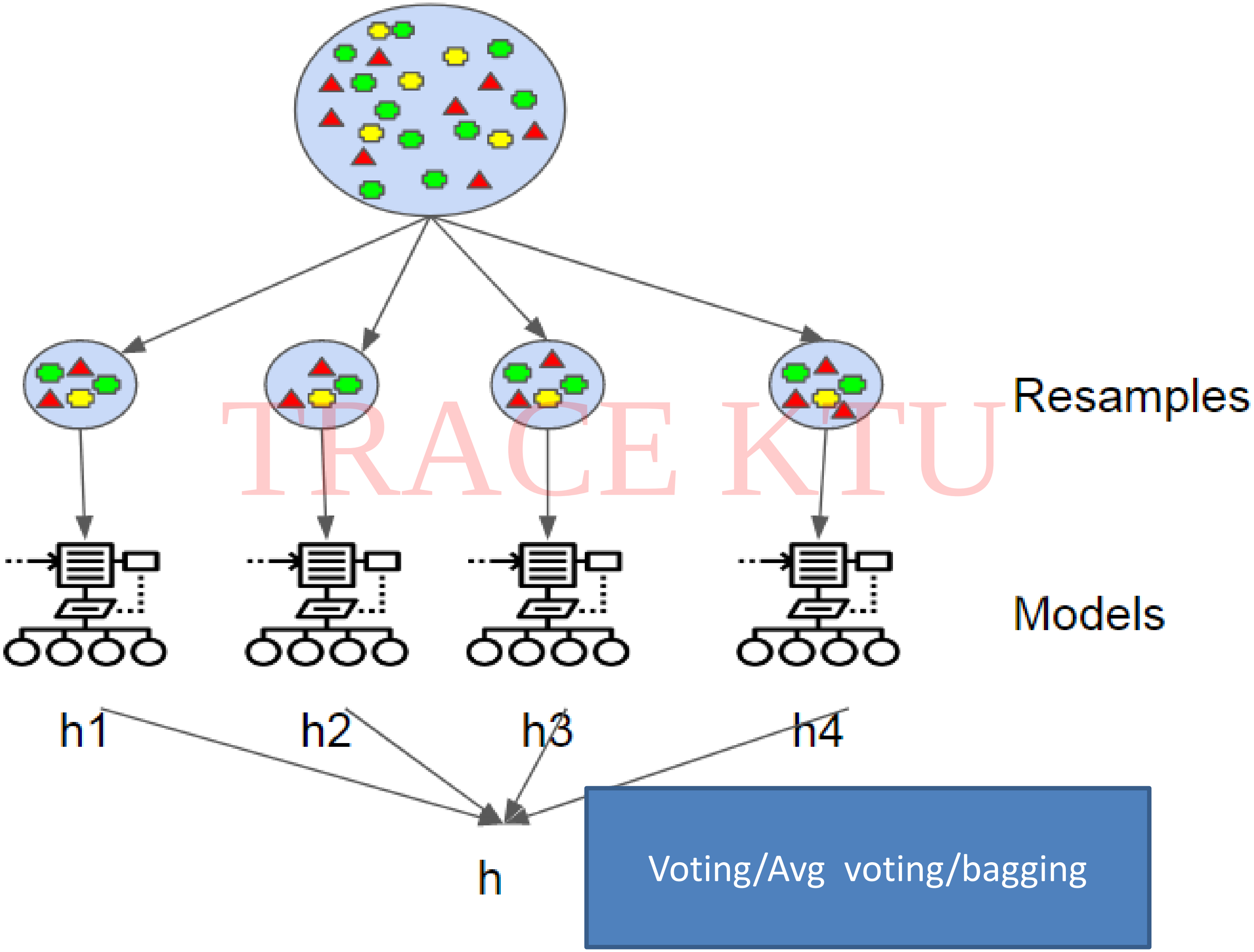
- In this method,the dataset X is divided into two equal parts
  - $X(1)^1$ and $X(1)^2$.
- We take as the training set $X(1)^1$ and $X(1)^2$ as the validation set.
- We then swap the two sets and
- take $X(1)^2$as the training set and $X(1)^1$ as the validation set. This is the first fold.
- The process repeated four more times to get ten pairs of training sets and validation set

$$\left.\begin{array}{ll} T_1 = X_1^{(1)}, & V_1 = X_1^{(2)} \\ T_2 = X_1^{(2)}, & V_2 = X_1^{(1)} \end{array}\right\} \text{One fold}$$

$$\begin{array}{ll} T_3 = X_2^{(1)}, & V_3 = X_2^{(2)} \\ T_4 = X_2^{(2)}, & V_4 = X_2^{(1)} \\ \vdots & \\ T_9 = X_5^{(1)}, & V_3 = X_5^{(2)} \\ T_{10} = X_5^{(2)}, & V_{10} = X_5^{(1)} \end{array}$$

Bootstrapping

- ❑ Process of computing performance measures using several randomly selected training and test datasets which are selected through a process of sampling with replacement, that is, through bootstrapping.
- ❑ Sample dataset are selected multiple times.
- ❑ The bootstrap procedure will create one or more new training datasets 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.

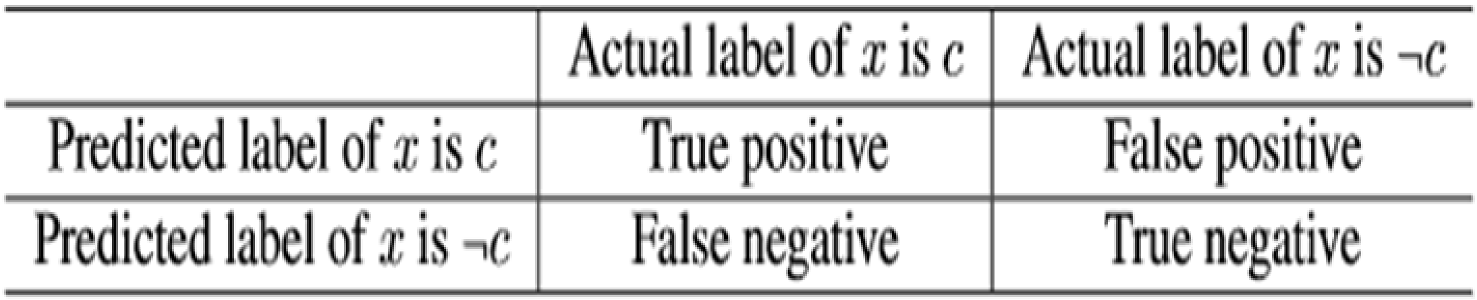


Voting/Avg voting/bagging

Measuring Error

Consider a Binary Classification Model derived from a two-class dataset.

Let x be a test instance



	Actual label of x is c	Actual label of x is $\neg c$
Predicted label of x is c	True positive	False positive
Predicted label of x is $\neg c$	False negative	True negative

Measuring Error

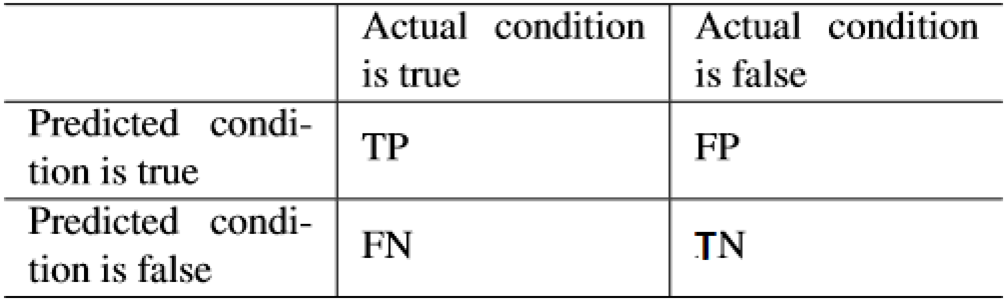
Test : Which objects are blue?

Objects: 🟦 , ⬛

Object	Test Result	Think	What is this?
🟦	Blue	Correctly identified	True Positive
⬛	Blue	Incorrectly identified	False Positive
🟦	Not Blue	Incorrectly rejected	False Negative
⬛	Not Blue	Correctly rejected	True Negative

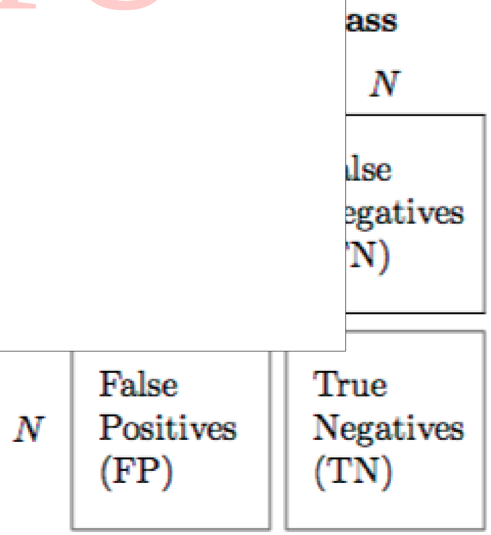
Confusion matrix

- ❑ A co e of a
  class ta for
  whic
- ❑ A co ns
  acco



	Actual condition is true	Actual condition is false
Predicted condition is true	TP	FP
Predicted condition is false	FN	TN

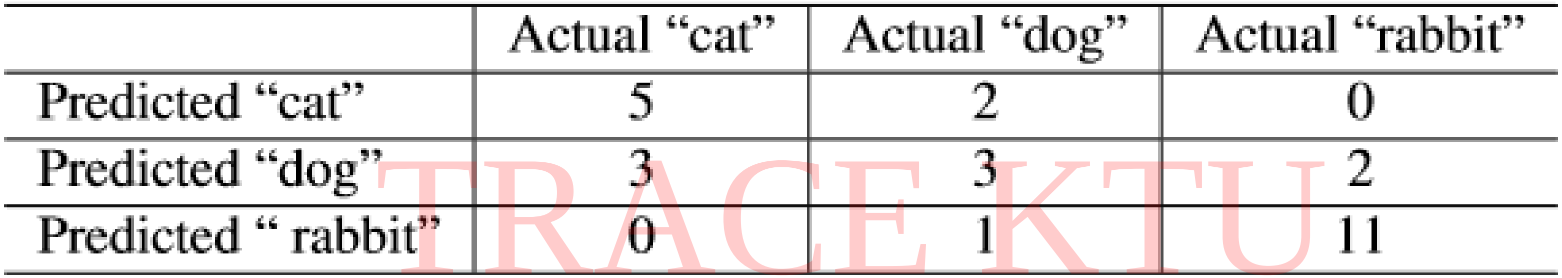
Table 5.1: Confusion matrix for two-class dataset



ass
N
lse
egatives
N)
N
False Positives (FP)
True Negatives (TN)

Confusion matrix

- If a classification system has been trained to distinguish between cats, dogs and rabbits, a confusion matrix will summarize the results of testing the algorithm for further inspection. Assuming a sample of 27animals-8 cats,6 dogs,and13 rabbits
- The resulting confusion matrix could look like the table below:



	Actual “cat”	Actual “dog”	Actual “rabbit”
Predicted “cat”	5	2	0
Predicted “dog”	3	3	2
Predicted “ rabbit”	0	1	11

Recall,Precision:

- ❑ In machine learning, precision and recall are two measures used to assess the quality of results produced by a binary classifier.

- ❑ **Recall:**the ratio of the total number of correctly classified positive examples divide to the total number of positive examples.
- ❑ High Recall indicates the class is correctly recognized

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

Precision:

- **Precision:**total number of correctly classified positive examples by the total number of predicted positive examples.

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

- **High recall, low precision: This** means that most of the positive examples are correctly recognized (low FN) but there are a lot of false positives.
- **Low recall, high precision: This** shows that we miss a lot of positive examples (high FN) but those we predict as positive are indeed positive (low FP)

- Classification Rate or Accuracy is given by the relation:

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

Problem 1

- Suppose a computer program for recognizing dogs in photographs identifies eight dogs in a picture containing 12 dogs and some cats. Of the eight dogs identified, five actually are dogs while the rest are cats. Compute the precision and recall of the computer program.
- TP =5
- FP = 3
- FN = 7
- The precision P is

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{5}{5+3} = \frac{5}{8}$$

- The recall R is

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \frac{5}{5+7} = \frac{5}{12}$$

Problem 2

- ❑ Let there be 10 balls (6 white and 4 red balls) in a box and let it be required to pick up the red balls from them. Suppose we pick up 7 balls as the red balls of which only 2 are actually red balls. What are the values of precision and recall in picking red ball?
- ❑ TP =2
- ❑ FP = 7-2 =5
- ❑ FN = 4-2=2

The *precision* P is

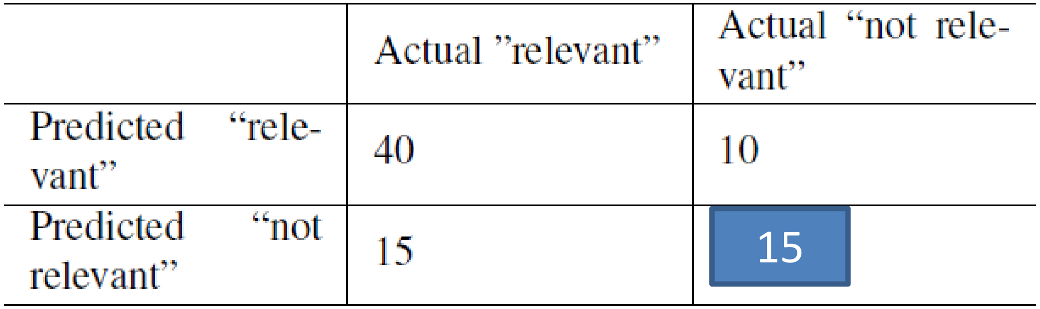
$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{2}{2+5} = \frac{2}{7}$$

The *recall* R is

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \frac{2}{2+2} = \frac{1}{2}$$

Problem 3

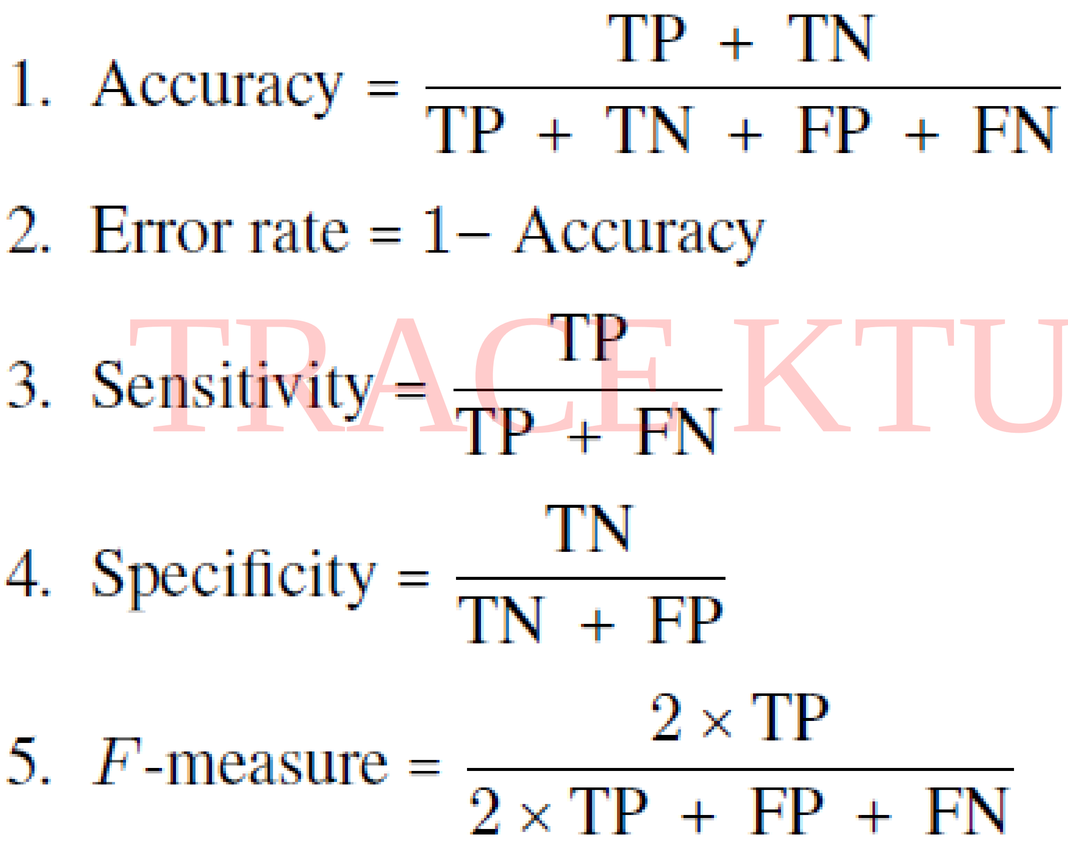
- ❑ Assume the following: A database contains 80 records on a particular topic of which 55 are relevant to a certain investigation. A search was conducted on that topic and 50 records were retrieved. Of the 50 records retrieved, 40 were relevant. Construct the confusion matrix for the search and calculate the precision and recall scores for the search.



	Actual ”relevant”	Actual “not relevant”
Predicted “relevant”	40	10
Predicted “not relevant”	15	15

55 (80-55=25) 80

Other measures of performance



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

2. $$\text{Error rate} = 1 - \text{Accuracy}$$

3. $$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

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

5. $$F\text{-measure} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}}$$

Receiver Operating Characteristic (ROC)

- ❑ **ROC** stands for Receiver Operating Characteristic
- ❑ The ROC curve was first developed by electrical engineers and radar engineers during World War II for detecting enemy objects in battlefields.
- ❑ They are now increasingly used in machine learning and data mining research.

TPR and FPR

Let a binary classifier classify a collection of test data.

Let,

TP=Number of true positives

TN=Number of true negatives

FP=Number of false positives

FN=Number of false negatives

$$\text{TPR} = \text{True Positive Rate} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

=Fraction of positive examples correctly classified

=Sensitivity(Recall)

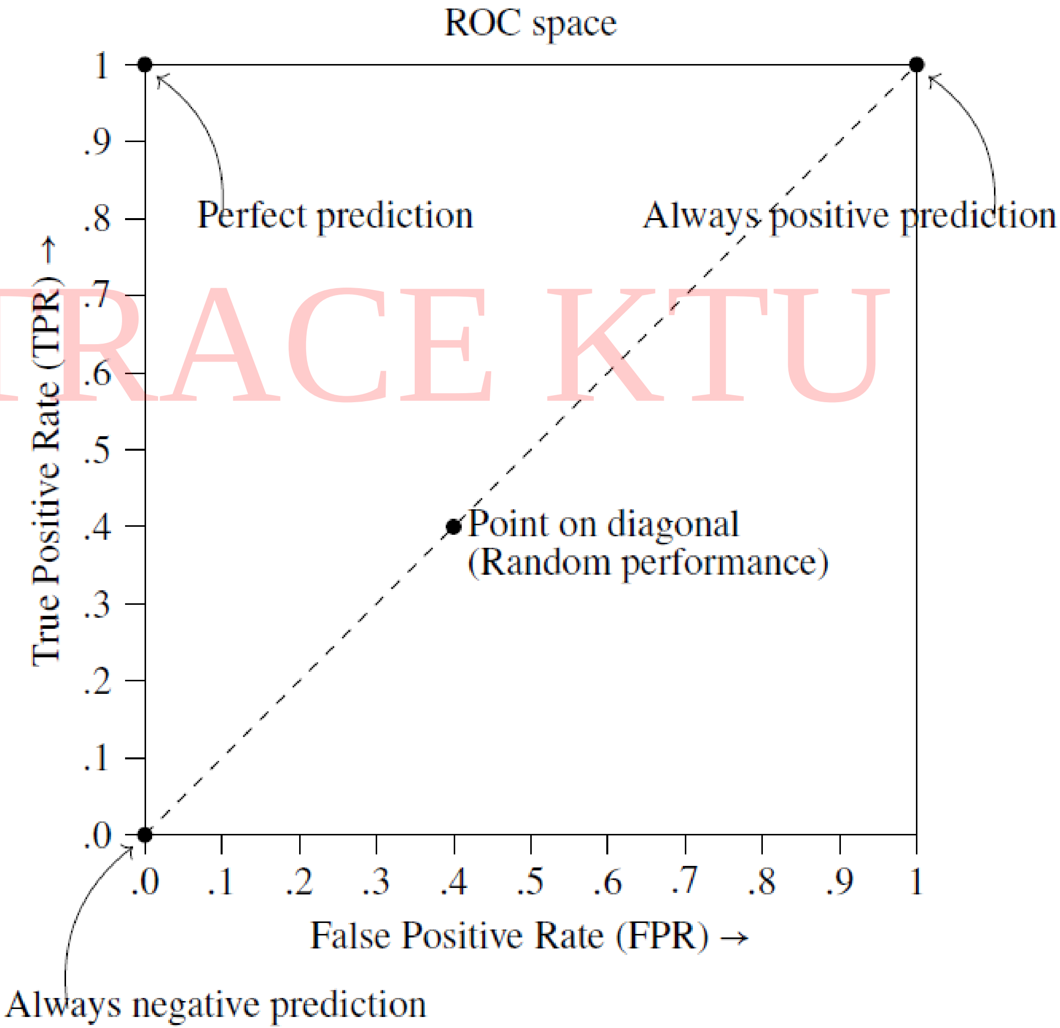
$$\text{FPR} = \text{False Positive Rate}$$

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

=Fraction of negative examples incorrectly classified
=1- Sensitivity

ROC space

visualize the performance of a binary classifier



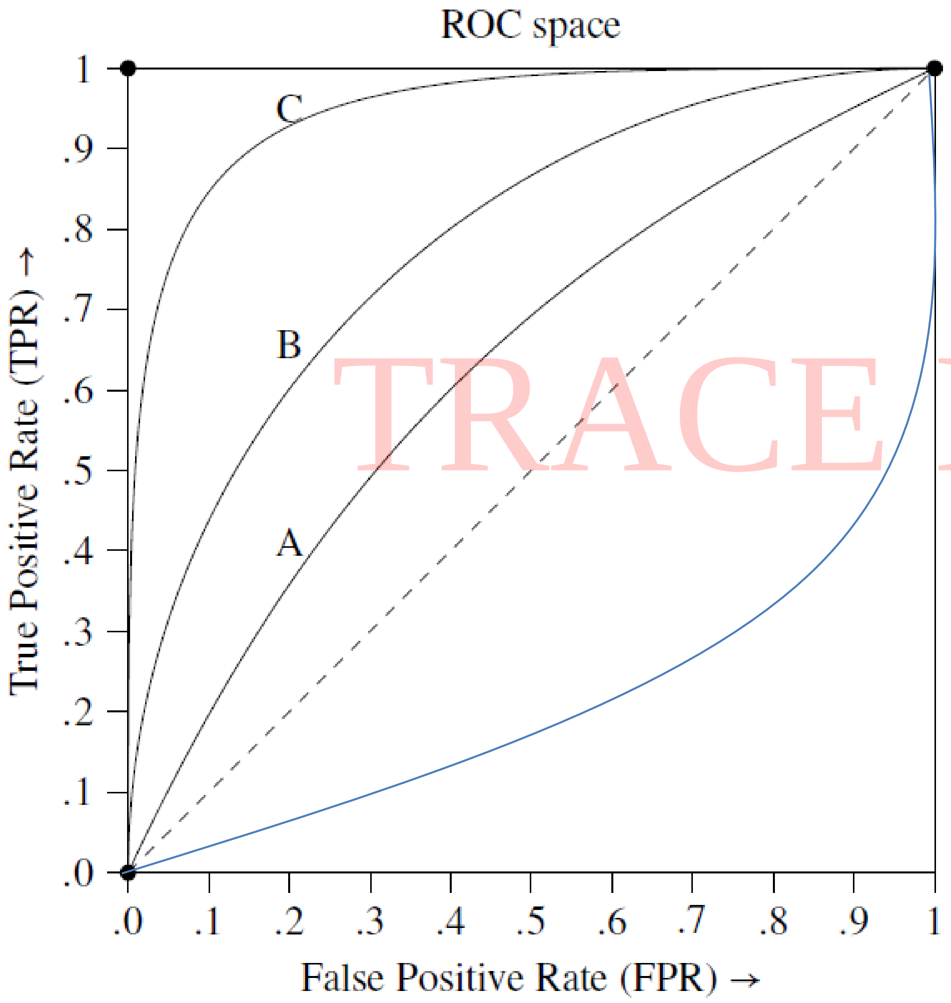
- We plot the values of FPR along the horizontal axis (that is , x-axis) and the values of TPR along the vertical axis (that is, y-axis) in a plane.
- For each classifier, there is a unique point in this plane with coordinates(FPR,TPR).
- The ROC space is the part of the plane whose points correspond to (FPR,TPR).
- The position of the point(FPR,TPR)in the ROC space gives an indication of the performance of the classifier.

ROC curve

- In the case of certain classification algorithms, the classifier may depend on a parameter.(threshold)
- Different values of the parameter will give different classifiers and these in turn give different values to TPR and FPR.

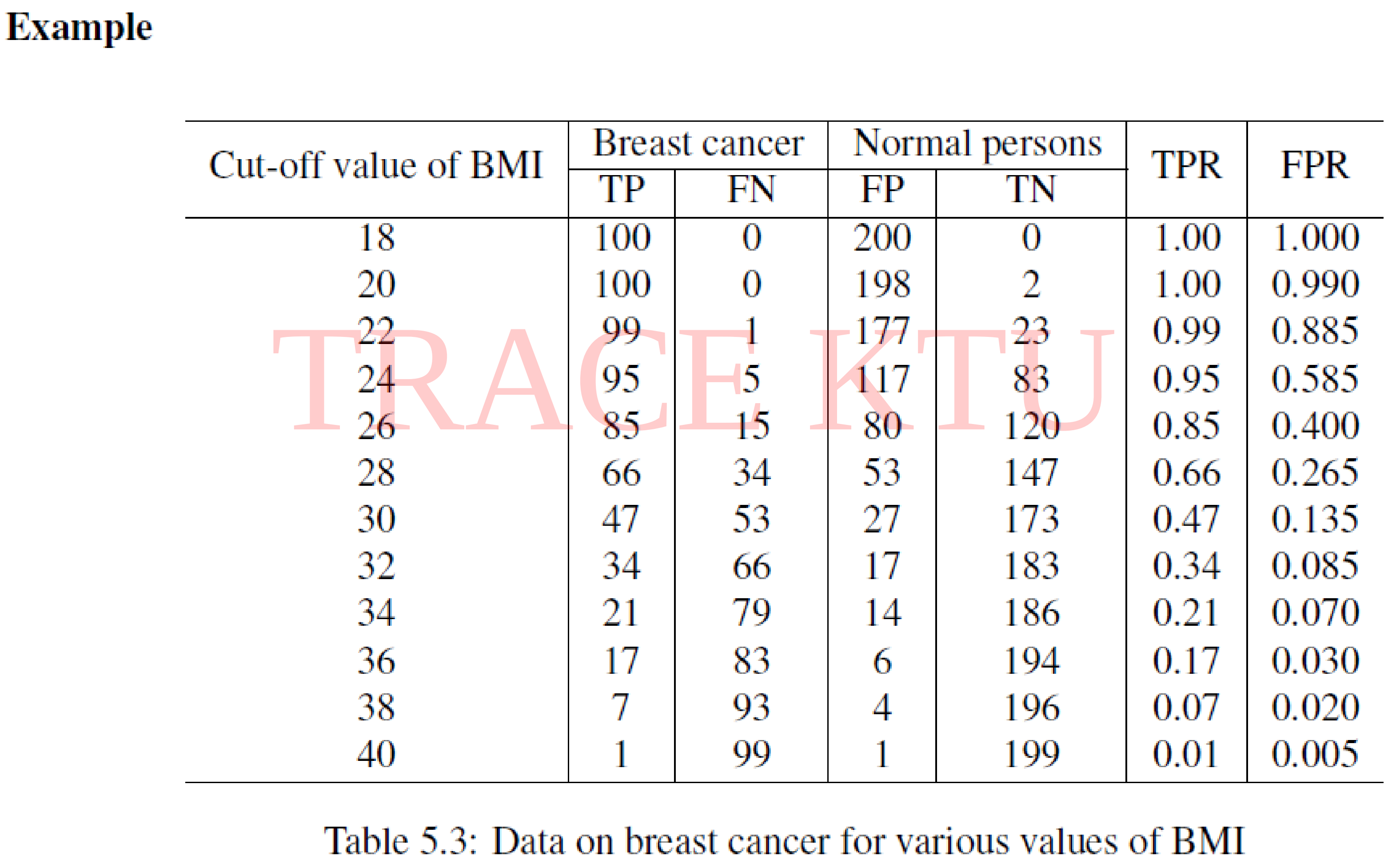
- The ROC curve is the curve obtained by plotting in the ROC space the points(TPR , FPR) obtained by assigning all possible values to the parameter in the classifier.
  - ie:is a graph showing the performance of a classifier at all classification thresholds

ROC space



The closer the ROC curve is to the top corner(0,1) of the ROC Space,
better the accuracy of the classifier

ROC space

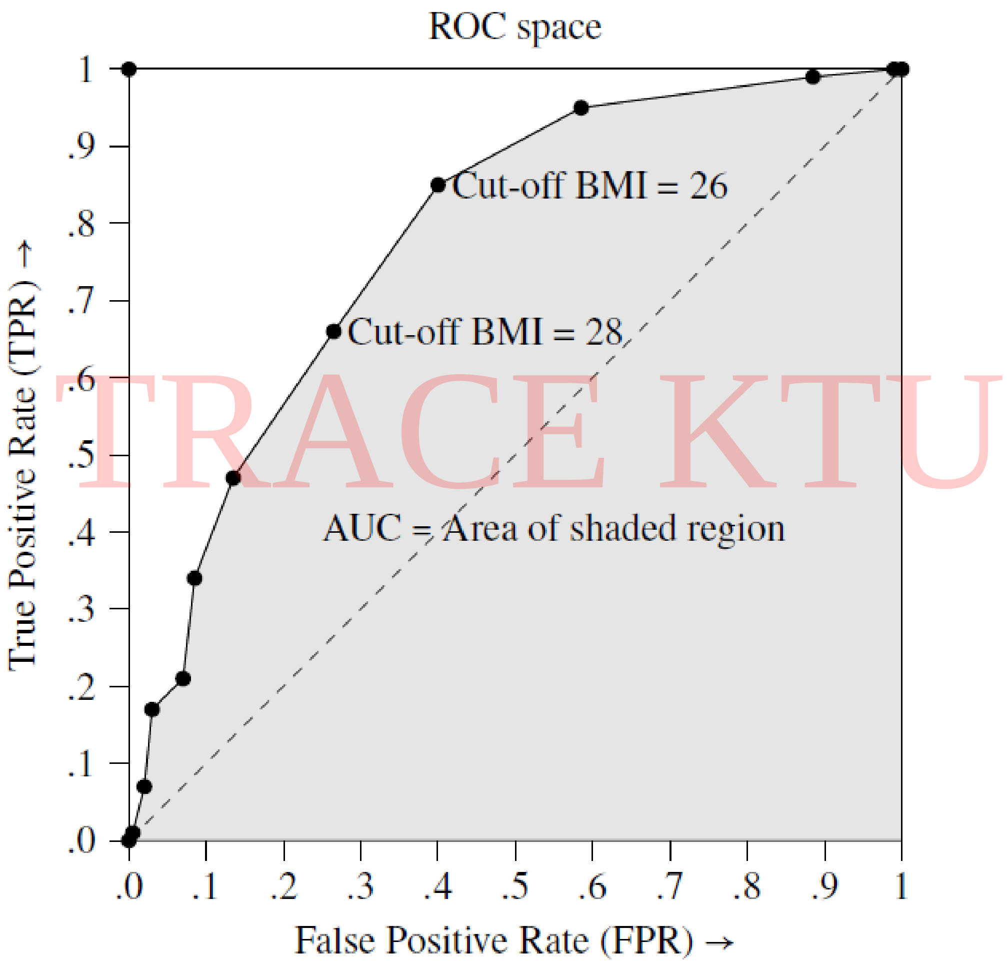


Example

Cut-off value of BMI	Breast cancer		Normal persons		TPR	FPR
	TP	FN	FP	TN		
18	100	0	200	0	1.00	1.000
20	100	0	198	2	1.00	0.990
22	99	1	177	23	0.99	0.885
24	95	5	117	83	0.95	0.585
26	85	15	80	120	0.85	0.400
28	66	34	53	147	0.66	0.265
30	47	53	27	173	0.47	0.135
32	34	66	17	183	0.34	0.085
34	21	79	14	186	0.21	0.070
36	17	83	6	194	0.17	0.030
38	7	93	4	196	0.07	0.020
40	1	99	1	199	0.01	0.005

Table 5.3: Data on breast cancer for various values of BMI

ROC space



ROC curve of data in Table 5.3 showing the points closest to the perfect prediction point

Area under the ROC curve :

- The measure of the area under the ROC curve is denoted by the acronym AUC.
- The value of AUC is a measure of the performance of a classifier.
- For the perfect classifier, AUC=1.0.

Different Classifiers

- Performance
  - None of the classifiers is perfect
  - Complementary
    - Examples which are not correctly classified by one classifier may be correctly classified by the other classifiers

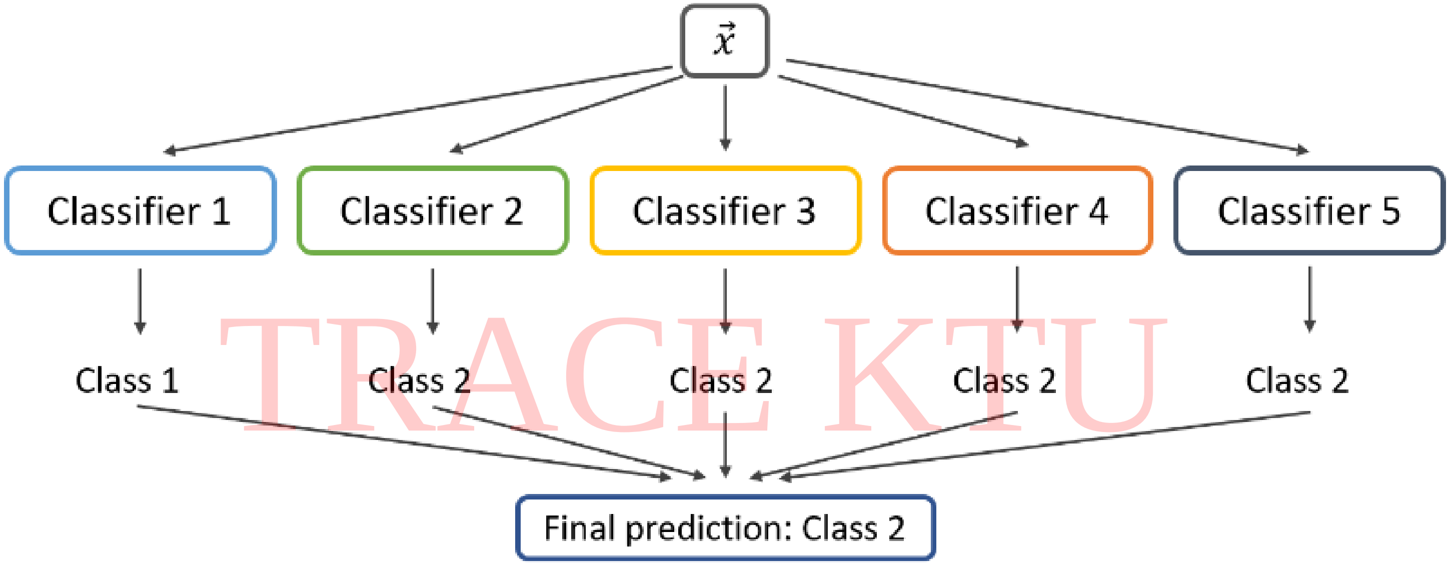
Ensembles of Classifiers

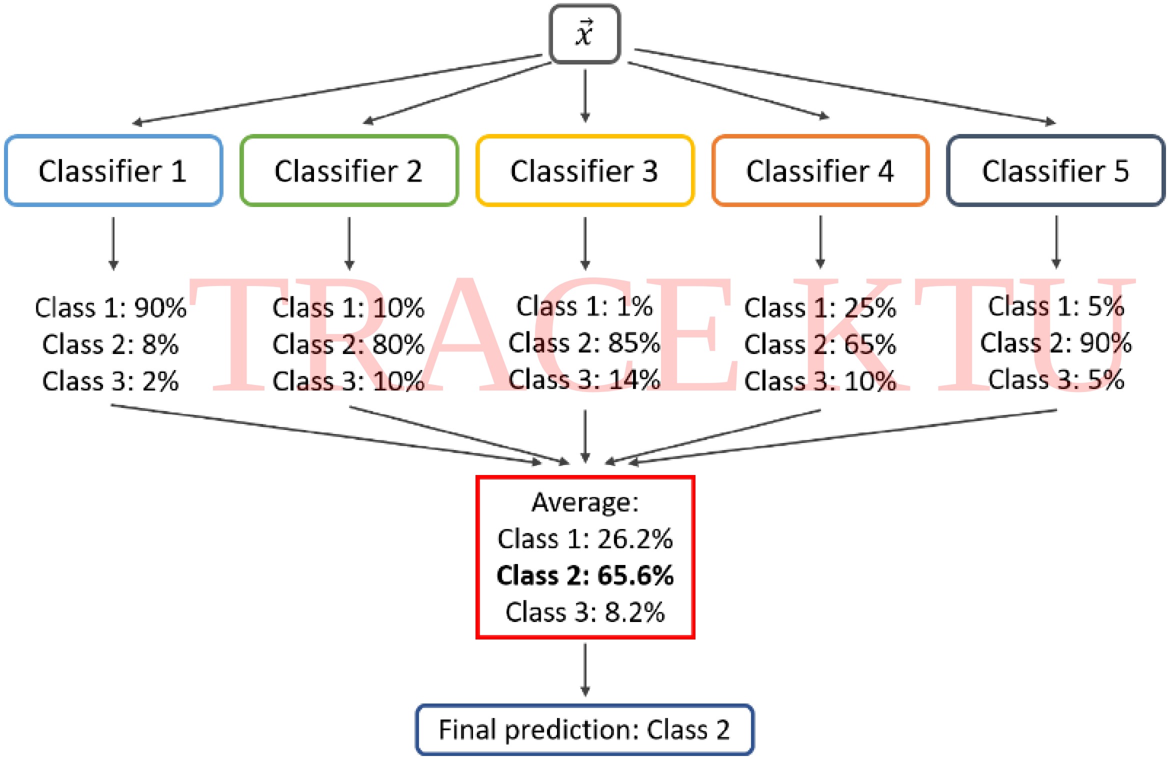
- Idea
  - Combine the classifiers to improve the performance
- Ensembles of Classifiers
  - Combine the classification results from different classifiers to produce the final output
    - Unweighted voting
    - Weighted voting

The different models may be chosen in many different ways:

- The models may be created using appropriate different algorithms like k-NN algorithm, NaiveBayes algorithm, decision tree algorithm, etc.
- The models may be created by using the same algorithm but using different splits of the same dataset into training data and test data.
- The models may be created by assigning different initial values to the parameters in the algorithm as in ANN algorithms.

- The models created in the ensemble learning methods are combined in several ways.
- Simple majority voting in classification problems: Every model makes a prediction (votes) for each test instance and the final output prediction is the one that receives more than half of the votes
- Weighted majority voting in classification problem: In weighted voting we count the prediction of the better models multiple times. Finding a reasonable set of weights is up to us.
- Simple averaging in prediction problems: In simple averaging method, for every instance of test dataset, the average predictions are calculated

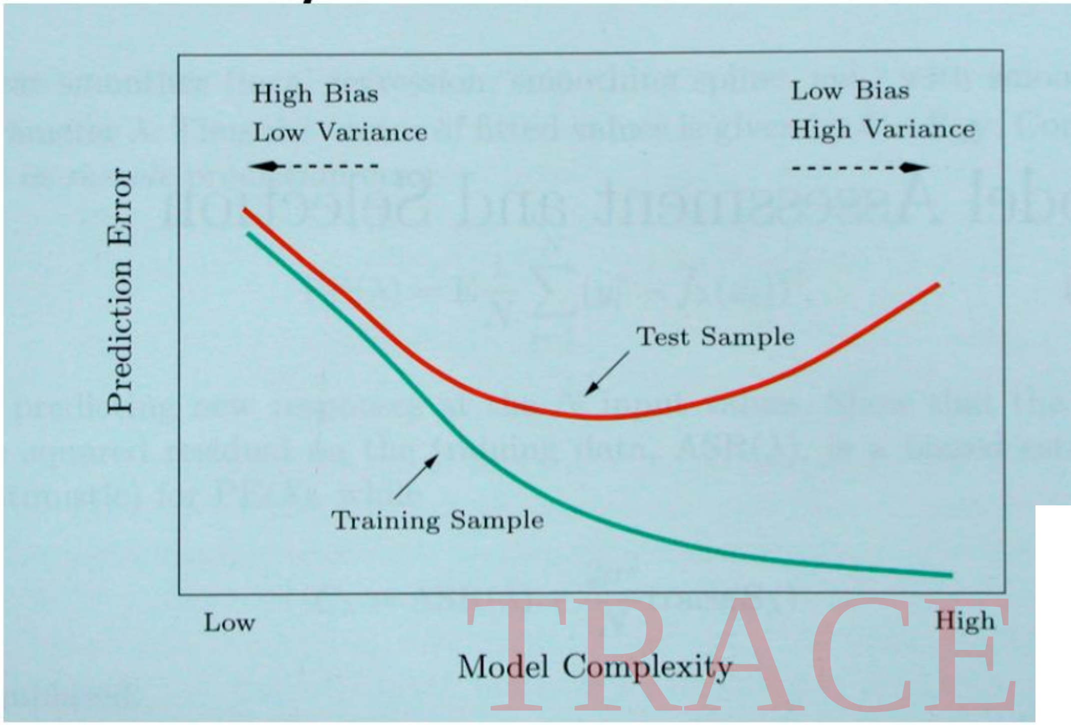

$\vec{x}$
Classifier 1
Classifier 2
Classifier 3
Classifier 4
Classifier 5
Class 1
Class 2
Class 2
Class 2
Class 2
Final prediction: Class 2

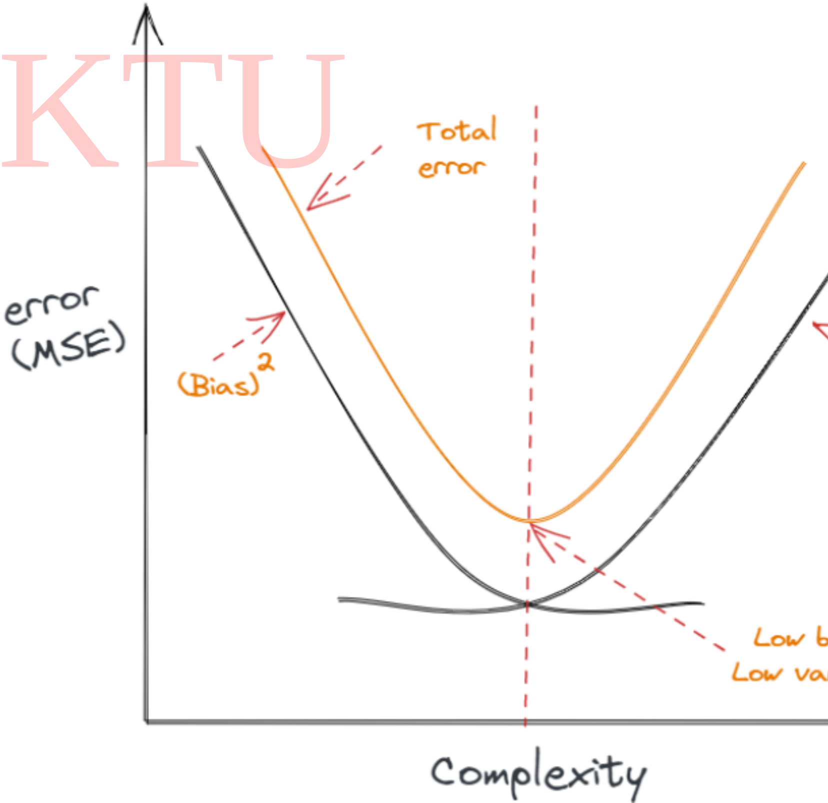

$\vec{x}$
Classifier 1
Classifier 2
Classifier 3
Classifier 4
Classifier 5
Class 1: 90%
Class 2: 8%
Class 3: 2%
Class 1: 10%
Class 2: 80%
Class 3: 10%
Class 1: 1%
Class 2: 85%
Class 3: 14%
Class 1: 25%
Class 2: 65%
Class 3: 10%
Class 1: 5%
Class 2: 90%
Class 3: 5%
Average:
Class 1: 26.2%
Class 2: 65.6%
Class 3: 8.2%
Final prediction: Class 2

Example: Weather Forecast

Reality	⛈	☀	☀	⛈	⛈	☀	☀
1	⛈	⛈ X	☀	☀ X	⛈	☀	⛈ X
2	☀ X	☀	☀	☀ X	⛈	☀	⛈ X
3	⛈	☀	⛈ X	⛈	☀ X	⛈ X	☀
4	⛈	☀	⛈ X	⛈	☀ X	☀	☀
5	⛈	⛈ X	☀	⛈	⛈	⛈ X	☀
Combine	⛈	☀	☀	⛈	⛈	☀	☀

Bias/Variance Tradeoff





it is required to make a balance between bias and variance errors, and this balance between the bias error and variance error is known as **the Bias-Variance trade-off.**

When to use bias-variance decomposition

- Since bias and variance are connected to underfitting and overfitting, decomposing the loss into bias and variance helps us understand learning algorithms.
  - **Low Bias:** Tends to suggest fewer implications about the target function's shape.
  - **High-Bias:** Suggests additional assumptions about the target function's shape.
  - **Low Variance:** Suggests minor changes to the target function estimate when the training dataset changes.
  - **High Variance:** Suggests that changes to the training dataset cause considerable variations in the target function estimate.
  - Theoretically, a model should have low bias and low variance but this is impossible to achieve.

- So, an optimal bias and variance are acceptable. Linear models have low variance but high bias and non-linear models have low bias but high variance.
- Linear models
  - (**Linear Regression, Logistic Regression, and Linear discriminant analysis**.
- Non Linear:
  - algorithms with high variance are **decision tree, Support Vector Machine, and K-nearest neighbours.**

- The total error of a machine learning algorithm has three components: bias, variance and noise.
- So decomposition is the process of derivation of total error
  - taking Mean Squared Error (MSE).
- Total error = Bias2 + Variance + Noise

Outline

- **Bias/Variance Tradeoff**

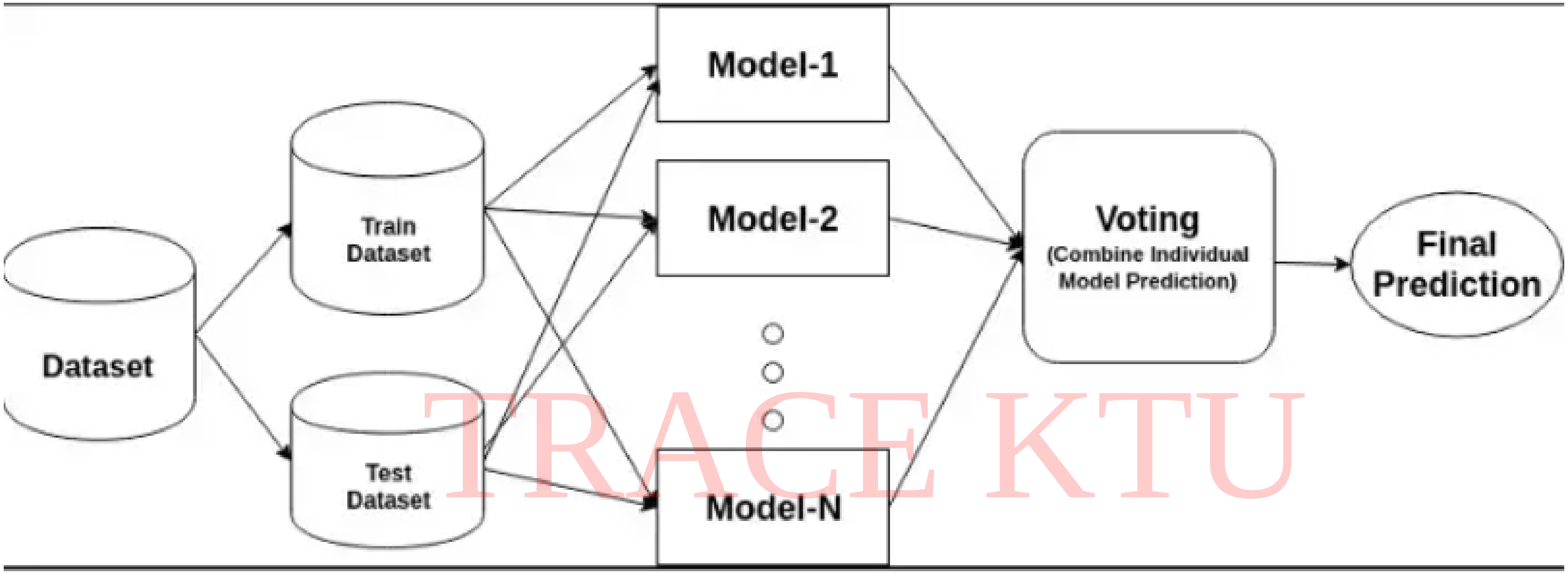
- Ensemble methods that minimize variance
  - Bagging
    - Random Forest

- Ensemble methods that minimize bias
  - Functional Gradient Descent
  - Boosting
  - Ensemble Selection

Ensemble learning

- *Ensemble learning is a way of generating various base classifiers from which a new classifier is derived which performs better than any constituent classifier*.
- These base classifiers may differ in the algorithm used, hyperparameters, representation or the training set.
- The key objective of the ensemble methods is to reduce **bias** and **variance**.

voting



Some of the advanced ensemble classifiers are:
Stacking
Blending
Bagging
Boosting

Bagging: Bootstrap Aggregation

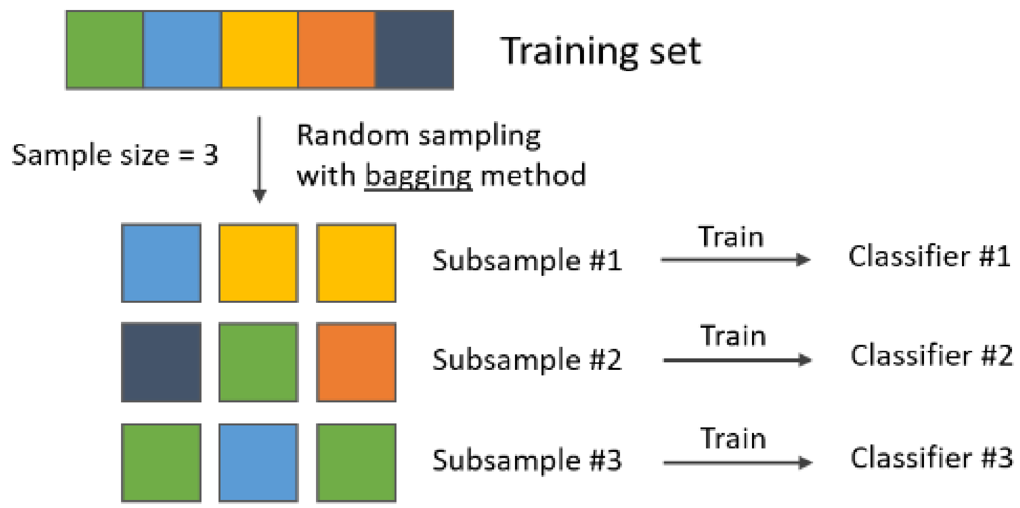
- Leo Breiman (1994)
- Take repeated bootstrap samples from training set *D*.
- *Bootstrap sampling*: Given set *D* containing *N* training examples, create *D'* by drawing *N* examples at random with replacement from *D*.

- Bagging:
  - Create *k* bootstrap samples D_1 … D_k.
  - Train distinct classifier on each D_i.
  - Classify new instance by majority vote / average.

Bagging (Bootstrap aggregation):

- In this method, *n samplings of training data are generated by picking various data items from the training data with replacement.*
  - **"with replacement" means that after one instance is taken randomly from the training set, a replacement of this instance is put into the training set. When the next instance is selected, there is a chance that this next instance selected is the same as the previous instance selected**.
- In bagging, the items in sampling are chosen randomly as the data is unweighted.
- For every iteration,
  - A base model is created on each of these samplings.
  - The models run in parallel and are independent of each other.
  - The final predictions are determined by combining the predictions from all the models.
  - These models collectively form a higher graded model to produce more accuracy.

- The final model is averaged by:
- $e = (\Sigma e_i)/n$
- where $e_1, e_2 \ldots\ldots e_n$ = base classifier
- e = final classifier
- Bagging algorithms:
  - Bagging meta-estimator
  - Random forest



Random forest

A random forest is an ensemble learning method where multiple decision trees are constructed and then they are merged to get a more accurate prediction

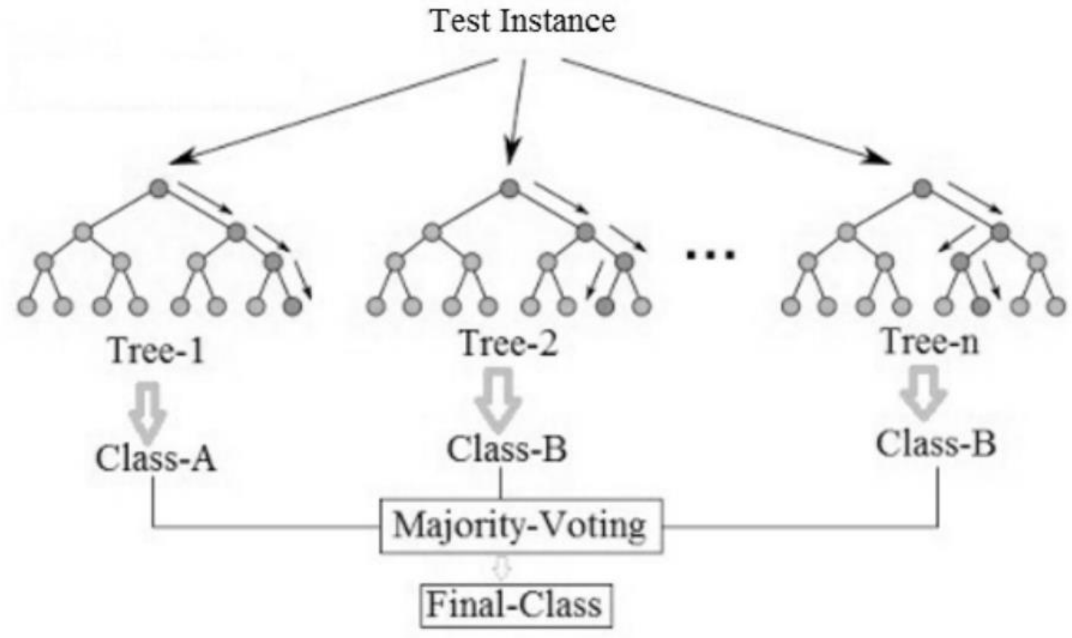


Figure 12.1: Example of random forest with majority voting

- Algorithm Here is an outline of the random forest algorithm.
- 1. The random forests algorithm generates many classification trees.
- Each tree is generated as follows:
  - (a) If the number of examples in the training set is N, take a sample of N examples at random - but with replacement, from the original data.

    This sample will be the training set for generating the tree.
  - (b) If there are M input variables, a number m is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node.
  - The value of m is held constant during the generation of the various trees in the forest.
  - (c) Each tree is grown to the largest extent possible.

2. To classify a new object from an input vector, put the input vector down each of the trees in the forest.

Each tree gives a classification, and we say the tree “votes” for that class. The forest chooses the classification

Boosting:

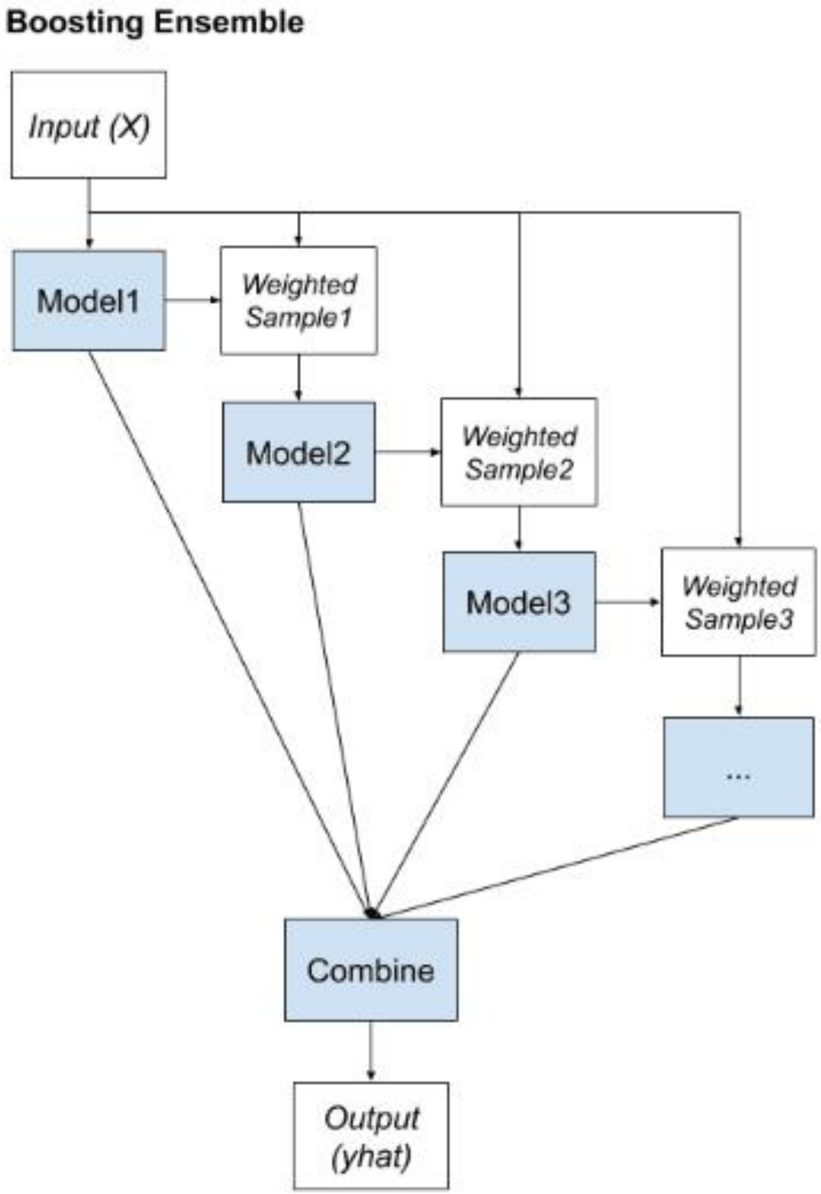
- Boosting is a self-learning technique.
- It learns by assigning a weight for various items in the data. The boosting technique initially starts with equal weights but after every model, each model is assigned a weight based on its performance.
- Similarly, after the evaluation of each model, the misclassified data are given more weight so that the next model has more focus on these items.
- For every iteration,
  - It weights each training example by how incorrectly it was classified.
  - Makes a hypothesis
  - Weights the hypothesis
  - Thus the final model is derived from various models which focuses on various groups of data, by voting them based on their weights.

- The boosting method

  1. Let d1, d2, d3 be three learning algorithms for a particular task. Let a large training set X be given.
  2. We randomly divide X into three sets, say X1, X2, X3
  3. We use X1 and train d1.
  4. We then take X2 and feed it to d1.
  5. We take all instances misclassified by d1 and also as many instances on which d1 is correct from X2, and these together form the training set of d2.
  6. We then take X3 and feed it to d1 and d2.

7. The instances on which d1 and d2 disagree form the training set of d3.
8. During testing, given an instance, we give it to d1 and d2 if they agree, that is the response; otherwise the response of d3 is taken as the output.

- The final model is averaged by using the weighted average method
- e= $((\boldsymbol{\Sigma}\ e_i w_i)/{*}\boldsymbol{\Sigma}\ w_i))/n$
- where, $e_1, e_2 \ldots.. e_n$ = base classifier
- $w_1, w_2 \ldots.. w_n$ =weights
- n=no. of models
- e = final classifier
- Boosting algorithms:
  - AdaBoost
  - GBM
  - XGBM



Boosting Ensemble

- In bagging, generating complementary base-learners is left to chance and to the unstability of the learning method.
- In boosting, we actively try to generate complementary base-learners by training the next learner on the mistakes of the previous learners.
- The original boosting algorithm combines three weak learners to generate a strong learner.
  - A weak learner has error probability less than 1/2, which makes it better than random guessing on a two-class problem, and a strong learner has arbitrarily small error probability

THANK YOU