Data Mining :: Unit-3

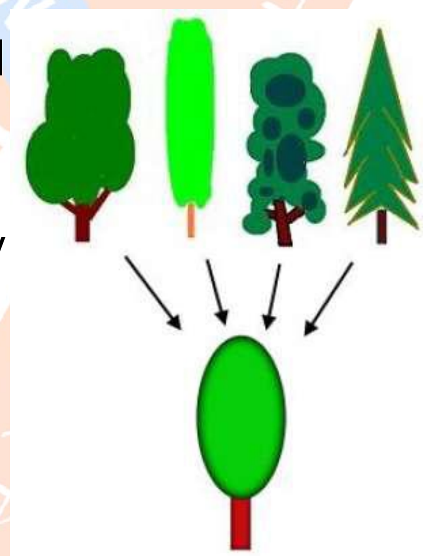
Classification Issues – Overfitting, Validation, Model Evaluation

Er. Dinesh Baniya Kshatri
(Lecturer)

Department of Electronics and Computer Engineering
Institute of Engineering, Thapathali Campus

Generalization

- The goal of machine learning model is to maximize the **generalization** ability:
  - Needs to perform well on previously unobserved inputs
    - Training data results in training error
    - Testing data results in testing error (generalization error)



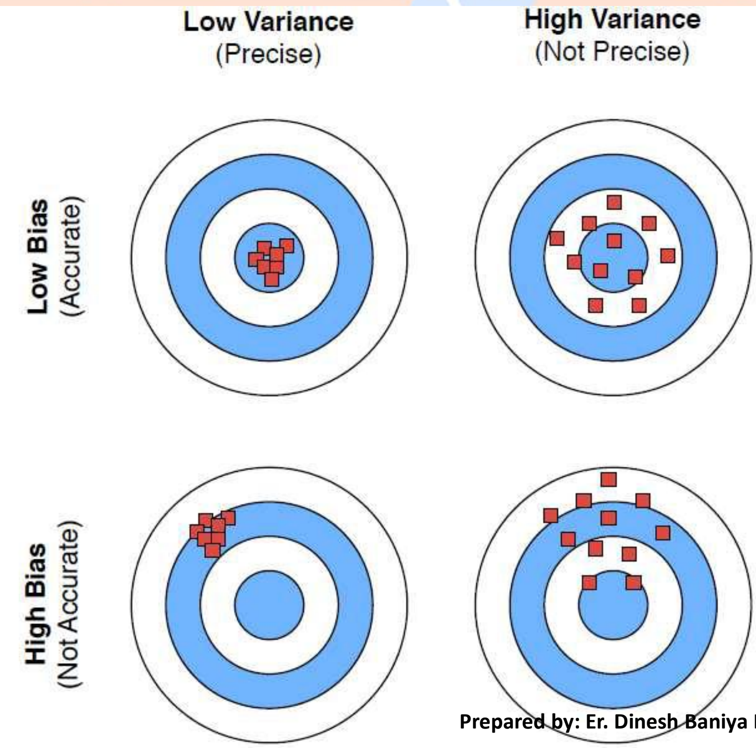
Training / Testing Data Split

- **Training Data:**
  – Is used to fit parameters of a classifier
- **Testing Data:**
  – Is used to assess how a classifier generalizes to new data

Underfitting vs Overfitting

- **Underfitting:**
  – Model does not fit training data well enough
  – Model does not capture the underlying structure and hence performs poorly
  – Results in excessively simple model
- **Overfitting:**
  – Model fits training data too well
  – A model with zero or very low training error is likely to perform well on the training data but generalize badly
  – Results in excessively complicated model

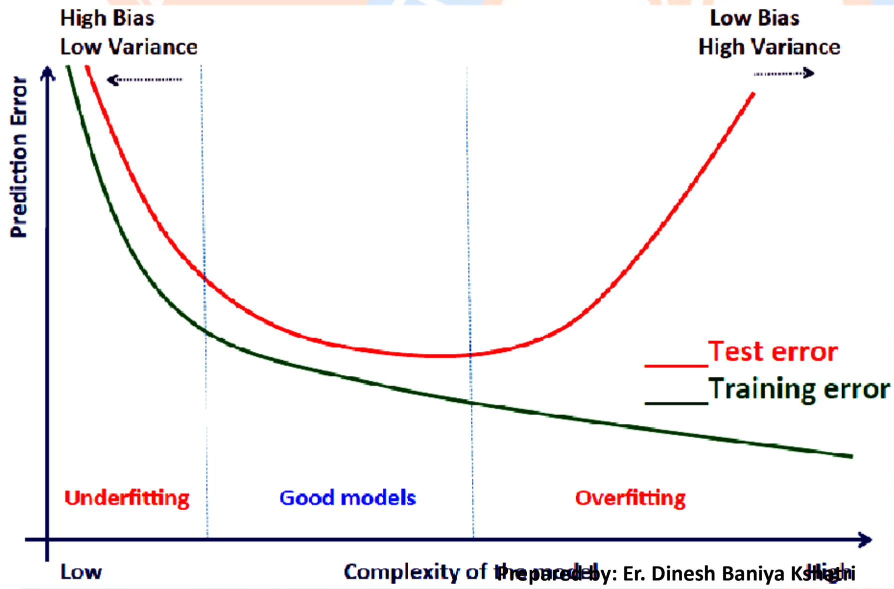
Bias-Variance Intuition



Bias-Variance Tradeoff

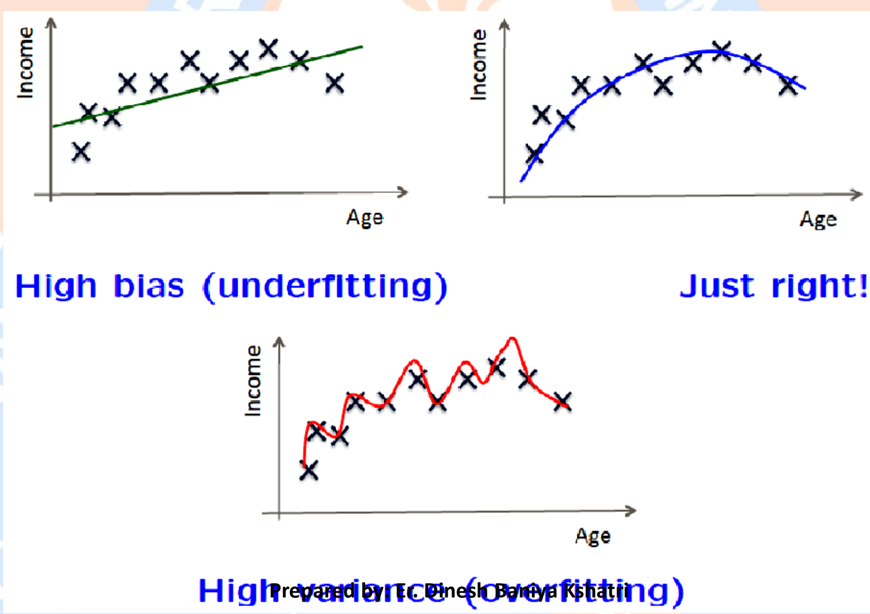
- **Bias:**
  – Measures the quality of the model family
  – Models with high bias pay little attention to the training data and are overly simplistic **(Underfitting)**
- **Variance:**
  – Adaptability of a model to new training data
  – Models with high variance pay too much attention to the training data, are overly complicated, and do not generalize well to future data **(Overfitting)**

Graphical Representation
(Bias-Variance Tradeoff)



- Both underfitting and overfitting lead to poor predictions on test data and they do not generalize well

Illustration of Underfitting & Overfitting



Training Error and Testing Error

- **Test error**: Prediction error over an independent sample.
- **Training error**: Average loss over the training samples

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}(\mathbf{x}_i))$$

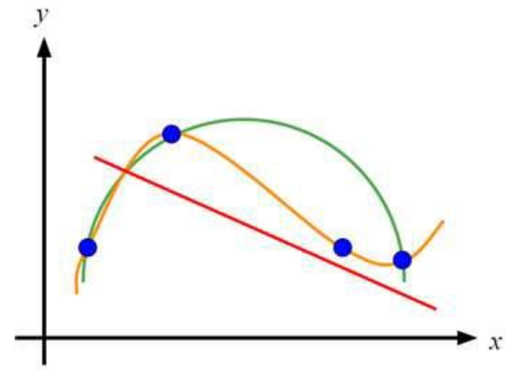
- As the model gets more complex it infers more information from the training data to represent more complicated underlying structures.

Single Training / Testing Partition (Limitations)

- **There may not be enough data to make sufficiently large training and testing datasets:**
  - A larger test set gives more reliable estimate of accuracy (lower variance estimate)
  - However, a larger training set will be more representative of the data that is actually used in the learning process

Diagnosing Underfitting / Overfitting – [1]
(Training Data)

- Want to fit a polynomial
- Instead of fixing polynomial degree, make it parameter **d**
  - learning machine $f(\mathbf{x},\mathbf{y},\mathbf{d})$
- Consider just three choices for **d**
  - degree 1
  - degree 2
  - degree 3



- Training error is a bad measure to choose **d**
  - degree 3 is the best according to the training error, but overfits the data

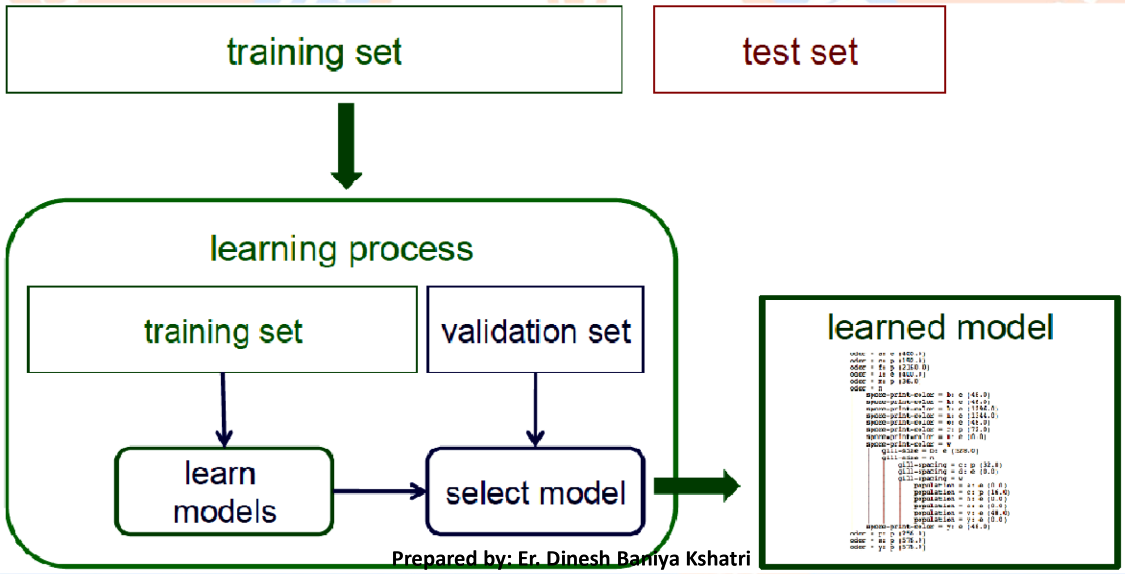
Diagnosing Underfitting / Overfitting – [2]
(Validation Data)

- **The polynomial degree example can be looked at as choosing among 3 classifiers (degree 1, 2, or 3)**
- **Split the labeled data into three parts:**

labeled data

Training ≈60%	Validation ≈20%	Test ≈20%
train tunable parameters **w**	train other parameters, or to select classifier	use **only** to assess final performance

Diagnosing Underfitting / Overfitting – [3]
(Model Learning Process)

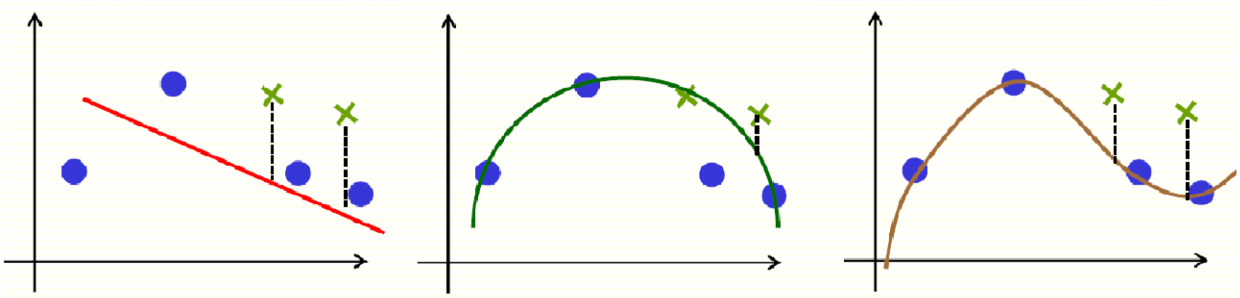


Diagnosing Underfitting / Overfitting – [4]
(Training / Validation / Testing Error)

labeled data

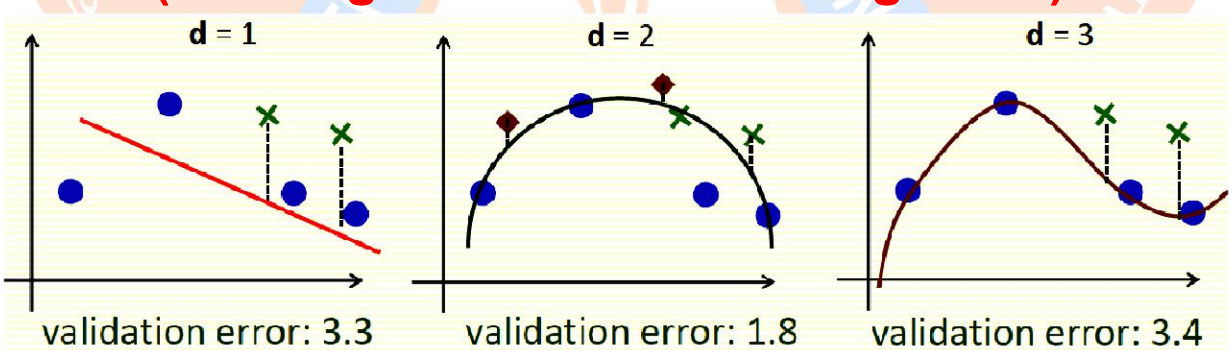
Training ≈60%	Validation ≈20%	Test ≈20%
Training error: computed on training examples	Validation error: computed on validation examples	Test error: computed on test examples

Diagnosing Underfitting / Overfitting – [4]
(Training and Validation Error)



Underfitting	Just Right	Overfitting
• large training error	• small training error	• small training error
• large validation error	• small validation error	• large validation error

Diagnosing Underfitting / Overfitting – [5]
(Training / Validation / Testing Error)



- Training Data
- Validation Data
  - **d** = 2 is chosen

- Test Data
  - 1.3 test error computed for **d** = 2

Cautious use of Testing Data

- Test set **should not** be used to tune your network
  - Network architecture
  - Number of layers
  - Hyper-parameters
- Failing to do so will overfit the network to your test set!

Train / Validate / Test Method
(Strengths and Weaknesses)

- Good news
  - Very simple
- Bad news:
  - Wastes data
    - in general, the more data we have, the better are the estimated parameters
    - we estimate parameters on 40% less data, since 20% removed for test and 20% for validation data
  - If we have a small dataset our test (validation) set might just be lucky or unlucky
- **Cross Validation is a method for performance evaluation that wastes less data**

Cross Validation

Cross validation is a way to estimate how well your algorithm will generalize.

But…

- make sure that the test data is drawn from the same distribution as the training data. (shuffle your training data before splitting them up)
- to see how well your algorithm might generalize, set aside 20% of your training data and use it as fake test data (called **validation set**).
- Remember to **never touch your test data**!

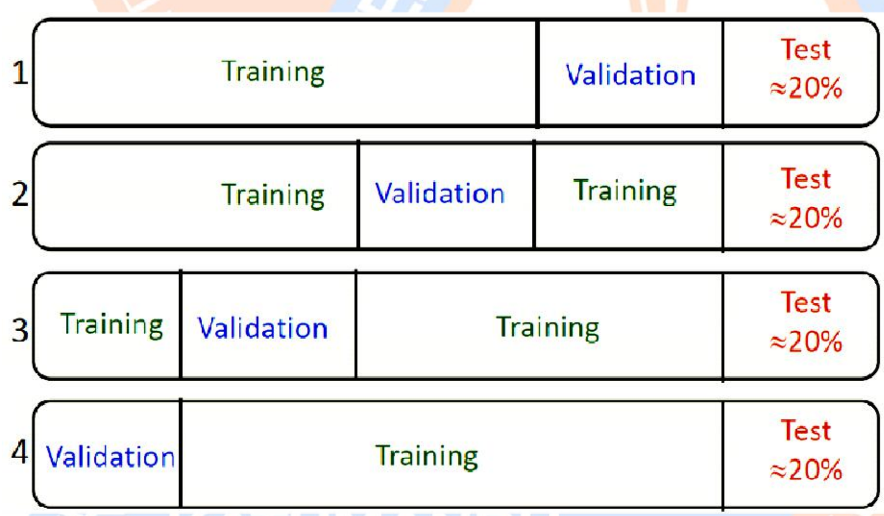
Types of Cross Validation

- **Leave-One-Out Cross Validation**
- **K-Fold Cross Validation**
- **Bootstrap Cross Validation**

Leave-One-Out Cross Validation

- Use all but one sample for training and assess performance on the excluded sample
  - Hold out one example, train on remaining examples
- For a data set with n samples, leave-one-out cross-validation is equivalent to n-fold cross-validation
- Not suitable if data set is very large and/or training the classifier takes a long time
  - Use if less than 100 samples (rough estimate)

K-Fold Cross Validation – [1]



- **Create multiple splits of training & validation**
- **Split data set into (k) equally large validation parts**
- **Average the results over the splits**

K-Fold Cross Validation – [2]

labeled data set

s$_1$	s$_2$	s$_3$	s$_4$	s$_5$

partition data into *n* subsamples

iteratively leave one subsample out for the test set, train on the rest

iteration	train on	test on
1	s$_2$ s$_3$ s$_4$ s$_5$	s$_1$
2	s$_1$ s$_3$ s$_4$ s$_5$	s$_2$
3	s$_1$ s$_2$ s$_4$ s$_5$	s$_3$
4	s$_1$ s$_2$ s$_3$ s$_5$	s$_4$
5	s$_1$ s$_2$ s$_3$ s$_4$	s$_5$

K-Fold Cross Validation – [3]

Suppose we have 100 instances, and we want to estimate accuracy with cross validation

iteration	train on	test on	correct
1	s$_2$ s$_3$ s$_4$ s$_5$	s$_1$	11 / 20
2	s$_1$ s$_3$ s$_4$ s$_5$	s$_2$	17 / 20
3	s$_1$ s$_2$ s$_4$ s$_5$	s$_3$	16 / 20
4	s$_1$ s$_2$ s$_3$ s$_5$	s$_4$	13 / 20
5	s$_1$ s$_2$ s$_3$ s$_4$	s$_5$	16 / 20

Accuracy = 73/100 = 73 %

Bootstrap Cross Validation – [1]

- **Bootstrap Sampling:**
  - Randomly draw samples with replacement from the original data set to generate new data sets of the same size
  - Sampling is repeated (B) times and samples not included in each bootstrap sample are recorded
    - Train model on each of the *B* bootstrap samples
    - For each sample of the original data set, asses performance only on bootstrap samples not containing the particular sample

Bootstrap Cross Validation – [2]

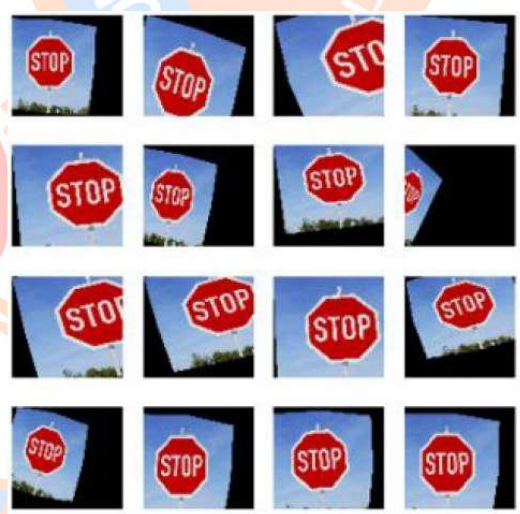
- Suppose that we have a dataset with 1000 points
- Sample randomly 1000 points from the dataset **with replacement**
  - Run the classifier on this bootstrapped sample
  - Test on the unselected data points
  - Repeat process certain times

Fixing Underfitting / Overfitting

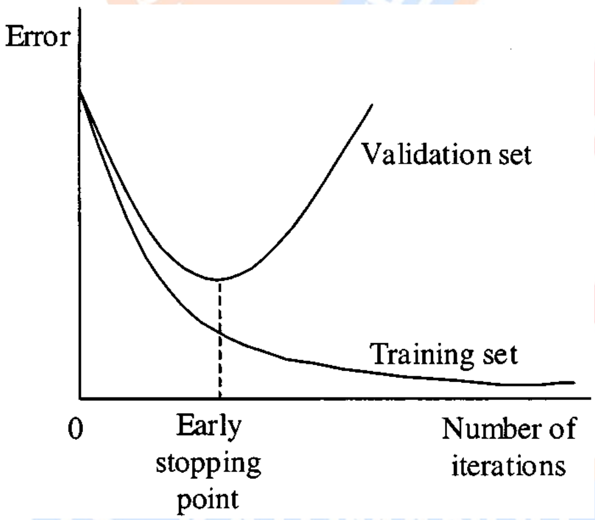
- **Fixing Underfitting**
  - Increase number of features
  - Try more complex classifier
- **Fixing Overfitting**
  - Try smaller feature set
  - Use less complex classifier
  - Getting more training examples (data augmentation)
  - Use early stopping during training
  - Perform dropout

Prevent Overfitting
(Data Augmentation)

- **Modify input samples artificially to increase the data size**
  - Inject Noise
  - Perform transformations
    - Flipping, translation, rotation, scaling, changing brightness



Prevent Overfitting
(Early Stopping)



- Beyond the early stopping point, improving the model’s fit to the training data leads to increased generalization error
- Early stopping rules provide guidance as to how many iterations can be run before the model begins to overfit

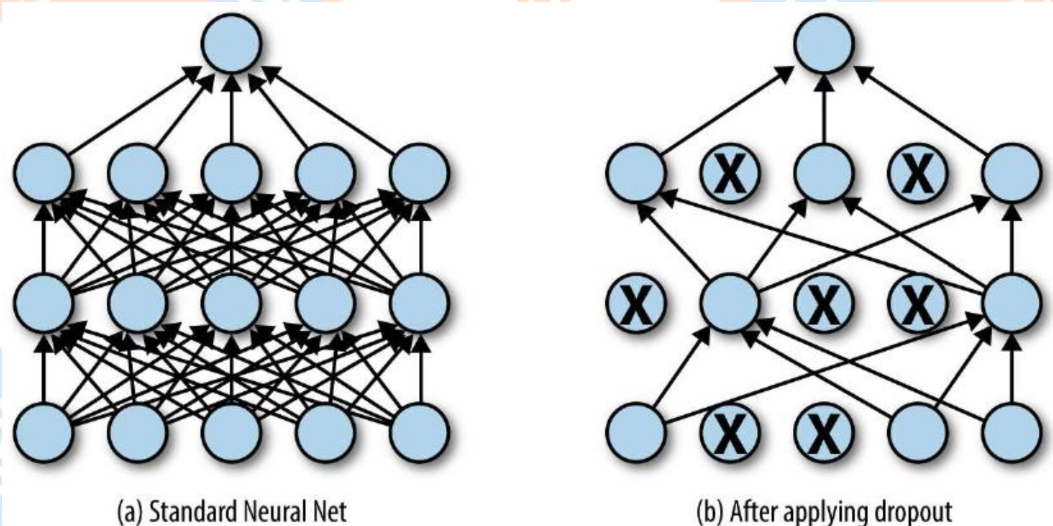
Prepared by: Er. Dinesh Baniya Kshatri 29

Prevent Overfitting
(Dropout) – [1]

- **Dropout modifies the network itself:**
  – It randomly drops neurons from the neural network during training in each iteration
  – Dropping different sets of neurons is equivalent to training different neural networks
  – Different networks will overfit in different ways, so the net effect of dropout will be to reduce overfitting

Prepared by: Er. Dinesh Baniya Kshatri 30

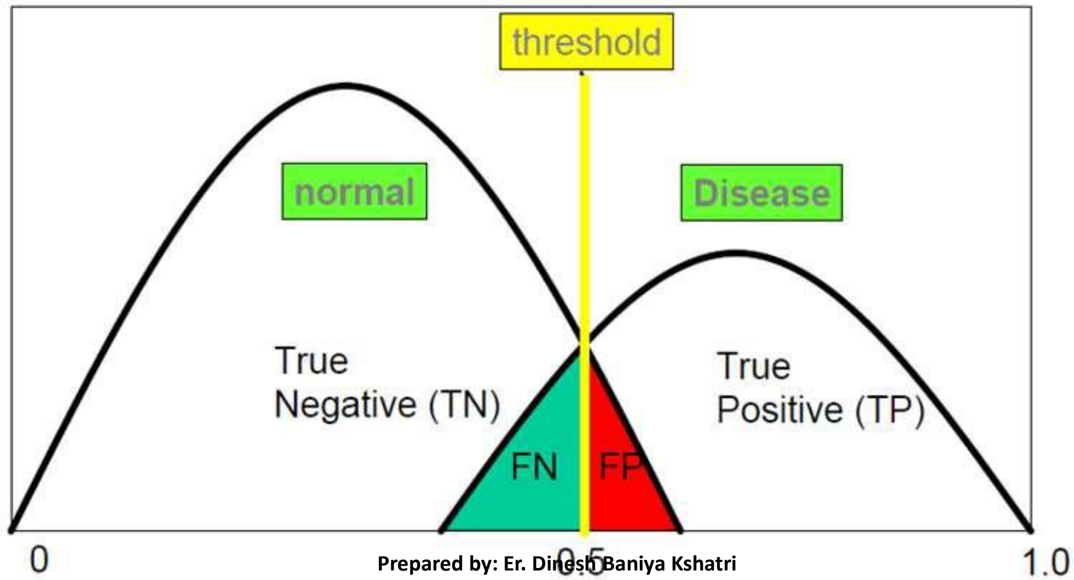
Prevent Overfitting
(Dropout) – [2]



Classifier Evaluation Metrics
(TP, FP, TN, FN)

- **With respect to class (c), a prediction is defined as:**
  - True Positive: (Hit)
    - The label is (c) and the classifier predicted (c)
  - False Positive: (False Alarm)
    - The label is not (c) but the classifier predicted (c)
  - True Negative: (Correct Rejection)
    - The label is not (c) and the classifier did not predict (c)
  - False Negative: (Miss)
    - The label is (c) but the classifier did not predict (c)

Classifier Evaluation Metrics
(Visualizing TN, TP, FN, FP)



Classifier Evaluation Metrics
(Error Types) – [1]

- Two different types of errors:
  - False Positive ("Type I" Error)
  - False Negative ("Type II" Error)

- Usually there is a tradeoff between these two
  - Can optimize for one at the expense of the other
  - Which one to favor? Depends on task

Classifier Evaluation Metrics
(Error Types) – [2]

E.g. Consider the diagnostic of a disease. Two types of mis-diagnostics:

- Patient does not have disease but received positive diagnostic (Type I error)
- Patient has disease but it was not detected (Type II error)

E.g. Consider the problem of spam classification:

- A message that is not spam is assigned to the spam folder (Type I error)
- A message that is spam appears in the regular folder (Type II error) .

Classifier Evaluation Metrics
(Confusion Matrix for Two Class Problem)

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

Classifier Evaluation Metrics
(Confusion Matrix for Multi-Class Problem)

True label \ Predicted label	V	O	T	E
V	56.13	31.61	5.16	7.10
O	10.77	63.08	10.77	15.38
T	0.00	18.75	50.00	31.25
E	0.00	3.70	3.70	92.59

		Ground Truth			
		Class A	Class B	Class C	Class D
Prediction	Class A	Correct	Wrong	Wrong	Wrong
	Class B	Wrong	Correct	Wrong	Wrong
	Class C	Wrong	Wrong	Correct	Wrong
	Class D	Wrong	Wrong	Wrong	Corrent

Classifier Evaluation Metrics
(Confusion Matrix for Multi-Class Problem)

- With (k) classes confusion matrix becomes a (k × k) matrix
- Choose one of (k) classes as positive (e.g.: class A)
  - Collapse all other classes into negative to obtain (k) different (2 × 2) matrices

		Ground Truth	
		Class A	Other
Pred.	Class A	True positive	False positive
	Other	False negative	True negative

Classifier Evaluation Metrics
(TPR, FPR, FNR, TNR)

- true positive rate: $tpr = \frac{tp}{tp + fn}$
  - percentage of *correctly* classified *positive* examples
- false positive rate: $fpr = \frac{fp}{fp + tn}$
  - percentage of negative examples *incorrectly* classified *as positive*
- false negative rate: $fnr = \frac{fn}{tp + fn} = 1 - tpr$
  - percentage of positive examples *incorrectly* classified *as negative*
- true negative rate: $tnr = \frac{tn}{fp + tn} = 1 - fpr$

Prepared by: Er. Dinesh Baniya Kshatri 39

Classifier Evaluation Metrics
(Sensitivity)

- **Sensitivity: True Positive Recognition Rate**
  - A sensitive classifier is one which almost always finds everything it is looking for, i.e. it has high recall

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

Prepared by: Er. Dinesh Baniya Kshatri 40

Classifier Evaluation Metrics
(Specificity)

- **Specificity: True Negative Recognition Rate**
  - A specific classifier is one that does a good job not finding the things that it doesn't want to find

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

Classifier Evaluation Metrics
(Accuracy & Error Rate) – [1]

- Accuracy refers to the number of correctly classified examples divided by the total number of examples

- Error Rate = 1 – Accuracy

Classifier Evaluation Metrics
(Accuracy & Error Rate) – [2]

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

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

Prepared by: Er. Dinesh Baniya Kshatri 43

Class Exercise
(Question – 1)

- **Prove that accuracy is a function of sensitivity and specificity:**

$$accuracy = sensitivity\frac{P}{(P+N)} + specificity\frac{N}{(P+N)}$$

Prepared by: Er. Dinesh Baniya Kshatri 44

Class Exercise
(Solution to Question – 1)

$$\begin{aligned} \text{accuracy} &= \frac{TP+TN}{(P+N)} \\ &= \frac{TP}{(P+N)} + \frac{TN}{(P+N)} \\ &= \frac{TP}{(P+N)} \times \frac{P}{P} + \frac{TN}{(P+N)} \times \frac{N}{N} \end{aligned}$$

Classifier Evaluation Metrics
(Recall) – [1]

- Number of correctly classified positive examples divided by the total number of positive examples
- Recall also refers to completeness
  – What percent of positive tuples did the classifier label as positive?
- High recall means that a class is correctly recognized (small number of FN)

Classifier Evaluation Metrics
(Recall) – [2]

Recall is the percentage of positive instances that were predicted to be positive

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

Fraud example:

- Low recall means there are fraudulent transactions that you aren't detecting

Prepared by: Er. Dinesh Baniya Kshatri 47

Classifier Evaluation Metrics
(Precision) – [1]

- Number of correctly classified positive examples divided by the total number of predicted positive examples
- Precision also refers to exactness
  - What percent of tuples that the classifier labeled as positive are actually positive?
- High precision means that a class labeled as positive is indeed positive (small number of FP)

Prepared by: Er. Dinesh Baniya Kshatri 48

Classifier Evaluation Metrics
(Precision) – [2]

Precision is the percentage of instances predicted to be positive that were actually positive

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

Fraud example:

- Low precision means you are classifying legitimate transactions as fraudulent

Classifier Evaluation Metrics
(Precision and Recall)

- **High Recall, Low Precision:**
  - Most of the positive examples are correctly recognized (low FN) but there are a lot of false positives (high FP)
- **Low Recall, High Precision:**
  - Miss a lot of positive examples (high FN) but those predicted as positive are indeed positive (low FP)

Classifier Evaluation Metrics
(F1 Score)

- F1 score is the harmonic mean of positive predictive value and sensitivity
- Harmonic mean favors systems that achieve equal precision and recall, i.e., when PRE=REC, then F1=PRE=REC
  - Both numbers have to be high for F1 to be high
  - F1 is therefore useful when both are important

$$F1 = 2\frac{PRE \times REC}{PRE + REC}$$

Class Exercise
(Question – 2)

- **Harmonic mean of the positive real numbers, x1, x2, . . . , xn is defined as:**

$$\begin{aligned} H &= \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \ldots + \frac{1}{x_n}} \\ &= \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} \end{aligned}$$

- **Use this fact to derive the equation for F1-score**

Class Exercise
(Solution to Question – 2)

$$F = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$
$$= \frac{2 \times precision \times recall}{precision + recall}$$

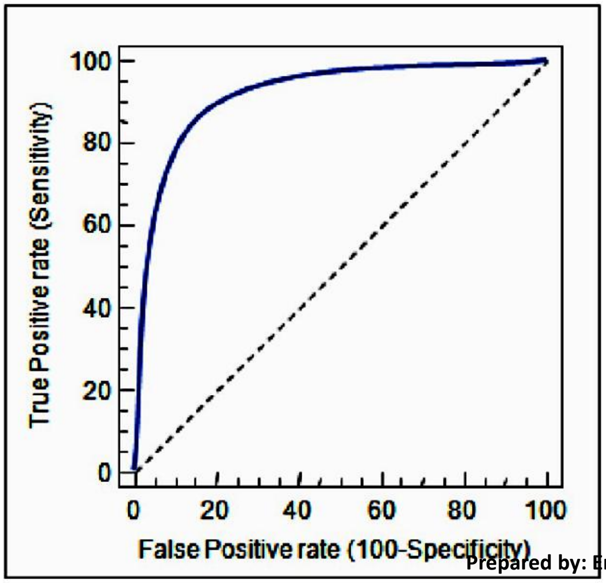
Prepared by: Er. Dinesh Baniya Kshatri 53

Classifier Evaluation Metrics
(Receiver Operating Characteristic Curve) – [1]

- It is a performance graphing method.
- A plot of True Positive Rate (TPR) and False Positive Rate (FPR)
- Used for evaluating data mining schemes, and comparing the relative performance among different classifiers.

Prepared by: Er. Dinesh Baniya Kshatri 54

Classifier Evaluation Metrics
(Receiver Operating Characteristic Curve) – [2]

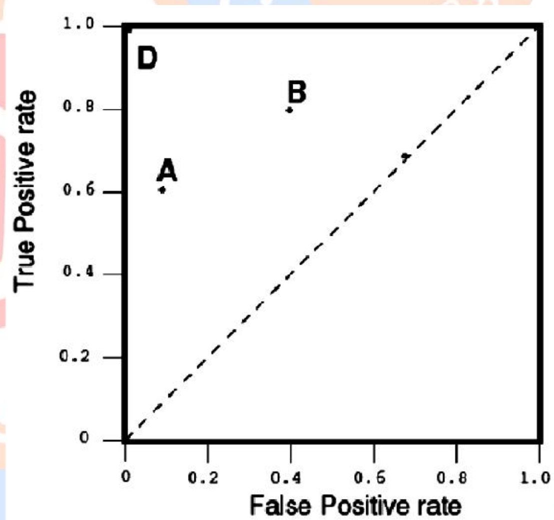


- TPR is plotted on the Y axis
- FPR is plotted on the X axis
- Depicts relative trade-offs between
  - Benefits (True Positives)
  - Costs (False Positives)

Prepared by: Er. Dinesh Baniya Kshatri 55

Classifier Evaluation Metrics
(ROC Space - Upper Triangle) – [3]

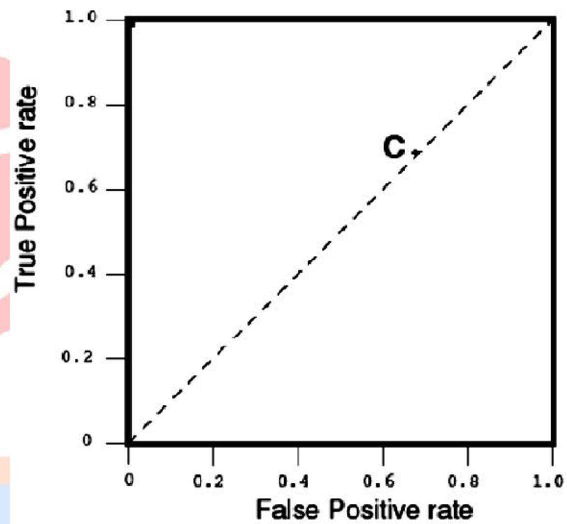
- A point in ROC space is better than another if it is to the upper left corner
- Point (D) is better than points (A) and (B)



Prepared by: Er. Dinesh Baniya Kshatri 56

Classifier Evaluation Metrics
(ROC Space - Random Performance) – [4]

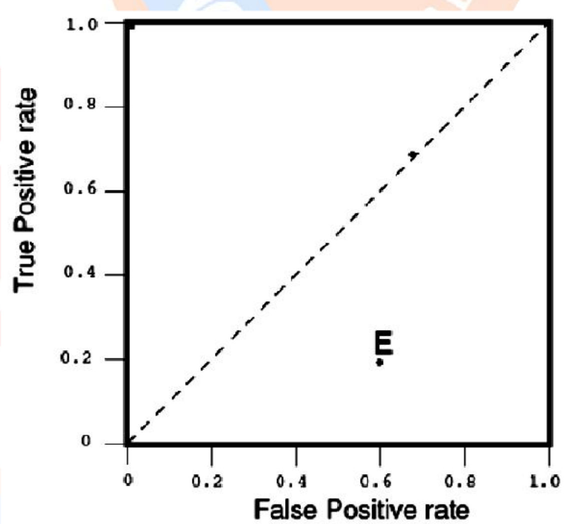
- The diagonal line y = x represents the strategy of randomly guessing a class
- Point C's performance is virtually random



Prepared by: Er. Dinesh Baniya Kshatri 57

Classifier Evaluation Metrics
(ROC Space - Lower Triangle) – [5]

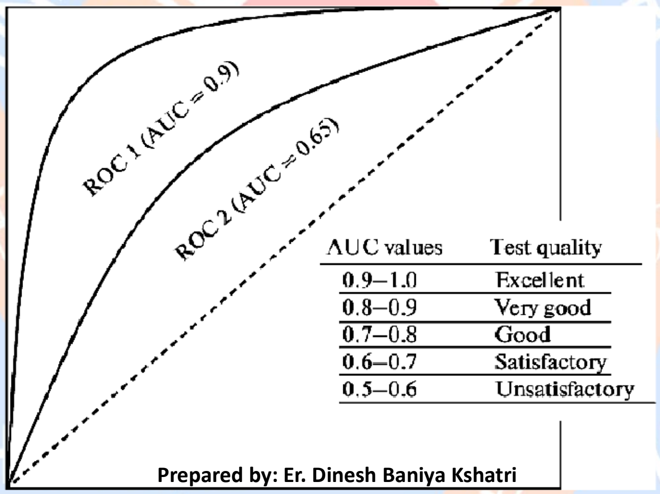
- Any classifier that appears in the lower right triangle (Point E) performs worse than random guessing
- This triangle should therefore be usually empty in ROC graphs



Prepared by: Er. Dinesh Baniya Kshatri 58

Classifier Evaluation Metrics
(Area under ROC Curve – AUC)

- The bigger the AUC the better



Class Exercise
(Question – 3)

- **For each tuple, compute the following:**
  - True positives (TP), false positives (FP), true negatives (TN), and false negatives (FN)
  - True positive rate (TPR) and false positive rate (FPR)
- **Plot the ROC curve for the data**

Tuple #	*Class*	*Prob.*
1	p	0.95
2	n	0.85
3	p	0.78
4	p	0.66
5	n	0.60
6	p	0.55
7	n	0.53
8	n	0.52
9	n	0.51
10	p	0.4

Class Exercise
(Solution to Question – 3) – [1]

Tuple #	*Class*	*Prob.*	TP	FP	TN	FN	TPR	FPR
1	p	0.95	1	0	5	4	0.2	0
2	n	0.85	1	1	4	4	0.2	0.2
3	p	0.78	2	1	4	3	0.4	0.2
4	p	0.66	3	1	4	2	0.6	0.2
5	n	0.60	3	2	3	2	0.6	0.4
6	p	0.55	4	2	3	1	0.8	0.4
7	n	0.53	4	3	2	1	0.8	0.6
8	n	0.52	4	4	1	1	0.8	0.8
9	n	0.51	4	5	0	1	0.8	1
10	p	0.4	5	5	0	0	1	1

Class Exercise
(Solution to Question – 3) – [2]

- **Case-1 (Assume Threshold = 0.95)**
  - **First** tuple is predicted to be positive
  - From table, actual class label of first tuple is positive
  - So, TP = 1 and FP = 0
  - Remaining **nine** tuples are predicted as negative
  - Actual class labels of remaining **nine** tuples consists of:
    - 5 negative and 4 positive
  - So, TN = 5 and FN = 4

Class Exercise
(Solution to Question – 3) – [3]

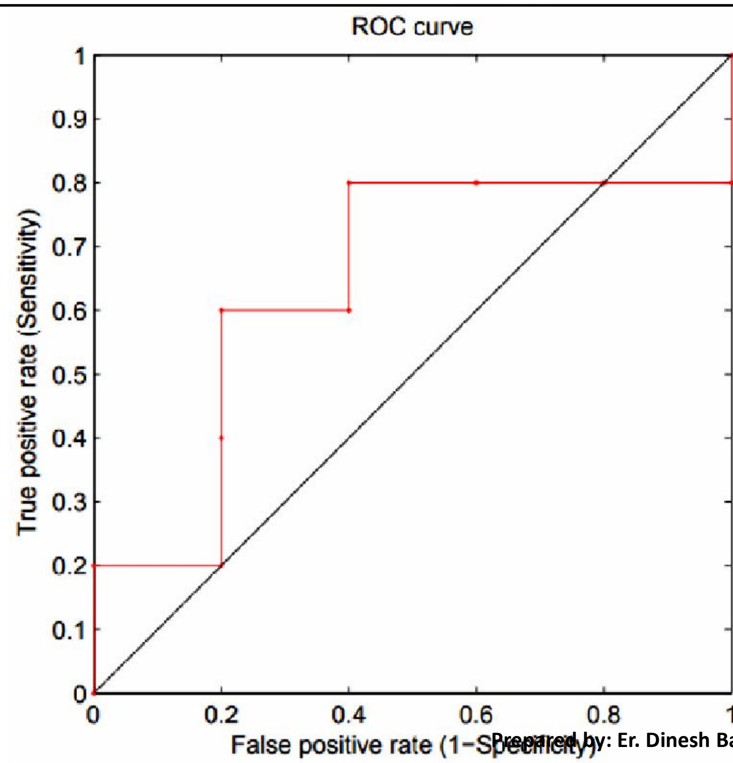
- **Case-2 (Assume Threshold = 0.85)**
  - **First** and **Second** tuples are predicted to be positive
  - From table, actual class labels of the **first two** tuples are:
    - 1 positive and 1 negative
  - So, TP = 1 and FP = 1
  - Remaining **eight** tuples are predicted as negative
  - From table, actual class labels of remaining **eight** tuples are:
    - 4 negative and 4 positive
  - So, TN = 4, FN = 4

Class Exercise
(Solution to Question – 3) – [4]

- **Case-3 (Assume Threshold = 0.78)**
  - **First**, **Second** and **Third** tuples are predicted to be positive
  - From table, actual class labels of the **first three** tuples are:
    - 2 positive and 1 negative
  - So, TP = 2 and FP = 1
  - Remaining **seven** tuples are predicted as negative
  - From table, actual class labels of remaining **seven** tuples are:
    - 4 negative and 3 positive
  - So, TN = 4, FN = 3

Class Exercise
(Solution to Question – 3) – [5]

- **Case-10 (Assume Threshold = 0.4)**
  - **<u>All Ten</u>** tuples are predicted to be positive
  - From table, actual class labels of **<u>all ten</u>** tuples are:
    - 5 positive and 5 negative
  - So, TP = 5 and FP = 5
  - There are **<u>zero</u>** tuples having threshold less than 0.4
  - So, TN = 0, FN = 0



Class Exercise
(Solution to Question – 3) – [6]

Classifier Evaluation Metrics
(Precision-Recall Curve) – [1]

- Compares precision (y-axis) to recall (x-axis)
- PR curve of optimal classifier is in the upper-right corner
- One point in PR space corresponds to a single confusion matrix
- Average precision is the area under the PR curve

- Note: Algorithms that optimize the area under the ROC curve are not guaranteed to optimize the area under the PR curve !!

Classifier Evaluation Metrics
(Precision-Recall Curve) – [2]