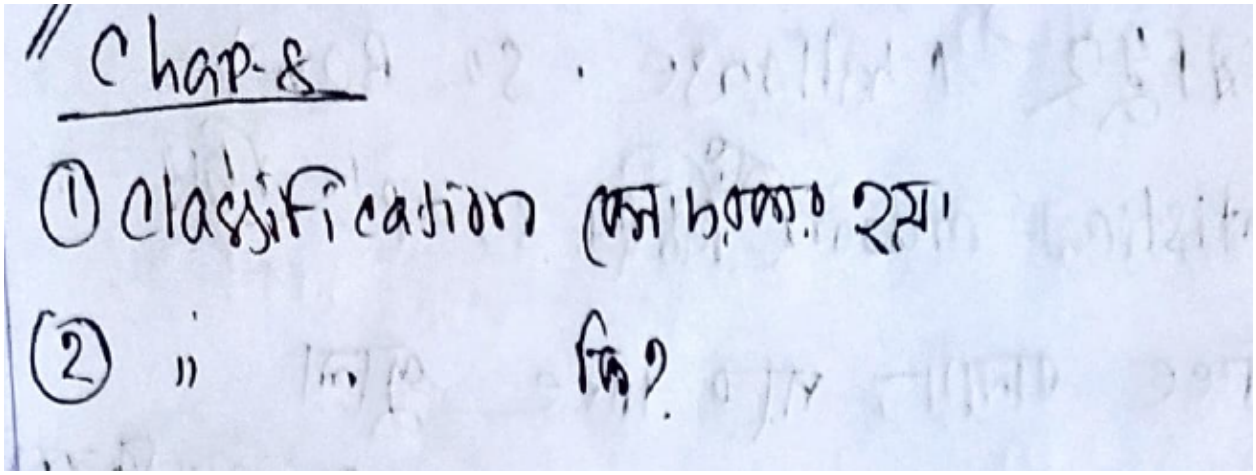
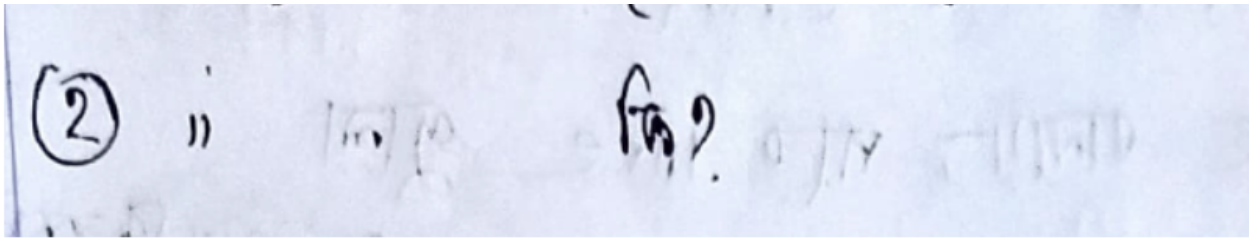
Classification is the process of categorizing data into predefined classes or categories. For



Classification is the process of categorizing data into predefined classes or categories. For example, email spam filters use classification to automatically label incoming emails as either "spam" or "non-spam" based on learned patterns.



Classification is sorting data into predefined categories. For example, spam filters categorize emails as "spam" or "non-spam" based on learned patterns.

ch – 8

classification → Decision Tree classification.
classification → Bayesian classifier.

classification → কোন data কোন class এ পড়ে।

যেমন: ব্যাংকের dataset এ থেকে loan নিলে safe না risky zone এ আছে এভাবে classify করা হয়।

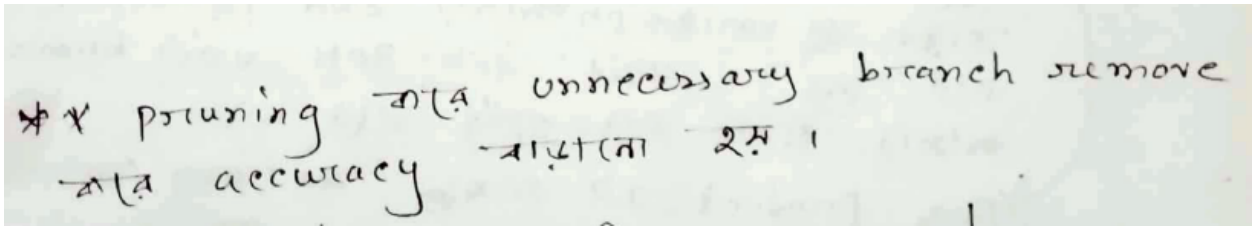
8.1.2 → Supervised Approach

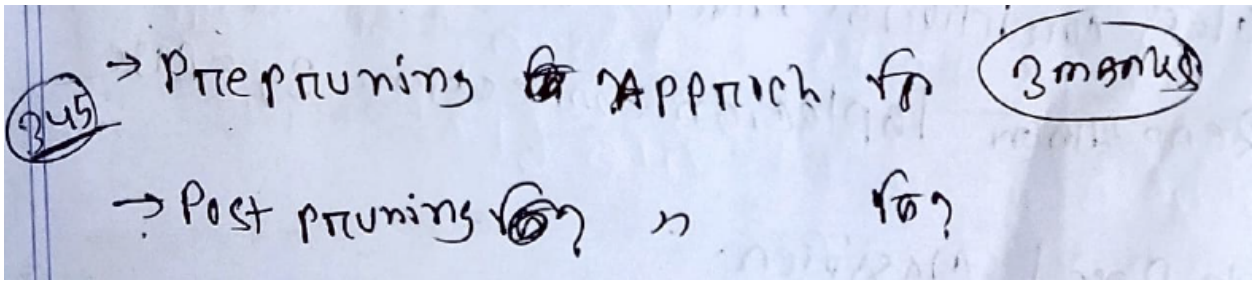
data দিয়ে machine কে learn করাইতে হবে। পরে নতুন data আসলে যে বলতে পারবে সে আসলে টা কী!

data tuple কে বলে, sample, example, data point, object.

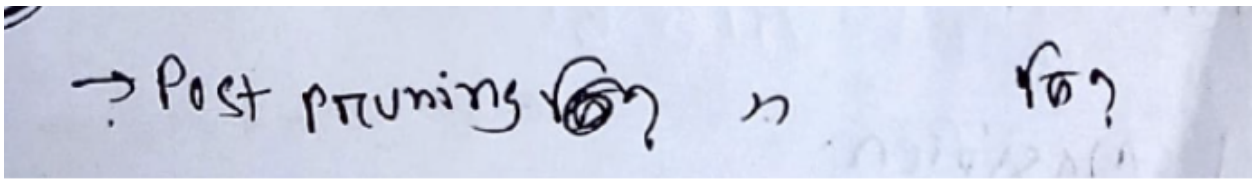
classification ২ ভাবে করা যায় →
১) supervised Learning
২) Unsupervised "

(৬) Tree pruning কেন করতে হয়?





Prepruning is a strategy in decision tree algorithms where the growth of the tree is restricted during training to prevent overfitting. It involves setting conditions, such as maximum depth or minimum samples, to control the complexity of the tree from the start.



Postpruning is a technique for decision trees where, after the tree is fully grown, certain branches are removed or collapsed to enhance generalization on new data. This helps prevent overfitting and improves the model's performance.

"*How does tree pruning work?*" There are two common approaches to tree pruning: *prepruning* and *postpruning*.

In the **prepruning** approach, a tree is "pruned" by halting its construction early (e.g., by deciding not to further split or partition the subset of training tuples at a given node). Upon halting, the node becomes a leaf. The leaf may hold the most frequent class among the subset tuples or the probability distribution of those tuples.

When constructing a tree, measures such as statistical significance, information gain, Gini index, and so on, can be used to assess the goodness of a split. If partitioning the tuples at a node would result in a split that falls below a prespecified threshold, then further partitioning of the given subset is halted. There are difficulties, however, in choosing an appropriate threshold. High thresholds could result in oversimplified trees, whereas low thresholds could result in very little simplification.

The second and more common approach is **postpruning**, which removes subtrees from a "fully grown" tree. A subtree at a given node is pruned by removing its branches and replacing it with a leaf. The leaf is labeled with the most frequent class among the subtree being replaced. For example, notice the subtree at node "A_3?" in the unpruned

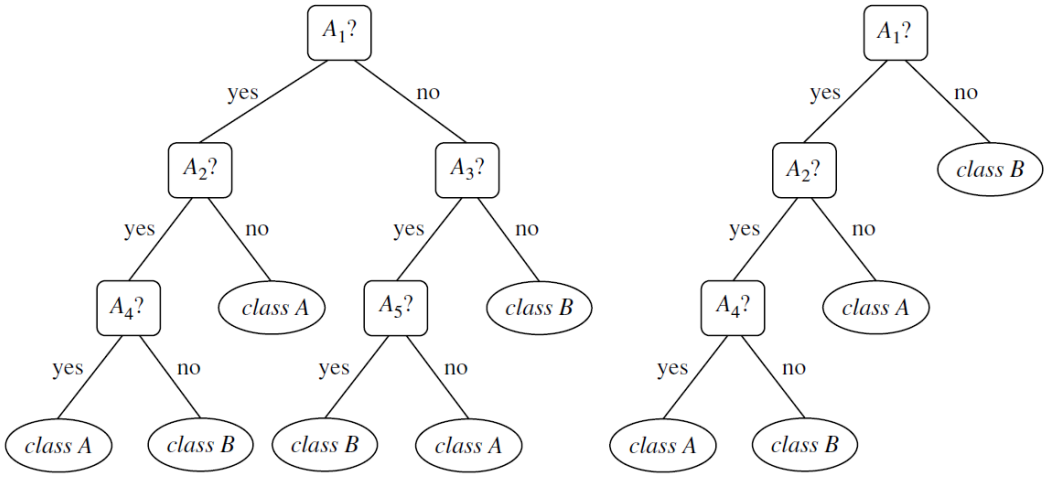
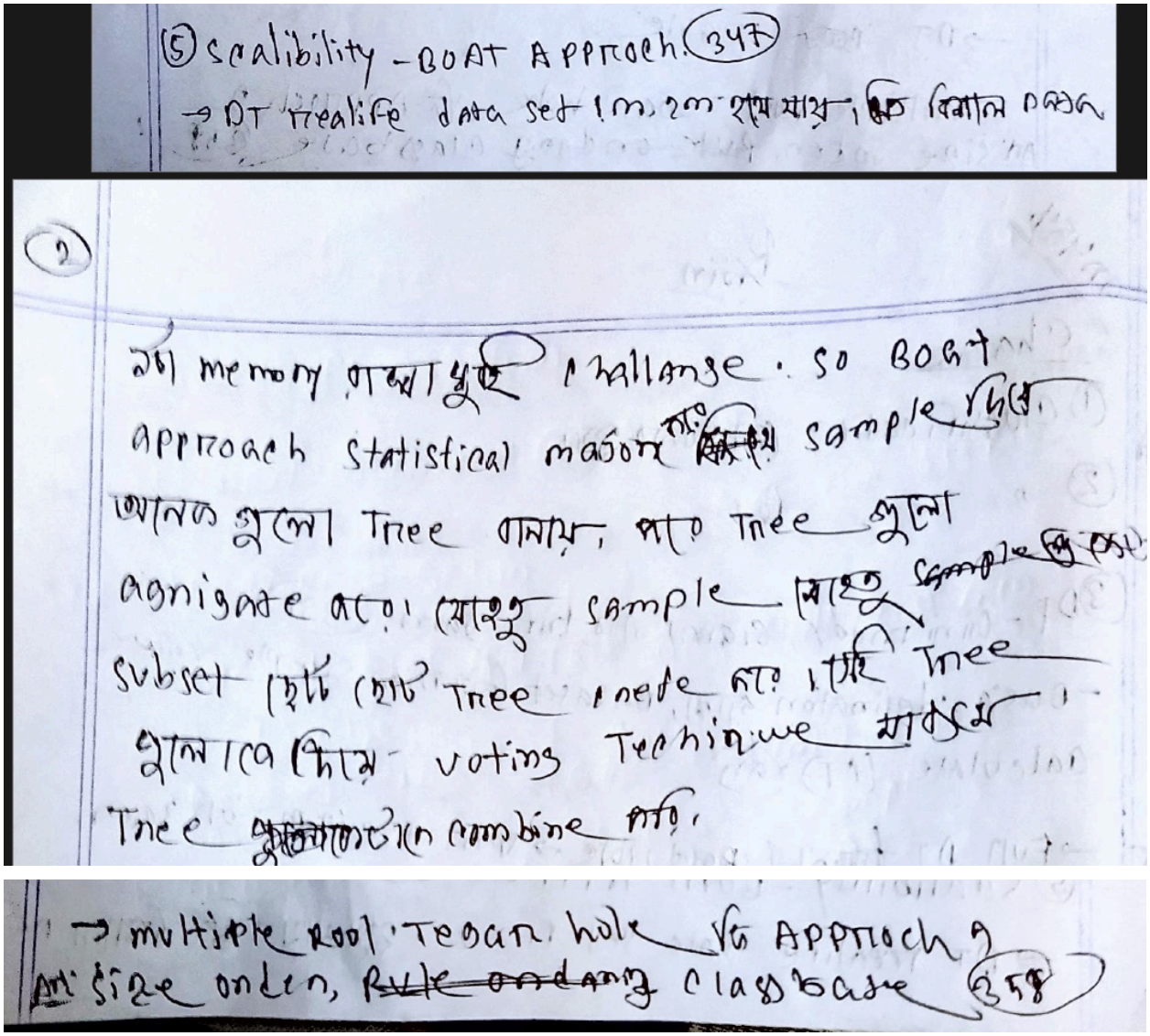
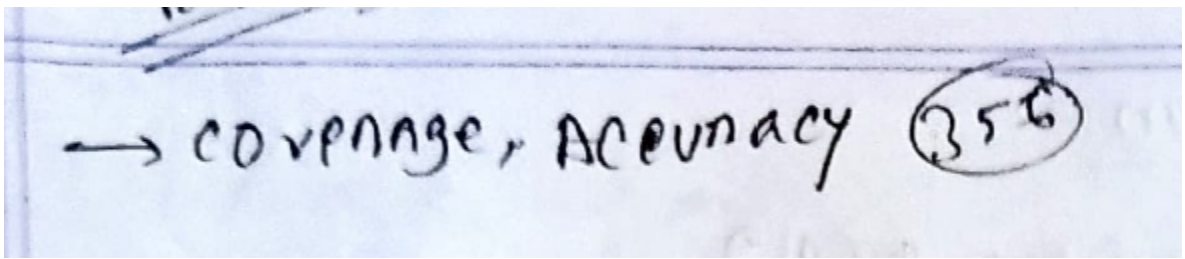


Figure 8.6 An unpruned decision tree and a pruned version of it.



The **size ordering** scheme assigns the highest priority to the triggering rule that has the "toughest" requirements, where toughness is measured by the rule antecedent *size*. That is, the triggering rule with the most attribute tests is fired.



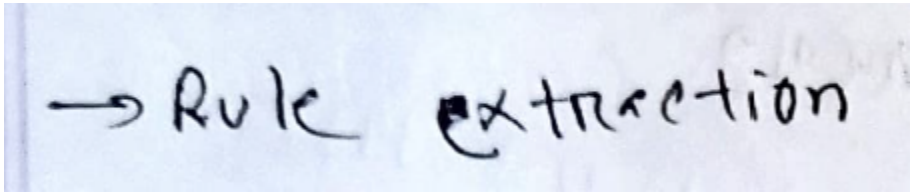
If the condition (i.e., all the attribute tests) in a rule antecedent holds true for a given tuple, we say that the rule antecedent is **satisfied** (or simply, that the rule is satisfied) and that the rule **covers** the tuple.

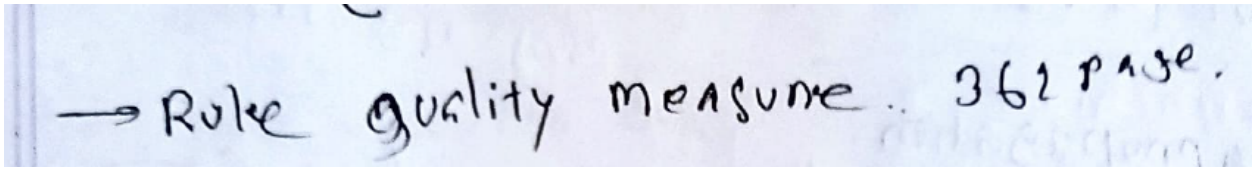
A rule R can be assessed by its coverage and accuracy. Given a tuple, $\boldsymbol{X}$, from a class-labeled data set, D, let n_{covers} be the number of tuples covered by R; $n_{correct}$ be the number of tuples correctly classified by R; and $|D|$ be the number of tuples in D. We can define the **coverage** and **accuracy** of R as

$$coverage(R) = \frac{n_{covers}}{|D|} \tag{8.16}$$

$$accuracy(R) = \frac{n_{correct}}{n_{covers}}. \tag{8.17}$$

That is, a rule's coverage is the percentage of tuples that are covered by the rule (i.e., their attribute values hold true for the rule's antecedent). For a rule's accuracy, we look at the tuples that it covers and see what percentage of them the rule can correctly classify.





overall at each step, rather than a single best candidate.

Rule Quality Measures

Learn_One_Rule needs a measure of rule quality. Every time it considers an attribute test, it must check to see if appending such a test to the current rule's condition will result in an improved rule. Accuracy may seem like an obvious choice at first, but consider Example 8.8.

Example 8.8 **Choosing between two rules based on accuracy.** Consider the two rules as illustrated in Figure 8.12. Both are for the class *loan_decision* = *accept*. We use "*a*" to represent the tuples of class "*accept*" and "*r*" for the tuples of class "*reject*." Rule *R*1 correctly classifies 38 of the 40 tuples it covers. Rule *R*2 covers only two tuples, which it correctly classifies. Their respective accuracies are 95% and 100%. Thus, *R*2 has greater accuracy than *R*1, but it is not the better rule because of its small coverage. ■

From this example, we see that accuracy on its own is not a reliable estimate of rule quality. Coverage on its own is not useful either—for a given class we could have a rule that covers many tuples, most of which belong to other classes! Thus, we seek other measures for evaluating rule quality, which may integrate aspects of accuracy and coverage. Here we will look at a few, namely *entropy*, another based on *information gain*, and a *statistical test* that considers coverage. For our discussion, suppose we are learning rules

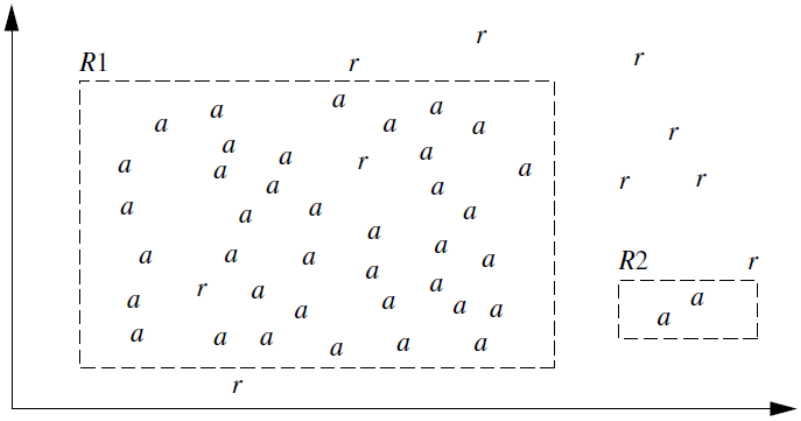
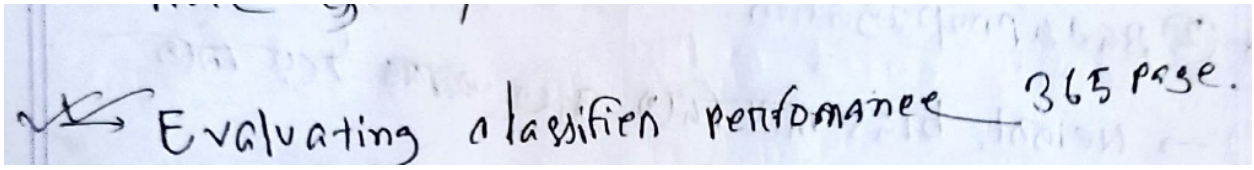


Figure 8.12 Rules for the class *loan_decision* = *accept*, showing *accept* (*a*) and *reject* (*r*) tuples.



8.5

Model evaluation

Accuracy বা recognition rate এর মাধ্যমে Model evaluation করা যায়.

page 364

positive plus / yes class / positive tuple
Negative " / No " / Negative tuple.

Multiple class এর ক্ষেত্রে সব class কে আমার target এর yes এর class এবং বাকিগুলো No class.

8.14

P → আমার actual dataset এ positive
P' → " Model এ positive
N → " actual dataset এ Negative
N' → " Model এ

$$\frac{6954+2588}{10000} = 95.42$$

$\therefore$ accuracy $= \frac{TP + TN}{P + N}$

Error Rate $= \frac{FP + FN}{P+N}$ $\quad$ 1 - accuracy

class imbalance problem → যানে data টা imbalance বা baised হয়ে থাকা data টা।

যেমন: 100 টা class এর 97 টা yes class আর 3 টা No class. কিন্তু আমি যদি বলি 97% accuracy দেয়, কিন্তু model যদি 3 টা No class কেও yes বলে। তাহলে accuracy ভালো আসে না। তাহলে বলা যায় class imbalance problem এর accuracy ভালো matrix না। তাই আসছে -

Sensitivity $= \frac{TP}{P}$ → যানে True positive ratio টাকে indicate করে।

যানে positive tuple গুলাকে correctly identify করতে পারছে

আর specificity $= \frac{TN}{N}$ → যানে Negative

tuple গুলোকে কতটুকু correctly identify করতে পারছি,

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

→ confusion matrix.

→ sensitivity, specificity. (367)

page 368

sensitivity আর specitivity প্রায় same হলে আমরা বলতে পারি যে model টা আমাদের ভালো perform করতেছে। একটা মানে অনেক ভালো আছে আর আরেকটা খারাপ আছে সেটা accurate scenary না।

Precision → measure of exactness

recall → " " completeness

$$\text{Precision} = \frac{TP}{TP+FP}$$ → আমার model কতগুলাকে positive বলে।

↓

এর value 1 হওয়া/100%

মানে আমার c class এ → yes class

যে যেগুলা belong করার কথা

সেগুলাই belong ~~করার কথা~~ করতেছে।

recall = 1 / 100% হওয়া মানে FN বলতে কোনো term নাই তার মানে আমার যেগুলা yes সে ও model গুলাকে NO বলতেছে না। but এ extra কিছু data কেও সে বলতে পারে positive. এটা আসলে ঠিক না।

তাই precision বা recall আসলে সবা কেউ ভালো না। তাই একের combined matrix আসছে। এগুলা আছে F1 Score.

F_mea

Harmonic Mean = $\frac{1}{a} + \frac{1}{b}$

Weight factor, precision, recall

$$F = \frac{2 \times P \times R}{P + R}$$

scalability → large data set এর কেস ভালো perform করে না।

3** → Roc curve - 375 page. (~~Positive~~, TP, FP, TPR, FPR)

SORT ~~করতে হবে~~,

ROC curve → 410 page

↓

cost benefit analysis করা।

→ cost associated with false negative is greater than false positive. যেমন: cancer patient 100 জন আছি 5 জন ব্যক্তি 95 হচ্ছে Non-cancer.

Roc curve ⇒ multiple threshold নিয়ে check করে দেখাবে।

যেমন:

	yes	NO
yes	TP	FN
NO	FP	TN

TPR

sensitivity = $\frac{TP}{P}$

specificity = $\frac{TN}{N}$

এদের আর্কিটেক্স

Our কাজের accuracy বের করা যায় mathematically দেখাব।

$1 - \frac{TN}{N} = \frac{FP}{N} = FPR$

Page 412 ***

Prob → 0.90 মানে 0.9 এর বেশি হলে positive

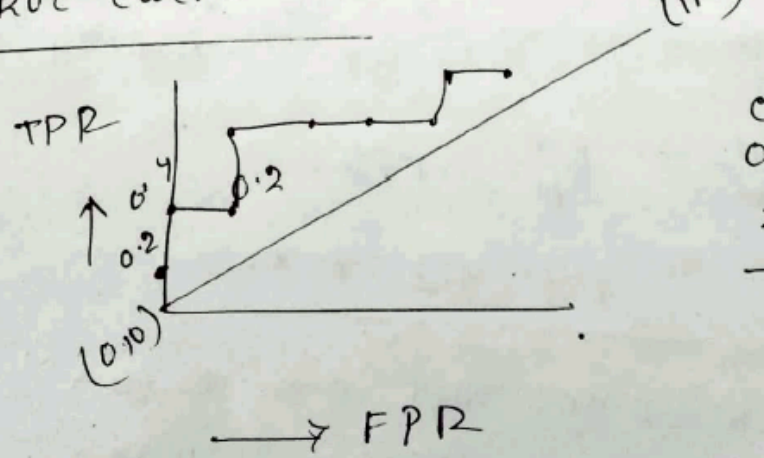
অন্য 0.9 এর কম হলে Negative.

* পরীক্ষায় sort কার নিতে হবে probability ত.

			TP	FP	TPR	
1	P	0.9 (P)	1	0	1/5=0.5	0
2	P	0.8 (P)	2	0	0.4	0
3	N	0.7 (P)	2	1	0.4	0.2
4	P	0.6 (P)	3	1	0.6	0.2
5	P	0.55 (P)	4	1	0.8	0.2
6	N	0.54 (P)	4	2	0.8	0.4
7	N	0.53 (P)	4	3		
8	N	0.51 (P)	4	4		
9	P	0.50 (P)	5	4		
10	N	0.40 (P)	5	5		

wrong

Roc curve



curve সবসময় 0-1 এর মধ্যে হবে কারণ probability

Ex. 12

TP যেখানে 1 TN 3 সেখানে 1.

(8) Ensemble methods.

→ Bagging, Boosting. Ada Boost. - for to class imbalance

→ To Resolve it?

→ Random forest.

→ oversampling (384)

→ under sampling (384 page) 8.12 example

→ Threshold moving.

Ensemble Learning

(i) Bagging

(ii) Boosting

(iii) Random Forest.

Boosting

Adaboost মানে adaptive (

DM

08/12/23

8.6 → 414 Page

classification improve করে ensemble model ফিরে।

↓

majority voting চাই যেখানে calculate করা হয়। 50% হলে average accuracy হয়।

Bagging

↓

মানে voting technique. Parallel process হয়।

Boosting → sequentially চলে আগের model এর weight পরবর্তী model এ use করে।

$t_1 \rightarrow w_1$

$t_2 \rightarrow (w_2) \rightarrow w_3$ এর depend করে।

* যদি M_i model দিয়ে train করে যদি misclassify করে তাহলে weight বাড়াই দিবো M_{i+1} এর জন্য। আবার কাজ করলে weight update না করে আগেরটাই modify করব।

→ slide থেকে ছবিগুলা দেখতে হবে।

Adaboost

→ First এ সব tuple কে weight same দিবো like : $1/d$

→ তারপর যে tuple incorrectly classify করলে তাহলে weight বাড়ায় দিবো। weight কমানো থাকলে ঐ model কে classify করা easy. weight আসলে reflect করে আমার model classify করা কতো hard.

8.6.4 Random Forest → 419 page

↓

অনেকগুলা decision tree এর সমাহার। majority voting এ যে class আসবে নিতে হবে।

positive class → [illegible]
Negative " → " " " " না।

8.6.5

class Imbalance problem → point of class interest class-টা into positive class.

→ Traditional da algo গুলা imbalance data নিয়ে concerned না।

oversampling:- 100 টা টা positive class
1000 " Negative "
↓
মানে অন্তত সংখ্যক point of interest class এর duplicate তৈরি করা। এখানে 100 টাকে ৪ replica করে Negative class এর সমান করা।

আর undersampling হলো Negative class কে থেকে eliminate করে দিবো।

আমার dataset বা আমার জন্য কোনটা ঠিক তার উপর নির্ভর করবে oversampling ভালো না under sampling ভালো।

The **threshold-moving** approach to the class imbalance problem does not involve any sampling. It applies to classifiers that, given an input tuple, return a continuous output value (just like in Section 8.5.6, where we discussed how to construct ROC curves). That is, for an input tuple, $\boldsymbol{X}$, such a classifier returns as output a mapping,

8.7 Summary

- **Classification** is a form of data analysis that extracts models describing data classes. A classifier, or classification model, predicts categorical labels (classes). **Numeric prediction** models continuous-valued functions. Classification and numeric prediction are the two major types of prediction problems.
- **Decision tree induction** is a top-down recursive tree induction algorithm, which uses an attribute selection measure to select the attribute tested for each nonleaf node in the tree. **ID3**, **C4.5**, and **CART** are examples of such algorithms using different attribute selection measures. **Tree pruning** algorithms attempt to improve accuracy by removing tree branches reflecting noise in the data. Early decision tree algorithms typically assume that the data are memory resident. Several scalable algorithms, such as **RainForest**, have been proposed for scalable tree induction.
- **Naïve Bayesian classification** is based on Bayes' theorem of posterior probability. It assumes class-conditional independence—that the effect of an attribute value on a given class is independent of the values of the other attributes.

- A **rule-based classifier** uses a set of IF-THEN rules for classification. Rules can be extracted from a decision tree. Rules may also be generated directly from training data using sequential covering algorithms.
- A **confusion matrix** can be used to evaluate a classifier's quality. For a two-class problem, it shows the *true positives, true negatives, false positives*, and *false negatives*. Measures that assess a classifier's predictive ability include **accuracy**, **sensitivity** (also known as **recall**), **specificity**, **precision**, F, and F_β. Reliance on the accuracy measure can be deceiving when the main class of interest is in the minority.
- Construction and evaluation of a classifier require partitioning labeled data into a training set and a test set. **Holdout, random sampling, cross-validation**, and **bootstrapping** are typical methods used for such partitioning.
- Significance tests and ROC curves are useful tools for model selection. **Significance tests** can be used to assess whether the difference in accuracy between two classifiers is due to chance. **ROC curves** plot the true positive rate (or sensitivity) versus the false positive rate (or $1 - specificity$) of one or more classifiers.
- **Ensemble methods** can be used to increase overall accuracy by learning and combining a series of individual (base) classifier models. **Bagging, boosting**, and **random forests** are popular ensemble methods.
- The **class imbalance problem** occurs when the main class of interest is represented by only a few tuples. Strategies to address this problem include **oversampling, undersampling, threshold moving,** and **ensemble techniques**.

→ Dataset

(3) SVM – Support vector machine (408)

→ SVM কি?

→ SVM Kernel বলতে কি বোঝো?

Theory → linearly Supperable কি হয়?

→ n m-Supperable?

(4) Lazy Learners vs Eager learners পার্থক্য।

- Naive Bayes কি LL নাকি EL?

- KNN কেন LL?

→ KNN নির্ণয়ে কোন কোন math

→

Feature	Eager Learner	Lazy Learner
When model is built	During training phase	During prediction phase
Prediction speed	Faster	Slower
Training speed	Slower	Faster
Memory usage	Lower	Higher
Example algorithms	Linear regression, decision trees	k-nearest neighbors, support vector machines

Lazy vs. Eager Learning

- Lazy vs. eager learning
  - Lazy learning (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - Eager learning (eg. Decision trees, SVM, NN): Given a set of training set, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

Naive Bayes is considered an **eager learner** in machine learning. Here's why:

- **Model building:** During the training phase, Naive Bayes constructs a complete probabilistic model of the data. This model represents the relationships between features and the target variable. Once built, the model is used for prediction without further modifications.
- **Prediction speed:** Once trained, Naive Bayes can make predictions quite quickly on new data points. It simply calculates the posterior probability for each class based on the features of the new data point and chooses the class with the highest probability.
- **Training speed:** While Naive Bayes is relatively fast at predicting, training it can be slower compared to some lazy learners, especially for large datasets. This is because it needs to build the entire probabilistic model upfront.

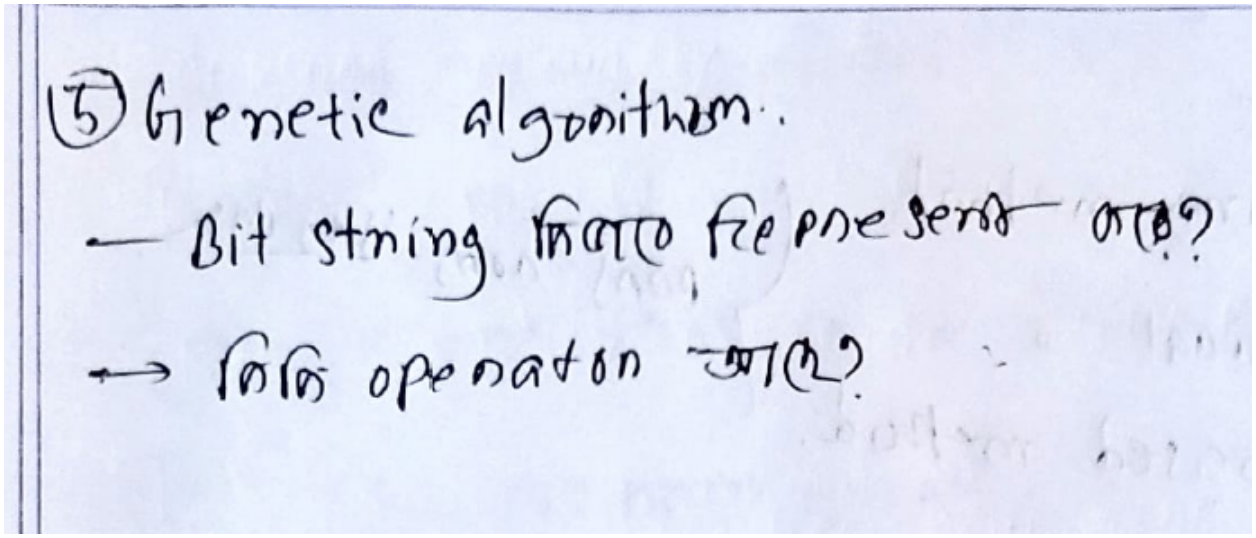
Therefore, based on the characteristics of eager learners, Naive Bayes fits the bill. It builds a complete model during training and uses it readily for predictions without further computation on new data points.

I hope this explanation clarifies the categorization of Naive Bayes! Feel free to ask if you have any further questions.

k-Nearest Neighbors (k-NN) is a lazy learner because it doesn't build a model during training. Instead, it memorizes the entire dataset and makes predictions for new instances based on the closest neighbors at prediction time.

1. **Delayed Model Building:** Unlike eager learners like Naive Bayes that build a complete model during training, kNN doesn't create a global model beforehand. Instead, it stores all the training data points. When given a new data point for prediction, kNN searches for its k nearest neighbors within the stored data. This "on-demand" approach to model building defines kNN as a lazy learner.
2. **Prediction at Test Time:** kNN's actual model building happens at prediction time. It analyzes the k nearest neighbors of the new data point and uses their class labels to predict the class for the new point. This "just-in-time" prediction process further solidifies kNN's lazy learner status.
3. **Memory Trade-off:** While kNN's training is faster than some eager learners due to the lack of upfront model building, it requires storing all the training data. This can lead to higher memory usage, especially for large datasets. This memory-prediction speed trade-off is another characteristic of lazy learners.

So, next time you encounter kNN, remember its delayed model building, just-in-time prediction, and memory-speed trade-off – all telltale signs of a true lazy learner!



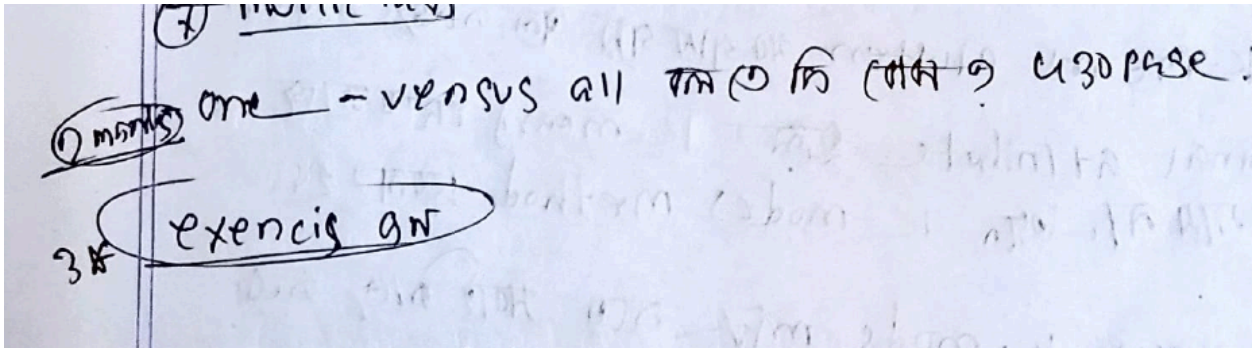
9.6.1 Genetic Algorithms

Genetic algorithms attempt to incorporate ideas of natural evolution. In general, genetic learning starts as follows. An initial **population** is created consisting of randomly generated rules. Each rule can be represented by a string of bits. As a simple example, suppose that samples in a given training set are described by two Boolean attributes, A_1 and A_2, and that there are two classes, C_1 and C_2. The rule "*IF* A_1 *AND NOT* A_2 *THEN* C_2" can be encoded as the bit string "100," where the two leftmost bits represent attributes A_1 and A_2, respectively, and the rightmost bit represents the class. Similarly, the rule "*IF NOT* A_1 *AND NOT* A_2 *THEN* C_1" can be encoded as "001." If an attribute has k values, where $k > 2$, then k bits may be used to encode the attribute's values. Classes can be encoded in a similar fashion.

Based on the notion of survival of the fittest, a new population is formed to consist of the *fittest* rules in the current population, as well as *offspring* of these rules. Typically, the **fitness** of a rule is assessed by its classification accuracy on a set of training samples.

Offspring are created by applying genetic operators such as crossover and mutation. In **crossover**, substrings from pairs of rules are swapped to form new pairs of rules. In **mutation**, randomly selected bits in a rule's string are inverted.

The process of generating new populations based on prior populations of rules continues until a population, P, evolves where each rule in P satisfies a prespecified fitness threshold.



All-versus-all (AVA) is an alternative approach that learns a classifier for each pair of classes. Given m classes, we construct $\frac{m(m-1)}{2}$ binary classifiers. A classifier is trained

using tuples of the two classes it should discriminate. To classify an unknown tuple, each classifier votes. The tuple is assigned the class with the maximum number of votes. All-versus-all tends to be superior to one-versus-all.

A problem with the previous schemes is that binary classifiers are sensitive to errors. If any classifier makes an error, it can affect the vote count.