PANIMALAR INSTITUTE OF TECHNOLOGY

DEPARTMENT OF CSE

R-2017

ACADEMIC YEAR	:2023-2024
BATCH	:2020-2024
YEAR/SEM	:IV/VIII
SUBJECTCODE/TITLE	:CS8080/INFORMATION RETRIEVAL TECHNIQUES

UNIT-3

TEXT CLASSIFICATION AND CLUSTERING

UNIT III
TEXT CLASSIFICATION AND CLUSTERING

A Characterization of Text Classification – Unsupervised Algorithms: Clustering – Naïve Text Classification – Supervised Algorithms – Decision Tree – k-NN Classifier – SVM Classifier – Feature Selection or Dimensionality Reduction – Evaluation metrics – Accuracy and Error – Organizing the classes – Indexing and Searching – Inverted Indexes – Sequential Searching – Multi-dimensional Indexing.

TEXT CLASSIFICATION

First tactic for categorizing documents is to assign a label to each document, but this solve the problem only when the users know the labels of the documents they looking for. This tactic does not solve more generic problem of finding documents on specific topic or subject. For that case, better solution is to group documents by common generic topics and label each group with a meaningful name. Each labeled group is called category or class.

Document classification is the process of categorizing documents under a given cluster or category using fully supervised learning process. Classification could be performed manually by domain experts or automatically using well-known and widely used classification algorithms such as decision tree and Naïve Bayes. Documents are classified according to other attributes (e.g. author, document type, publishing year etc.) or according to their subjects. However, there are two main kind of subject classification of documents: The content based approach and the request based approach. In Content based classification, the weight that is given to subjects in a document decides the class to which the document is assigned. For example, it is a rule in some library classification that at least 15% of the content of a book should be about the class to which the book is assigned. In automatic classification, the number of times given words appears in a document determine the class. In Request oriented classification, the anticipated request from users is impacting how documents are being classified. The classifier asks himself:

"Under which description should this entity be found?" and "think of all the possible queries and decide for which ones the entity at hand is relevant". Automatic document classification tasks can be divided into three types

- Unsupervised document classification (document clustering): the classification must be done totally without reference to external information.
- Semi-supervised document classification: parts of the documents are labeled by the external method.
- Supervised document classification where some external method (such as human feedback) provides information on the correct classification for documents

Unsupervised Learning

Unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:
"*Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision".*

Unsupervised learning cannot be directly applied to a regression or classification problem because unlike supervised learning, we have the input data but no corresponding output data. The goal of unsupervised learning is to **find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.**

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will

perform this task by clustering the image dataset into the groups according to similarities between images.

By Simply,

- ✓ no training data is provided

Examples:

- neural network models
- independent component analysis
- clustering

UNSUPERVISED ALGORITHMS

Below is the list of some popular unsupervised learning algorithms:

- K-means clustering
- KNN (k-nearest neighbors)
- Hierarchal clustering
- Anomaly detection
- Neural Networks
- Principle Component Analysis
- Independent Component Analysis
- Apriori algorithm
- Singular value decomposition

Advantages of Unsupervised Learning

- Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labeled input data.
- Unsupervised learning is preferable as it is easy to get unlabeled data in comparison to labeled data.

Disadvantages of Unsupervised Learning

- Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labeled, and algorithms do not know the exact output in advance.

Difference between Supervised and Unsupervised Learning

Supervised Learning	**Unsupervised Learning**
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in **Classification** and **Regression** p roblems.	Unsupervised Learning can be classified in **Clustering** and **Associations** prob lems.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
It includes various algorithms such	It includes various algorithms such

as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	as Clustering, KNN, and Apriori algorithm.

CLUSTERING

Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as ***"A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."***

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.

It is an unsupervised learning method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.

After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML system can use this id to simplify the processing of large and complex datasets.

The clustering technique is commonly used for **statistical data analysis.**

Clustering Algorithms

The Clustering algorithms can be divided based on their models that are explained above. There are different types of clustering algorithms published, but only a few are commonly used. The clustering algorithm is based on the kind of data that we are using. Such as, some algorithms need to guess the number of clusters in the given dataset, whereas some are required to find the minimum distance between the observation of the dataset. Here we are discussing mainly popular Clustering algorithms that are widely used in machine learning:

- K-Means algorithm
- Mean-shift algorithm
- DBSCAN Algorithm
- Expectation-Maximization Clustering using GMM
- Agglomerative Hierarchical algorithm
- Affinity Propagation

NAIVE TEXT CLASSIFICATION

Naive Bayes classifiers are a collection of classification algorithms based on Bayes Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Naive Bayes classifiers have been heavily used for text classification and text analysis machine learning problems.

Text Analysis is a major application field for machine learning algorithms. However the raw data, a sequence of symbols (i.e. strings) cannot be fed directly to the algorithms themselves as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length.

The Naive Bayes algorithm

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

The dataset is divided into two parts, namely, **feature matrix** and the **response/target vector**.

- The **Feature matrix** (X) contains all the vectors(rows) of the dataset in which each vector consists of the value of **dependent features**. The number of features is **d** i.e. **X = (x1,x2,x2, xd).**
- The **Response/target vector** (y) contains the value of **class/group variable** for **each row of feature matrix.**

The Bayes' Theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as follows:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

where:

- **A** and **B** are called **events.**
- P(A | B) is the probability of event A, given the event B is true (has occured). Event B is also termed as **evidence**.
- P(A) is the **priori** of A (the prior independent probability, i.e. probability of event before evidence is seen).
- P(B | A) is the probability of B given event A, i.e. probability of event B after evidence A is seen.

Summary

A, B = events
$P(A \mid B)$ = probability of A given B is true
$P(B \mid A)$ = probability of B given A is true
$P(A), P(B)$ = the independent probabilities of A and B

The Naive Bayes Model

Given a data matrix **X** and a target vector **y,** we state our problem as:

$$P(y \mid X) = \frac{P(X \mid y)P(y)}{P(X)}$$

where, **y** is **class variable** and **X** is a **dependent feature vector with dimension d i.e. X = (x1,x2,x2, xd),** where **d** is the number of variables/features of the sample. P(y|X) is the probability of observing the class **y** given the sample **X** with **X = (x1,x2,x2, xd),** where **d** is the number of variables/features of the sample. Now the "naïve" conditional independence assumptions come into play: assume that all features in **X** are mutually independent, conditional on the category **y**:

$$P(y \mid x_1, \ldots, x_d) = \frac{P(y)\prod_{i=1}^{d} P(x_i|y)}{P(x_1)P(x_2)\ldots P(x_d)}$$

The denominator remains constant for a given input, so we can remove that term:

$$P(y \mid x_1, \ldots, x_d) \propto P(y)\prod_{i-1}^{d} P(x_i \mid y)$$

Finally, to find the probability of a given **sample** for all possible values of the class variable ***y***, we just need to find the output with maximum probability:

$$y = \operatorname{argmax}_y P(y)\prod_{i=1}^{n} P(x_i \mid y)$$

Dealing with text data

Text Analysis is a major application field for machine learning algorithms. However the raw data, a sequence of symbols (i.e. strings) cannot be fed directly to the algorithms themselves as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length.

In order to address this, scikit-learn provides utilities for the most common ways to extract numerical features from text content, namely:

- **tokenizing** strings and giving an integer id for each possible token, for instance by using w ite-spaces and punctuation as token separators.
- **counting** the occurrences of tokens in each document.

In this scheme, features and samples are defined as follows:

- each **individual token occurrence frequency** is treated as a **feature**.
- the vector of all the token frequencies for a given **document** is considered a multivariate **sample**.

Example 1 : Using the Naive Bayesian Classiier

We will consider the following training set. The data samples are described by attributes age, income, student, and credit. The class label attribute, buy, tells whether the person buys a computer, has two distinct values, yes (class C_1) and no (class C_2).

RID	Age	Income	student	credit	C_i: buy
1	Youth	High	no	fair	C_2: no
2	Youth	High	no	excellent	C_2: no
3	middle-aged	High	no	fair	C1: yes
4	Senior	medium	no	fair	C1: yes
5	Senior	Low	yes	fair	C1: yes
6	Senior	Low	yes	excellent	C2: no
7	middle-aged	Low	yes	excellent	C1: yes
8	Youth	medium	no	fair	C2: no
9	Youth	Low	yes	fair	C1: yes
10	Senior	medium	yes	fair	C1: yes
11	Youth	medium	yes	excellent	C1: yes
12	middle-aged	medium	no	excellent	C1: yes
13	middle-aged	High	yes	fair	C1: yes
14	Senior	medium	no	excellent	C2: no

The sample we wish to classify is

X = (age = youth, income = medium, student = yes, credit = fair)

We need to maximize $P(X|C_i)P(C_i)$, for i = 1, 2. $P(C_i)$, the a priori probability of each

class, can be estimated based on the training samples:

P(buy =yes) = 9 /14
P(buy =no) = 5 /14

To compute P (X|Ci), for i = 1, 2, we compute the following conditional probabilities:

P(age=youth | buy =yes) = 2/9
P(income =medium | buy =yes) = 4/9
P(student =yes | buy =yes) = 6/9
P(credit =fair | buy =yes) = 6/9

P(age=youth | buy =no) = 3/5
P(income =medium | buy =no) = 2/5
P(student =yes | buy =no) = 1/5
P(credit =fair | buy =no) = 2/5

Using the above probabilities, we obtain

$$P(\mathbf{X}|buy = yes) = P(age = youth|buy = yes) \\ P(income = medium|buy = yes) \\ P(student = yes|buy = yes) \\ P(credit = fair|buy = yes) \\ = \frac{2}{9}\frac{4}{9}\frac{6}{9}\frac{6}{9} = 0.044.$$

Similarly

$$P(\mathbf{X}|buy = no) = \frac{3}{5}\frac{2}{5}\frac{1}{5}\frac{2}{5} = 0.019$$

To ind the class that maximizes P (X|Ci)P (Ci), we compute

$$P(\mathbf{X}|buy = yes)P(buy = yes) = 0.028$$
$$P(\mathbf{X}|buy = no)P(buy = no) = 0.007$$

Thus the naive Bayesian classiier predicts buy = yes for sample X.

Example 2:

Predicting a class label using naïve Bayesian classiication. The training data set is given below. The data tuples are described by the attributes Owns Home?, Married, Gender and Employed. The class label attribute Risk Class has three distinct values. Let C1 corresponds to the class A, and C2 corresponds to the class B and C3 corresponds to the class C.

The tuple is to classify is,

X = (Owns Home = Yes, Married = No, Gender = Female, Employed = Yes)

Owns Home	Married	Gender	Employed	Risk Class
Yes	Yes	Male	Yes	B
No	No	Female	Yes	A
Yes	Yes	Female	Yes	C
Yes	No	Male	No	B
No	Yes	Female	Yes	C
No	No	Female	Yes	A

No	No	Male	No	B
Yes	No	Female	Yes	A
No	Yes	Female	Yes	C
Yes	Yes	Female	Yes	C

Solution

There are 10 samples and three classes.

Risk class A = 3

Risk class B = 3

Risk class C = 4

The prior probabilities are obtained by dividing these frequencies by the total number in the training data,

P(A) = 3/10 = 0.3
P(B) = 3/10 = 0.3
P(C) = 4/10 = 0.4

To compute P(X/Ci) =P {yes, no, female, yes}/Ci) for each of the classes, the conditional probabilities for each:

P(Owns Home = Yes/A) = 1/3 =0.33
P(Married = No/A) = 3/3 =1
P(Gender = Female/A) = 3/3 = 1
P(Employed = Yes/A) = 3/3 = 1

P(Owns Home = Yes/B) = 2/3 =0.67
P(Married = No/B) = 2/3 =0.67
P(Gender = Female/B) = 0/3 = 0
P(Employed = Yes/B) = 1/3 = 0.33

P(Owns Home = Yes/C) = 2/4 =0.5
P(Married = No/C) = 0/4 =0
P(Gender = Female/C) = 4/4 = 1
P(Employed = Yes/C) = 4/4 = 1

Using the above probabilities, we obtain
P(X/A)= P(Owns Home = Yes/A) X P(Married = No/A) x P(Gender = Female/A) X P(Employed = Yes/A)

= 0.33 x 1 x 1 x 1 = 0.33

Similarly, P(X/B)= 0 , P(X/C) =0

To ind the class, G, that maximizes,
P(X/Ci)P(Ci), we compute,

P(X/A) P(A) = 0.33 X 0.3 = 0.099
P(X/B) P(B) =0 X 0.3 = 0
P(X/C) P(C) = 0 X 0.4 = 0.0

Therefore x is assigned to class A

Advantages:

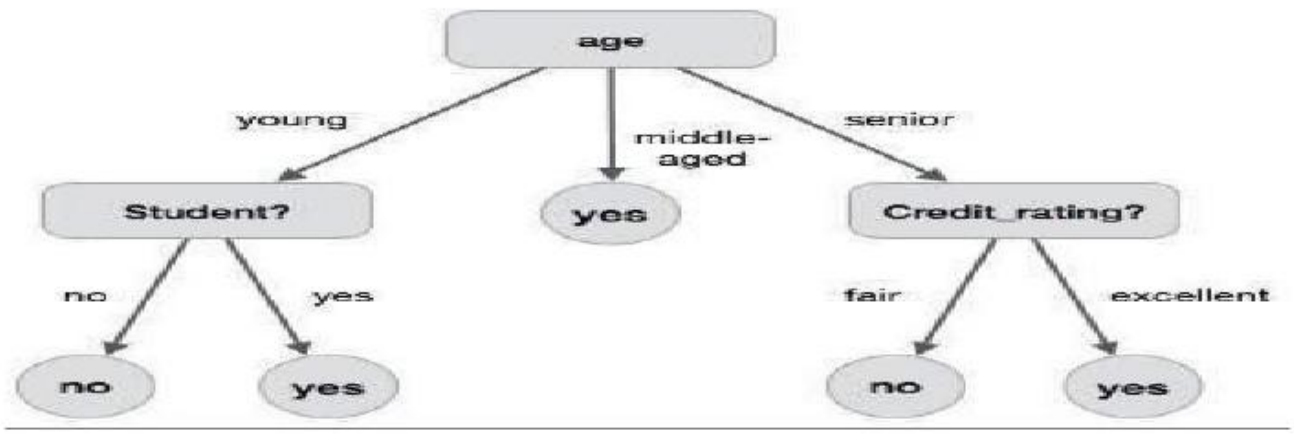
- Have the minimum error rate in comparison to all other classifiers.
- Easy to implement
- Good results obtained in most of the cases.
- They provide theoretical justiication for other classiiers that do not explicitly use

- Lack of available probability data.
- Inaccuracies in the assumption.

5.5.2 Decision Trees

A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

The following decision tree is for the concept buy_computer that indicates whether a customer at a company is likely to buy a computer or not. Each internal node represents a test on an attribute. Each leaf node represents a class.



The beneits of having a decision tree are as follows –

- It does not require any domain knowledge.
- It is easy to comprehend.
- The learning and classiication steps of a decision tree are simple and fast.

Decision Tree Induction Algorithm

- A machine researcher named J. Ross Quinlan in 1980 developed a decision tree algorithm known as ID3 (Iterative Dichotomiser). Later, he presented C4.5, which was the successor of ID3. ID3 and C4.5 adopt a greedy approach. In this algorithm, there is no backtracking; the trees are constructed in a top-down recursive divide-and-conquer manner.

Generating a decision tree form training tuples of data partition D

Algorithm : Generate_decision_tree

Input:

- Data partition, D, which is a set of training tuples and their associated class labels.
- attribute_list, the set of candidate attributes.
- Attribute selection method, a procedure to determine the splitting criterion that best partitions that the data tuples into individual classes. This criterion includes a splitting_attribute and either a splitting point or splitting subset.

Output: A Decision Tree

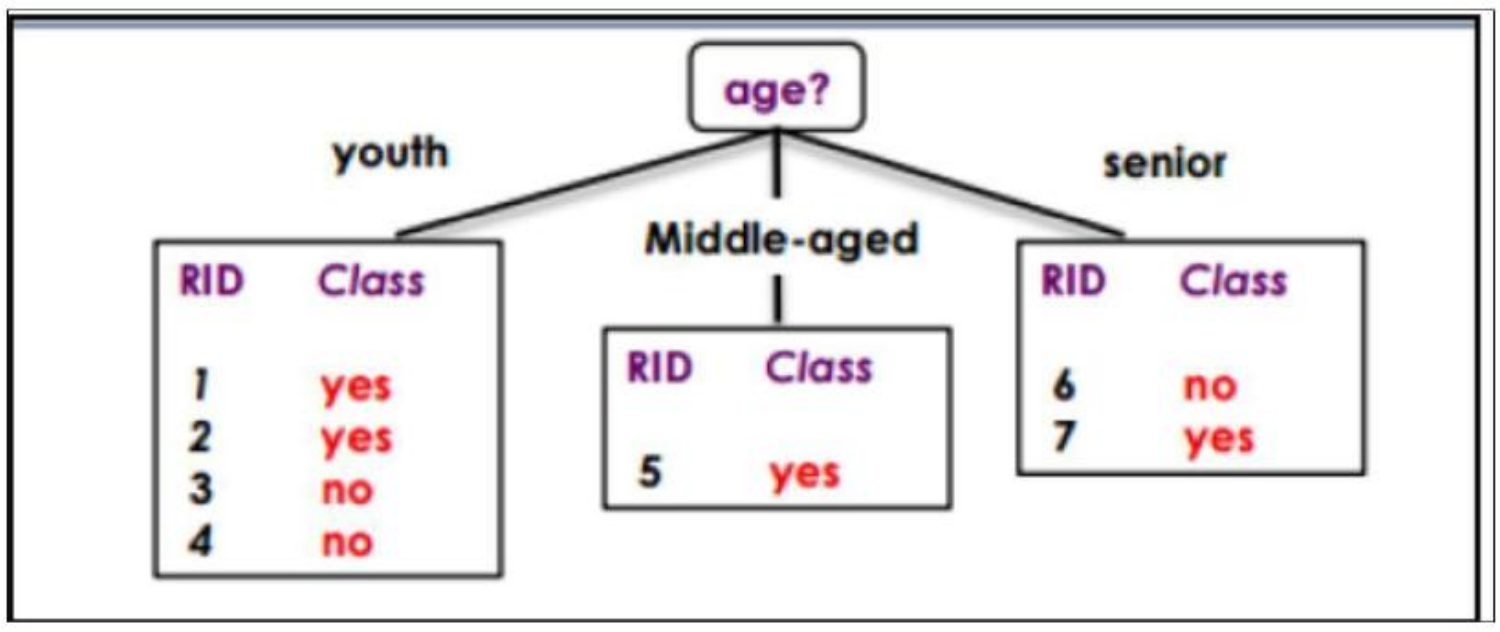
Method

1. create a node N;
2. if tuples in D are all of the same class, C then
3. return N as leaf node labeled with class C;
4. if attribute_list is empty then
5. return N as leaf node with labeled with majority class in D;//majority voting
6. apply attribute_selection_method(D, attribute_list) to ind the best splitting_criterion;
7. label node N with splitting_criterion;
8. if splitting_attribute is discrete-valued and multiway splits allowed then // not restricted to binary trees
9. attribute_list = attribute_list - splitting attribute; // remove splitting attribute
10. for each outcome j of splitting criterion
    // partition the tuples and grow subtrees for each partition
11. let Dj be the set of data tuples in D satisfying outcome j; // a partition
12. if Dj is empty then
13. attach a leaf labeled with the majority class in D to node N;
14. else attach the node returned by Generate_decision tree(Dj, attribute list) to node N;
    end for
15. return N;

Example : customer likely to purchase a computer

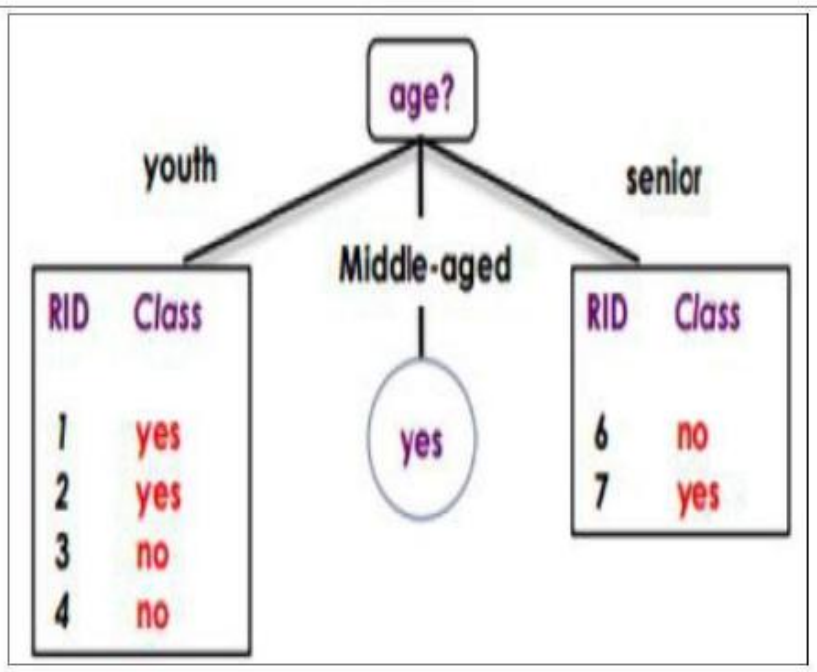
From the training dataset , calculate entropy value, which indicates that splitting attribute is: age

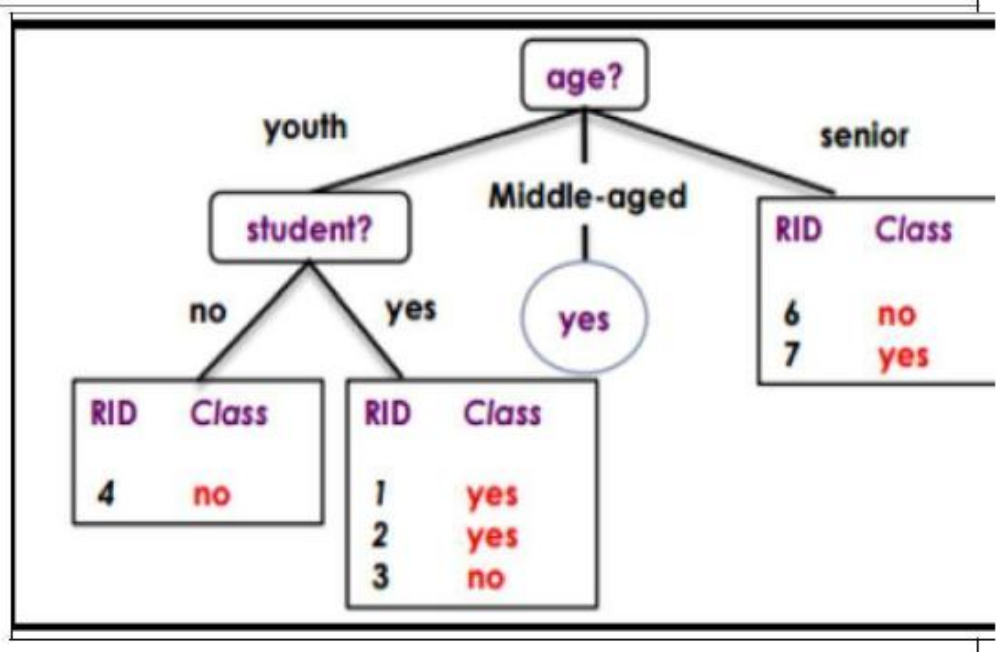
RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes



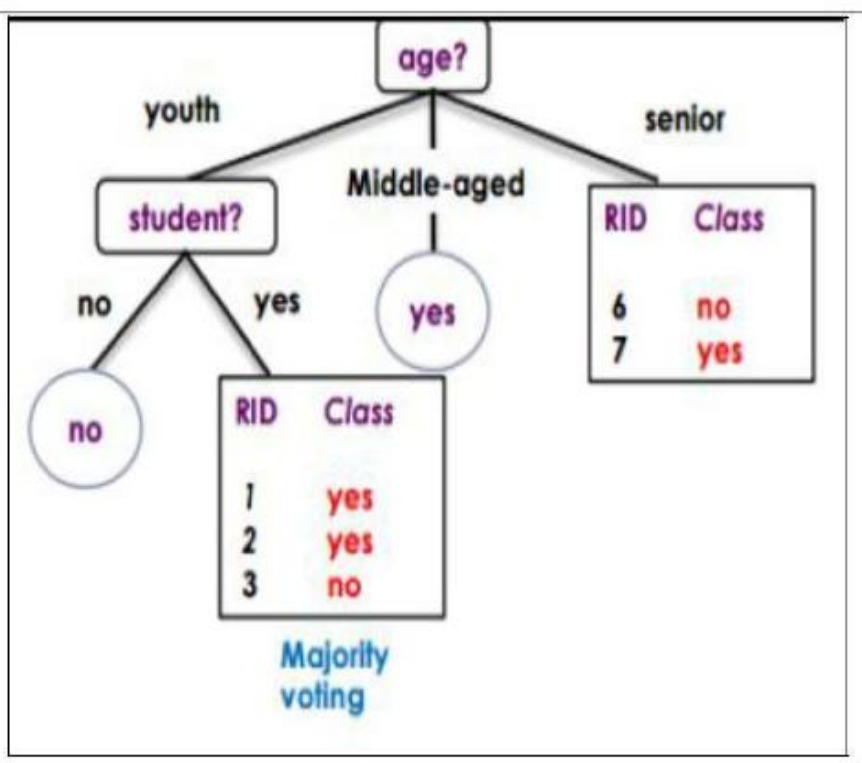
Age has 3 partitions :

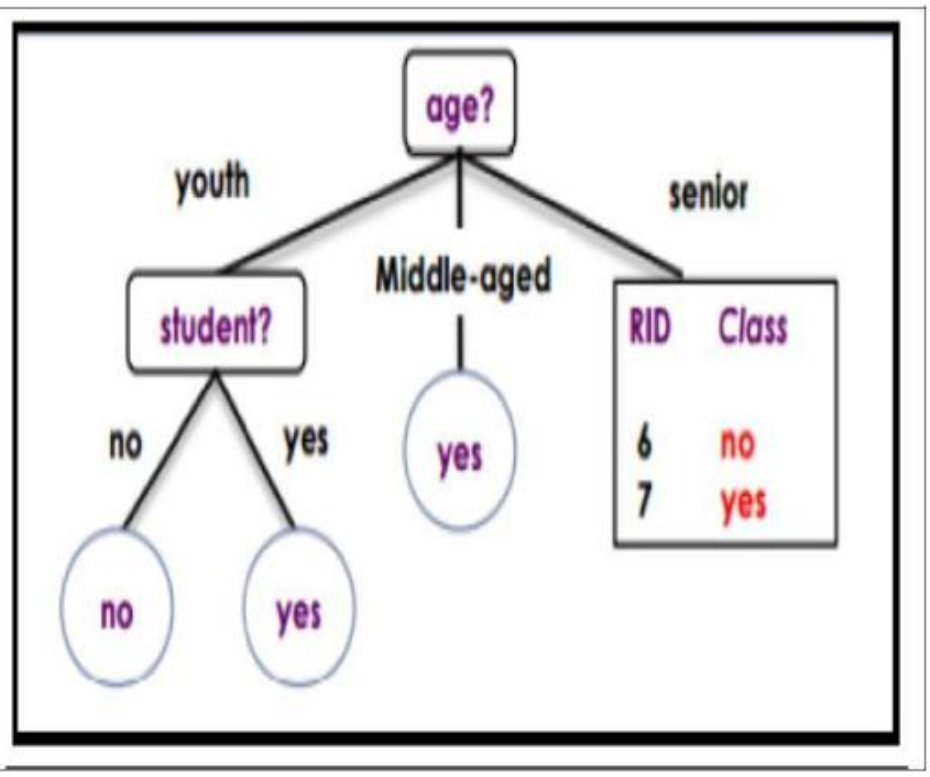
From the training data set , age= youth has 2 classes based on student attribute



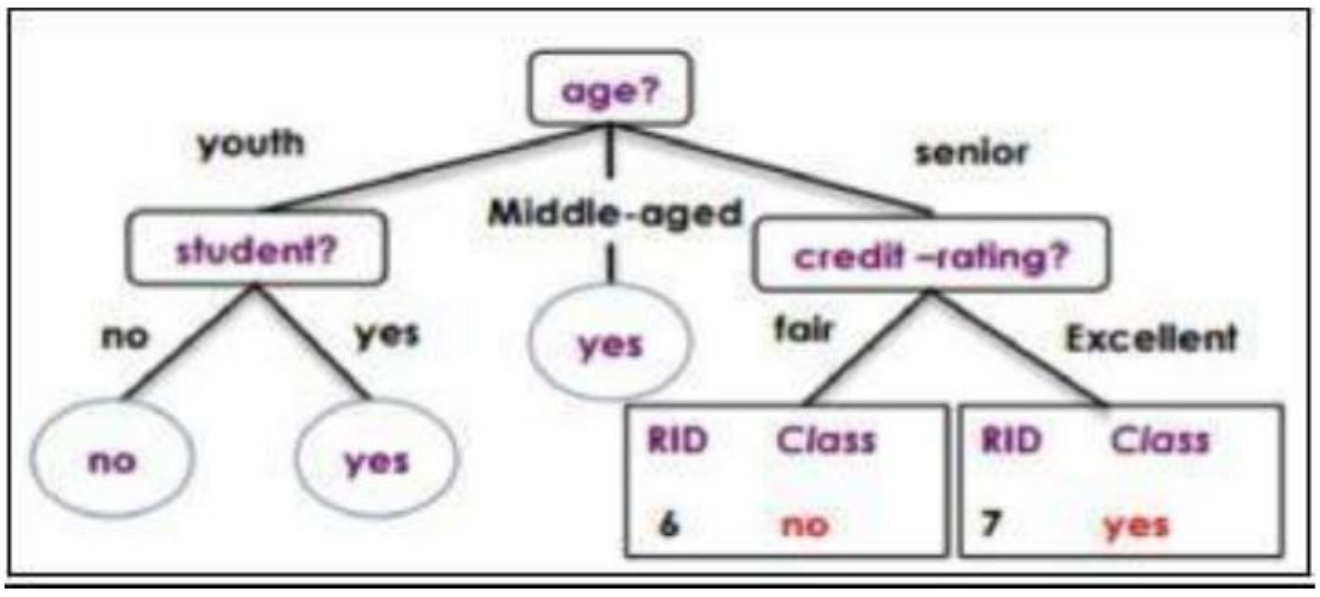


based on majority voting in student attribute , RID=3 is grouped under yes group.

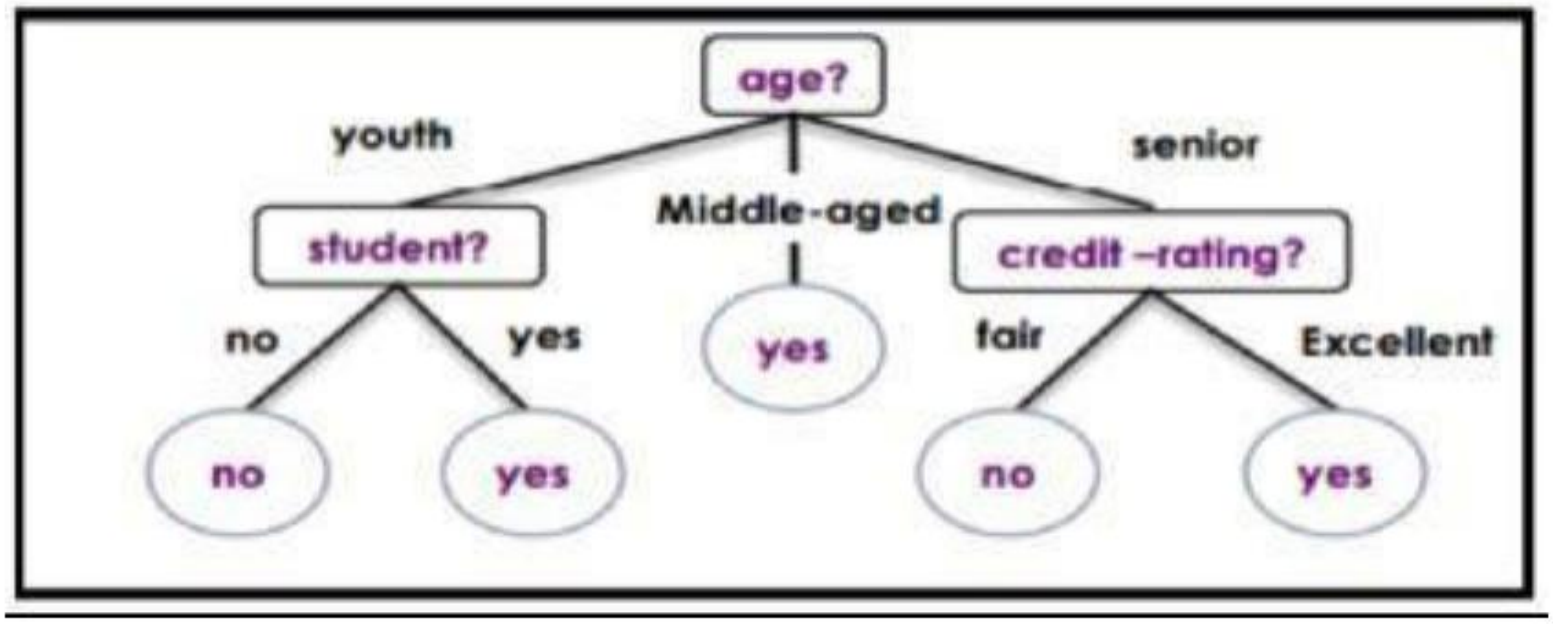




From the training data set , age= senior has 2 classes based on credit rating.

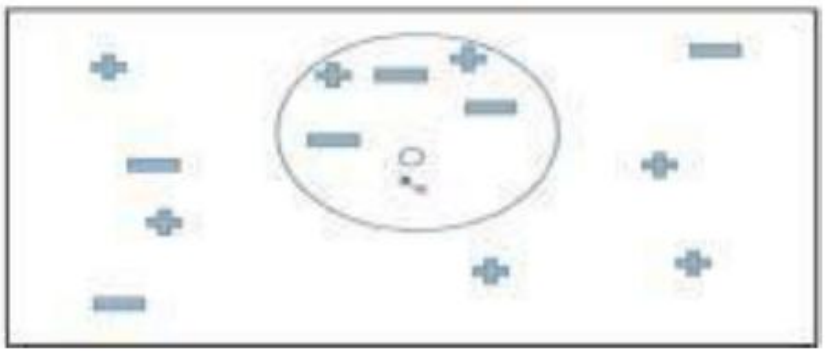


Final Decision Tree



5.5.3 K Nearest Neighbor Classiication

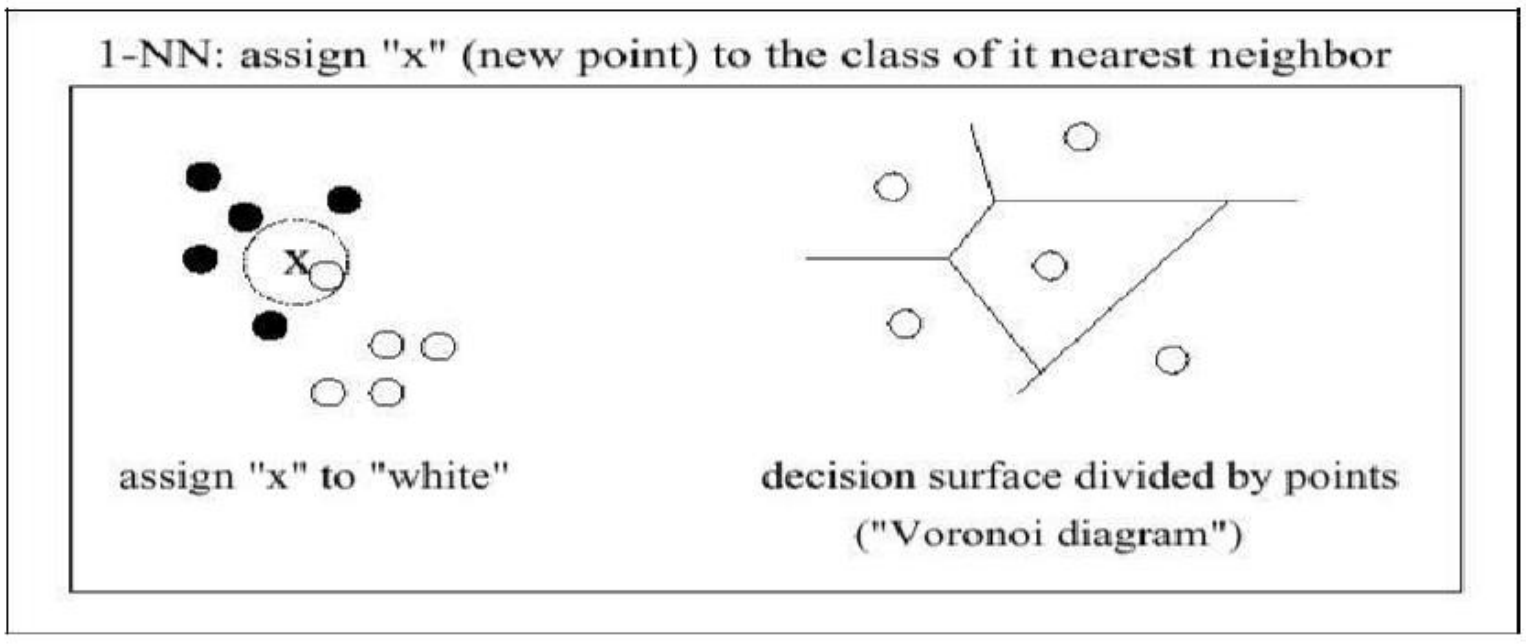
- K nearest neighbors is a simple algorithm that stores all available cases and classiies new cases based on a similarity measure (e.g., distance functions). K represents number of nearest neighbors.
- It classify an unknown example with the most common class among k closest examples
- KNN is based on "tell me who your neighbors are, and I'll tell you who you are"
- Example:

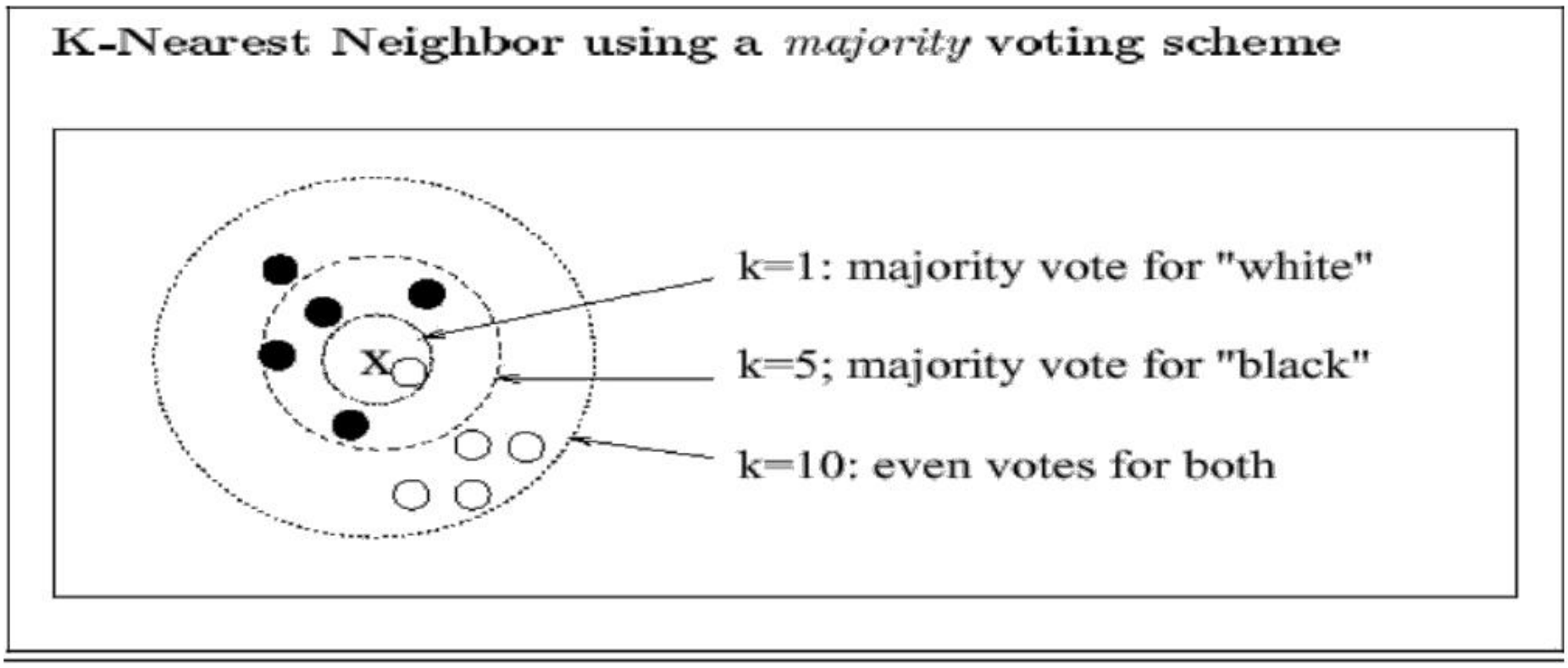


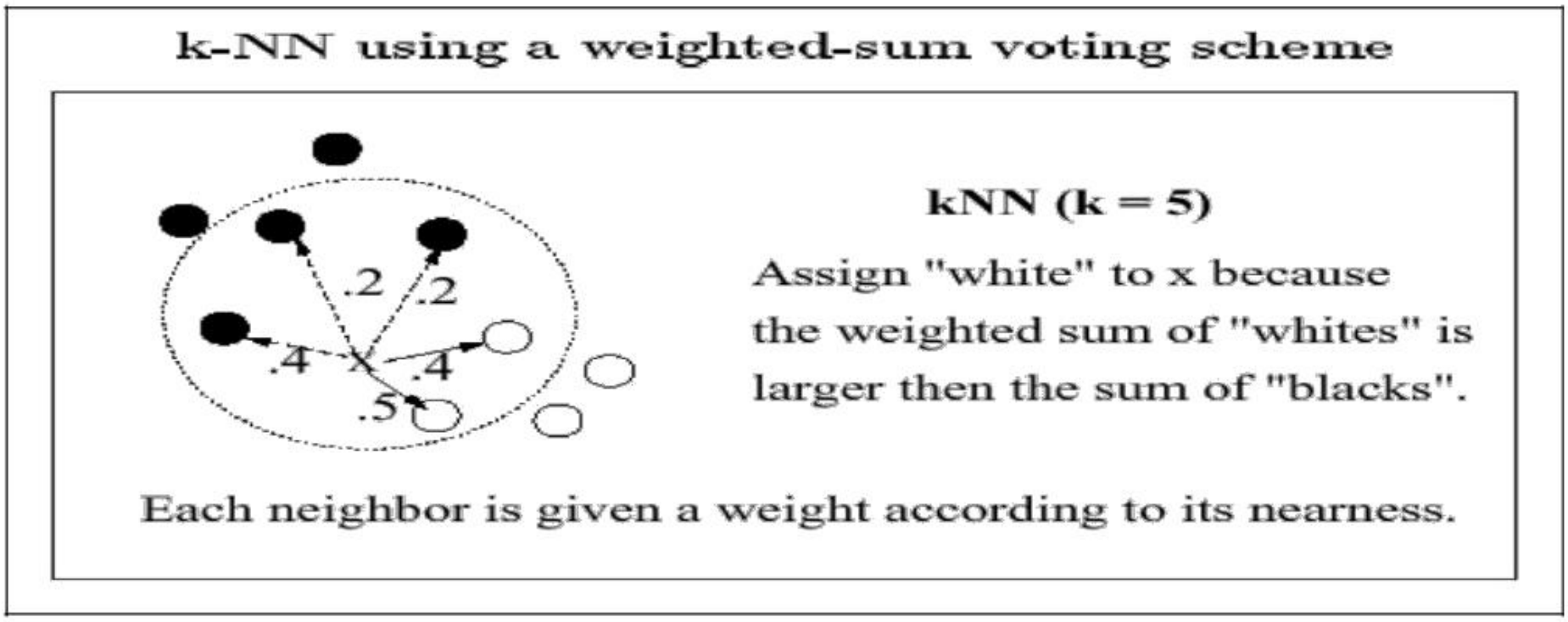
- If K = 5, then in this case query instance x_q will be classiied as negative since three of its nearest neighbors are classiied as negative.

Different Schemes of KNN

a. 1-Nearest Neighbor
b. K-Nearest Neighbor using a majority voting scheme
c. K-NN using a weighted-sum voting Scheme







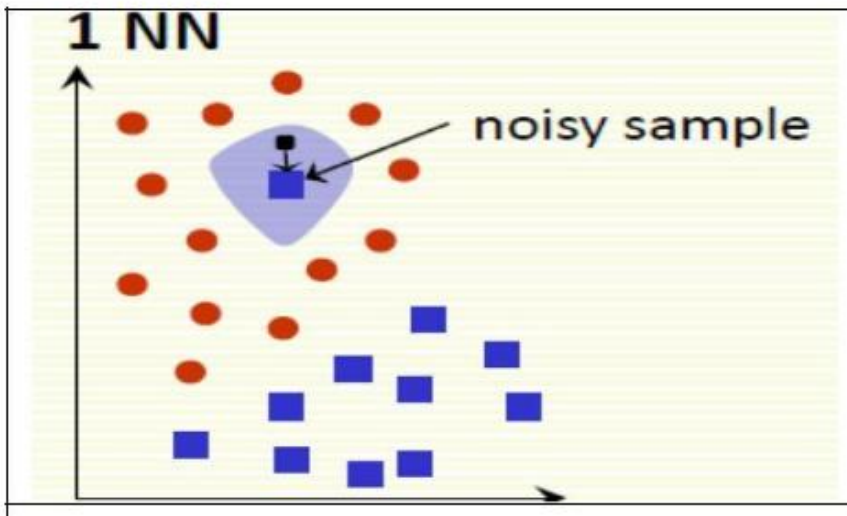
kNN: How to Choose k?

In theory, if ininite number of samples available, the larger is ***k***, the better is classification

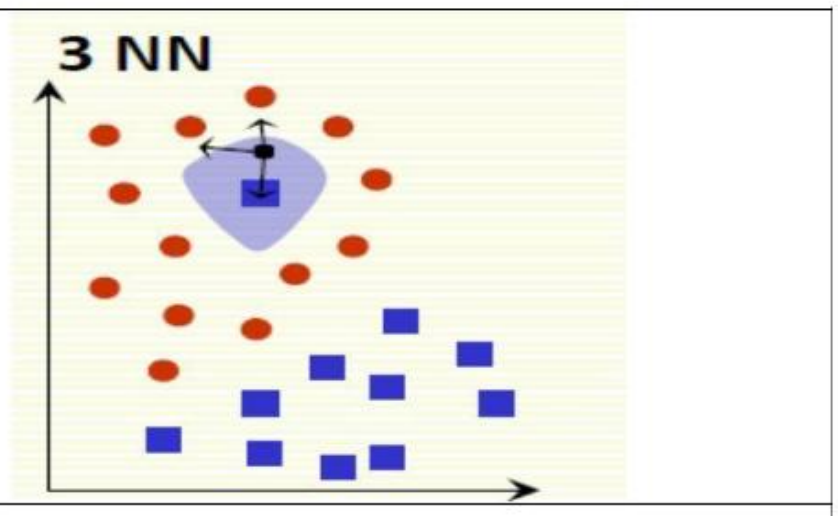
The limitation is that all ***k*** neighbors have to be close

- Possible when ininite no of samples available
- Impossible in practice since no of samples is inite

k = 1 is often used for eficiency, but sensitive to “noise”



every example in the blue shaded area will be misclassiied as the blue class(rectangle)



every example in the blue shaded area will be classiied correctly as the red class(circle)

Larger k gives smoother boundaries, better for generalization But only if locality is preserved. Locality is not preserved if end up looking at samples too far away, not from the same class.

•Interesting theoretical properties if k < sqrt(n), n is # of examples .

Find a heuristically optimal number k of nearest neighbors, based on RMSE(root-mean-square error). This is done using cross validation.

Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.

Distance Measure in KNN

There are three distance measures are valid for continuous variables.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

It should also be noted that all In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

Hamming Distance

$$D_H = \sum_{i-1}^{k}|x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

Simple KNN - Algorithm:

For each training example , add the example to the list of training_examples.

Given a query instance x_q to be classiied,

- Let x1 ,x2….xk denote the k instances from training_examples that are nearest to x_q .
- Return the class that represents the maximum of the k instances

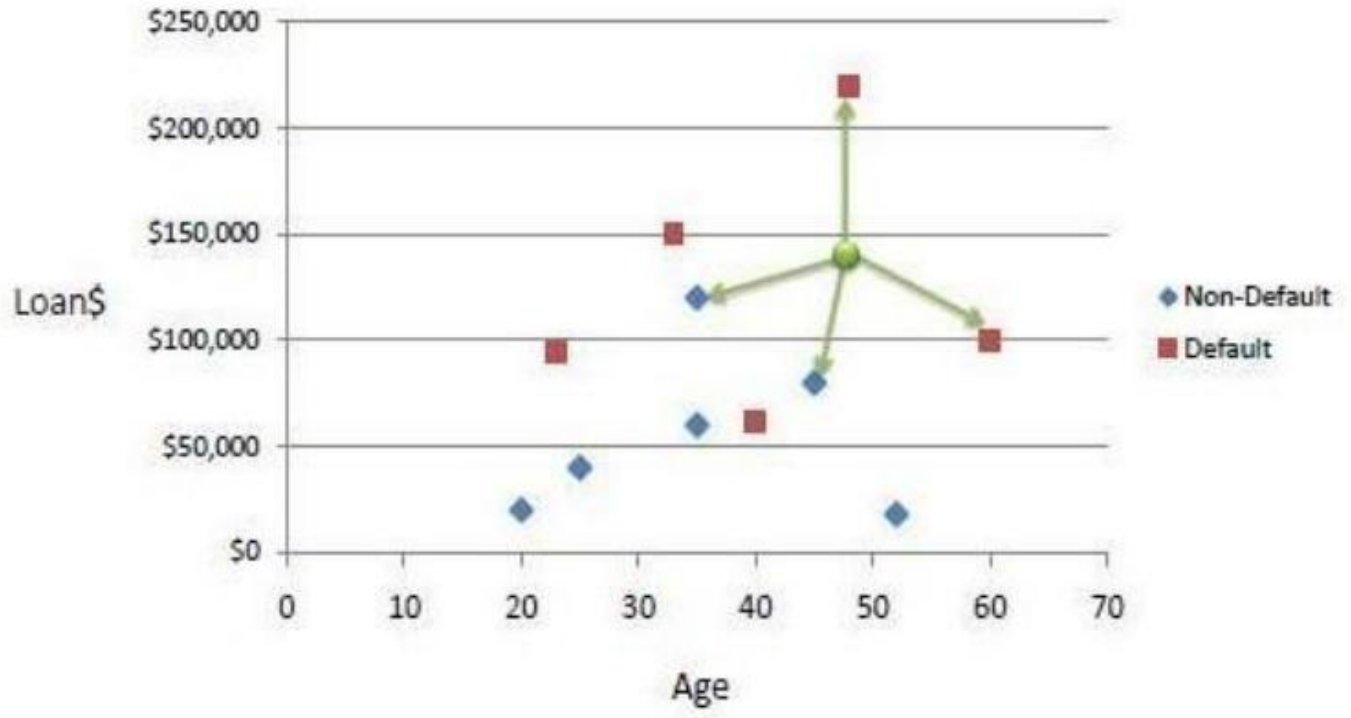
Steps:

1. Determine parameter k= no of nearest neighbor
2. Calculate the distance between the query instance and all the training samples

3. Sort the distance and determine nearest neighbor based on the k –th minimum distance
4. Gather the category of the nearest neighbors
5. Use simple majority of the category of nearest neighbors as the prediction value of the query instance.

Example:

Consider the following data concerning credit default. Age and Loan are two numerical variables (predictors) and Default is the target.



Given Training Data set :

Age	Loan	Default
25	$40,000	N
35	$60,000	N
45	$80,000	N
20	$20,000	N
35	$120,000	N
52	$18,000	N
23	$95,000	Y
40	$62,000	Y
60	$100,000	Y
48	$220,000	Y
33	$150,000	Y

Data to Classify:

to classify an unknown case (Age=48 and Loan=$142,000) using Euclidean distance.

Step1: Determine parameter k

K=3

Step 2: Calculate the distance

D = Sqrt[(48-33)^2 + (142000-150000)^2] = 8000.01 >> Default=Y

Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	$142,000	?		

Euclidean Distance

$$D=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$$

Step 3: Sort the distance (refer above diagram) and mark upto k_{th} rank i.e 1 to 3.

Step 4: Gather the category of the nearest neighbors

Age	Loan	Default	Distance
33	$150000	Y	8000
35	$120000	N	22000
60	$100000	Y	42000

With K=3, there are two Default=Y and one Default=N out of three closest neighbors. The prediction for the unknown case is Default=Y.

Standardized Distance (Feature Normalization)

One major drawback in calculating distance measures directly from the training set is in the case where variables have different measurement scales or there is a mixture of numerical and categorical variables. For example, if one variable is based on annual income in dollars, and the other is based on age in years then income will have a much higher inluence on the distance calculated. One solution is to standardize the training set as shown below.

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	0.61	?	

Standardized Variable

$$X_s=\frac{X-Min}{Max-Min}$$

For ex loan , X =$ 40000 ,

$$Xs = \frac{40000-20000}{220000-20000} = 0.11$$

Same way , calculate the standardized values for age and loan attributes, then apply the KNN algorithm.

Advantages

- Can be applied to the data from any distribution
    - for example, data does not have to be separable with a linear boundary
- Very simple and intuitive
- Good classification if the number of samples is large enough

Disadvantages

- Choosing k may be tricky
- Test stage is computationally expensive
    - No training stage, all the work is done during the test stage
    - This is actually the opposite of what we want. Usually we can afford training step to take a long time, but we want fast test step
- Need large number of samples for accuracy

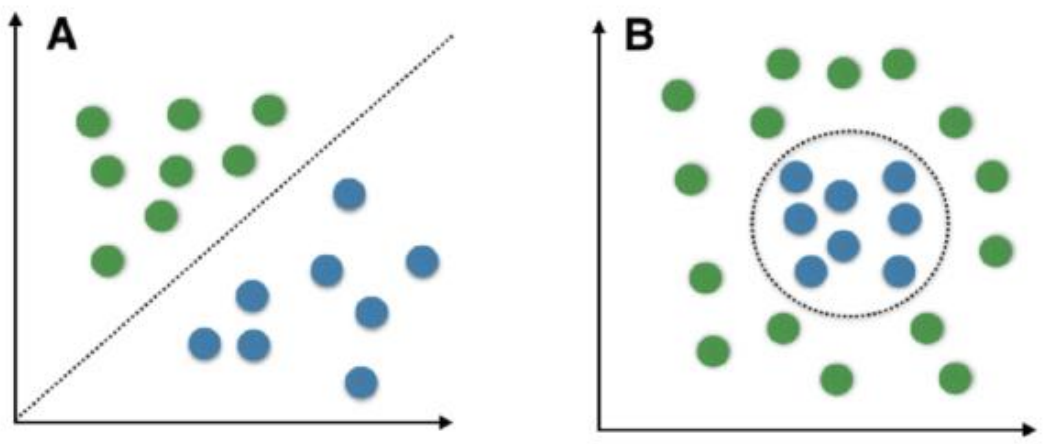
SUPPORT VECTOR MACHINE

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line. In SVM, we plot each data item in the dataset in an N-dimensional space, where N is the number of features/attributes in the data. Next, find the optimal hyperplane to separate the data. So by this, you must have understood that inherently, SVM can only perform binary classification (i.e., choose between two classes). However, there are various techniques to use for multi-class problems. **Support Vector Machine for Multi-class Problems** To perform SVM on multi-class problems, we can create a binary classifier for each class of the data. The two results of each classifier will be :

- The data point belongs to that class OR
- The data point does not belong to that class.

For example, in a class of fruits, to perform multi-class classification, we can create a binary classifier for each fruit. For say, the 'mango' class, there will be a binary classifier to predict if it IS a mango OR it is NOT a mango. The classifier with the highest score is chosen as the output of the SVM.

SVM for complex (Non Linearly Separable) SVM works very well without any modifications for linearly separable data. **Linearly Separable Data** is any data that can be plotted in a graph and can be separated into classes using a straight line.

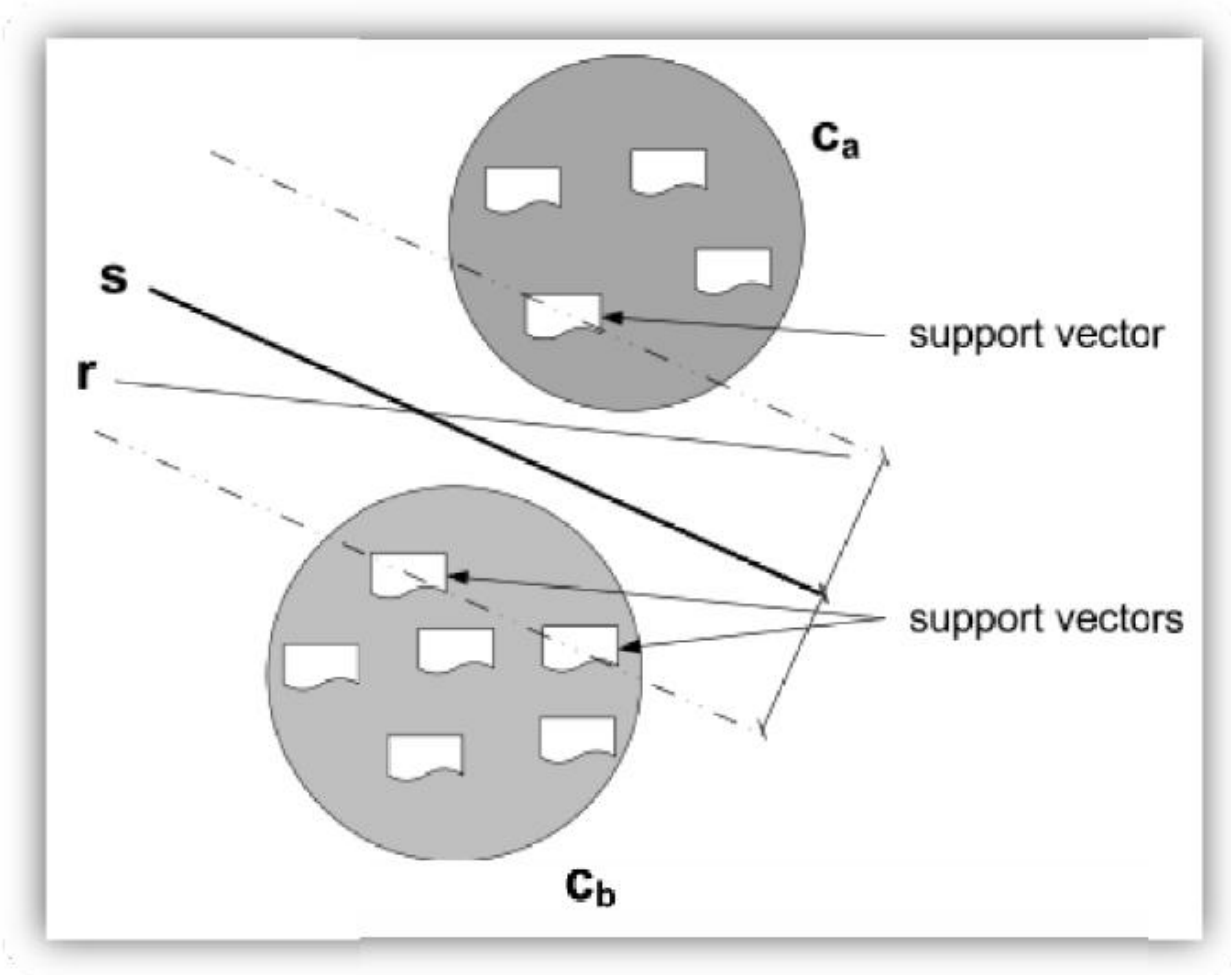


A: Linearly Separable Data B: Non-Linearly Separable Data

SVM CLASSIFIER

- ✓ a vector space method for binary classification problems
- ✓ documents represented in t-dimensional space
- ✓ find a **decision surface (hyperplane)** that best separate
- ✓ documents of two classes
- ✓ new document classified by its position relative to hyperplane.

Simple 2D example: training documents linearly separable

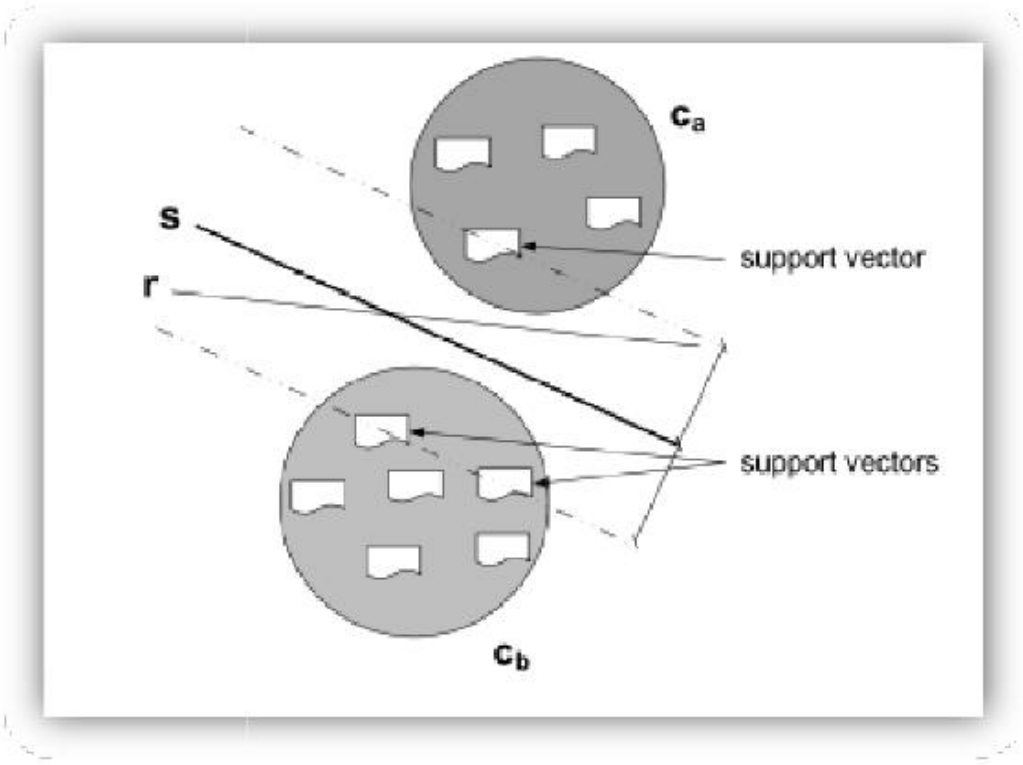


Line s**—The Decision Hyperplane**

- ✓ maximizes distances to closest docs of each class
- ✓ it is the best separating hyperplane

Delimiting Hyperplanes

- ✓ parallel dashed lines that delimit region where to look for a solution

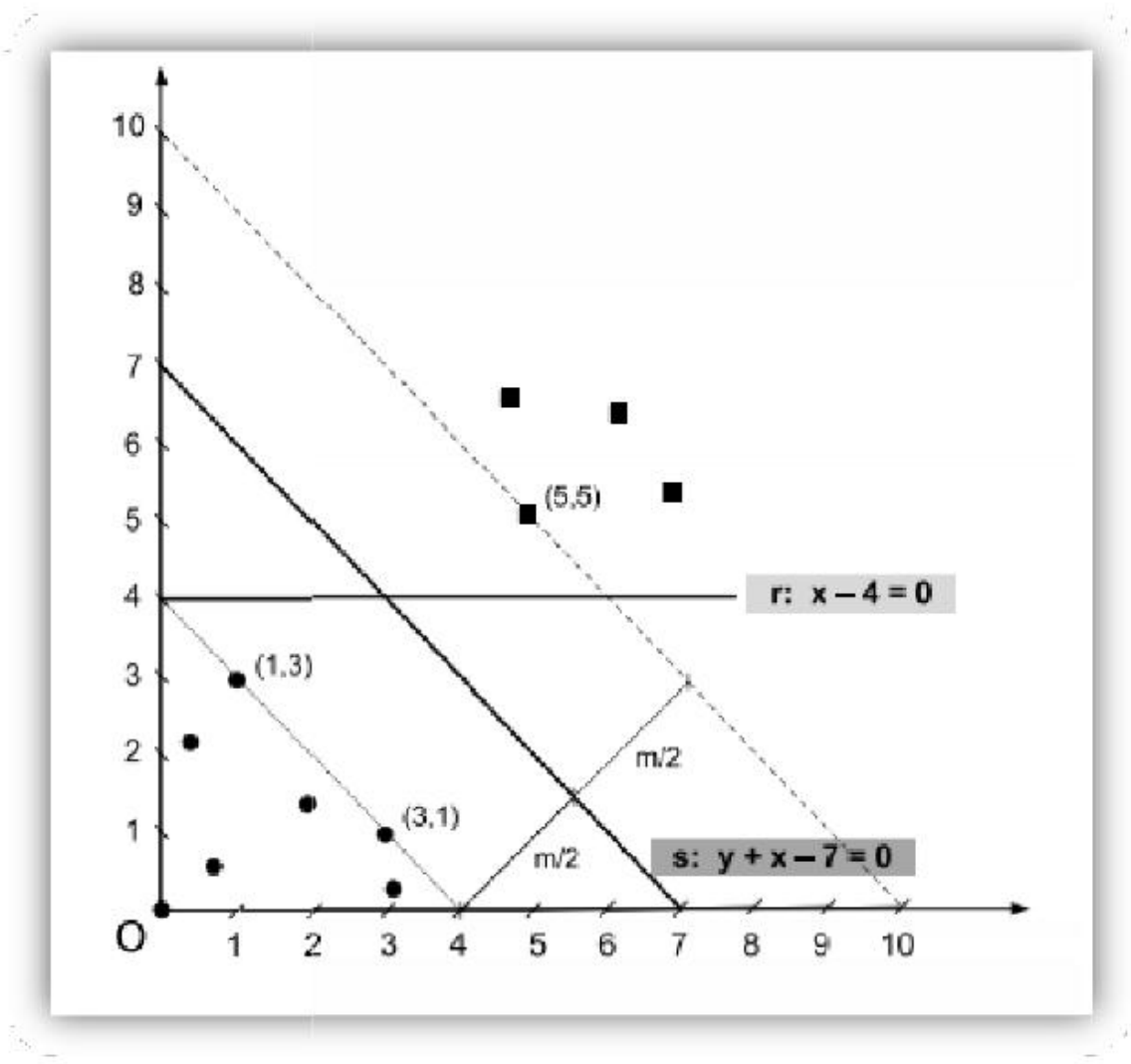


✓ Lines that cross the delimiting hyperplanes.

- candidates to be selected as the decision hyperplane
- lines that are parallel to delimiting hyperplanes: best candidates

Support vectors: documents that belong to, and define, the delimiting hyperplanes

Our example in a 2-dimensional system of coordinates



Let,

- $\mathcal{H}_w$: a hyperplane that separates docs in classes c_a and c_b
- m_a: distance of $\mathcal{H}_w$ to the closest document in class c_a
- m_b: distance of $\mathcal{H}_w$ to the closest document in class c_b
- $m_a + m_b$: **margin** m of the SVM

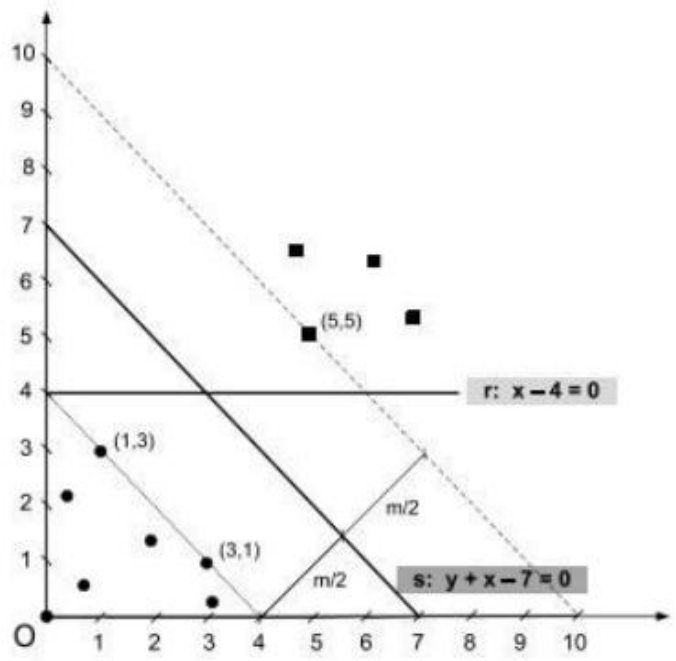
The **decision hyperplane** maximizes the margin m

Hyperplane $r : x - 4 = 0$ separates docs in two sets

- its distances to closest docs in either class is 1
- thus, its margin m is 2

Hyperplane $s : y + x - 7 = 0$ has margin equal to $3\sqrt{2}$

- maximum for this case
- s is the decision hyperplane



FEATURE SELECTION OR DIMENSIONALITY REDUCTION

Feature selection and dimensionality reduction allow us to minimize the number of features in a dataset by only keeping features that are important. In other words, we want to retain features that contain the most useful information that is needed by our model to make accurate predictions while discarding redundant features that contain little to no information. There are several benefits in performing feature selection and dimensionality reduction which include model

interpretability, minimizing overfitting as well as reducing the size of the training set and consequently training time.

Dimensionality Reduction

The number of input variables or features for a dataset is referred to as its dimensionality. Dimensionality reduction refers to techniques that reduce the number of input variables in a dataset. More input features often make a predictive modeling task more challenging to model, more generally referred to as the curse of dimensionality. High-dimensionality statistics and dimensionality reduction techniques are often used for data visualization. Nevertheless these techniques can be used in applied machine learning to simplify a classification or regression dataset in order to better fit a predictive model.

Problem With Many Input Variables

If your data is represented using rows and columns, such as in a spreadsheet, then the input variables are the columns that are fed as input to a model to predict the target variable. Input variables are also called features. We can consider the columns of data representing dimensions on an n-dimensional feature space and the rows of data as points in that space. This is a useful geometric interpretation of a dataset. Having a large number of dimensions in the feature space can mean that the volume of that space is very large, and in turn, the points that we have in that space (rows of data) often represent a small and non-representative sample. This can dramatically impact the performance of machine learning algorithms fit on data with many input features, generally referred to as the "curse of dimensionality."

Therefore, it is often desirable to reduce the number of input features. This reduces the number of dimensions of the feature space, hence the name "*dimensionality reduction*."

Dimensionality Reduction

Dimensionality reduction refers to techniques for reducing the number of input variables in training data.

When dealing with high dimensional data, it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the "essence" of the data. This is called dimensionality reduction.

Fewer input dimensions often mean correspondingly fewer parameters or a simpler structure in the machine learning model, referred to as degrees of freedom. A model with too many degrees of freedom is likely to overfit the training dataset and therefore may not perform well on new data.

It is desirable to have simple models that generalize well, and in turn, input data with few input variables. This is particularly true for linear models where the number of inputs and the degrees of freedom of the model are often closely related.

Techniques for Dimensionality Reduction

There are many techniques that can be used for dimensionality reduction.

- Feature Selection Methods
- Matrix Factorization
- Manifold Learning
- Auto encoder Methods

Feature Selection Methods

Feature selection is also called variable selection or attribute selection.

It is the automatic selection of attributes in your data (such as columns in tabular data) that are most relevant to the predictive modeling problem you are working on.

feature selection… is the process of selecting a subset of relevant features for use in model construction

Feature selection is different from dimensionality reduction. Both methods seek to reduce the number of attributes in the dataset, but a dimensionality reduction

method do so by creating new combinations of attributes, where as feature selection methods include and exclude attributes present in the data without changing them. Examples of dimensionality reduction methods include Principal Component Analysis, Singular Value Decomposition and Sammon's Mapping.

Feature selection is itself useful, but it mostly acts as a filter, muting out features that aren't useful in addition to your existing features.

Feature Selection Algorithms

Filter Methods

Filter feature selection methods apply a statistical measure to assign a scoring to each feature. The features are ranked by the score and either selected to be kept or removed from the dataset. The methods are often univariate and consider the feature independently, or with regard to the dependent variable. Some examples of some filter methods include the Chi squared test, information gain and correlation coefficient scores.

Wrapper Methods

Wrapper methods consider the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. A predictive model us used to evaluate a combination of features and assign a score based on model accuracy. The search process may be methodical such as a best-first search, it may stochastic such as a random hill-climbing algorithm, or it may use heuristics, like forward and backward passes to add and remove features. An example if a wrapper method is the recursive feature elimination algorithm.

Embedded Methods

Embedded methods learn which features best contribute to the accuracy of the model while the model is being created. The most common type of embedded feature selection methods are regularization methods. Regularization methods are also called penalization methods that introduce additional constraints into the

optimization of a predictive algorithm (such as a regression algorithm) that bias the model toward lower complexity (fewer coefficients). Examples of regularization algorithms are the LASSO, Elastic Net and Ridge Regression.

EVALUATION METRICS

Evaluation

- important for any text classification method
- key step to validate a newly proposed classification method

Contingency Table

- Let
  - $\mathcal{D}$: collection of documents
  - $\mathcal{D}_t$: subset composed of training documents
  - N_t: number of documents in $\mathcal{D}_t$
  - $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$: set of all L classes
- Further let
  - $\mathcal{T} : \mathcal{D}_t \times \mathcal{C} \rightarrow [0, 1]$: training set function
  - n_t: number of docs from training set $\mathcal{D}_t$ in class c_p
  - $\mathcal{F} : \mathcal{D} \times \mathcal{C} \rightarrow [0, 1]$: text classifiier function
  - n_f: number of docs from training set assigned to class c_p by the classifier

- Apply classifier to all documents in training set
- Contingency table is given by

$Case$	$\mathcal{T}(d_j, c_p) = 1$	$\mathcal{T}(d_j, c_p) = 0$	$Total$
$\mathcal{F}(d_j, c_p) = 1$	$n_{f,t}$	$n_f - n_{f,t}$	n_f
$\mathcal{F}(d_j, c_p) = 0$	$n_t - n_{f,t}$	$N_t - n_f - n_t + n_{f,t}$	$N_t - n_f$
$All\ docs$	n_t	$N_t - n_t$	N_t

  - $n_{f,t}$: number of docs that both the training and classifier functions assigned to class c_p
  - $n_t - n_{f,t}$: number of training docs in class c_p that were miss-classified
  - The remaining quantities are calculated analogously

Accuracy and Error

- Accuracy and error metrics, relative to a given class c_p

$$\begin{aligned} Acc(c_p) &= \frac{n_{f,t} + (N_t - n_f - n_t + n_{f,t})}{N_t} \\ Err(c_p) &= \frac{(n_f - n_{f,t}) + (n_t - n_{f,t})}{N_t} \\ Acc(c_p) + Err(c_p) &= 1 \end{aligned}$$

- These metrics are commonly used for evaluating classifiers
- Accuracy and error have disadvantages
  - consider classification with only two categories c_p and c_r
  - assume that out of 1,000 docs, 20 are in class c_p
  - a classifier that assumes all docs not in class c_p
    - accuracy = 98%
    - error = 2%

    which erroneously suggests a very good classifier

- Consider now a second classifier that correctly predicts 50% of the documents in c_p

	$\mathcal{T}(d_j, c_p) = 1$	$\mathcal{T}(d_j, c_p) = 0$	
$\mathcal{F}(d_j, c_p) = 1$	10	0	10
$\mathcal{F}(d_j, c_p) = 0$	10	980	990
all docs	20	980	1,000

- In this case, accuracy and error are given by

$$Acc(c_p) = \frac{10 + 980}{1,000} = 99\%$$

$$Err(c_p) = \frac{10 + 0}{1,000} = 1\%$$

- This classifier is much better than one that guesses that all documents are not in class c_p
- However, its accuracy is just 1% better, it increased from 98% to 99%
- This suggests that the two classifiers are almost equivalent, which is not the case.

Precision and Recall

- Variants of precision and recall metrics in IR
- Precision P and recall R relative to a class c_p

$$P(c_p) = \frac{n_{f,t}}{n_f} \qquad R(c_p) = \frac{n_{f,t}}{n_t}$$

- Precision is the fraction of all docs assigned to class c_p by the classifier that really belong to class c_p
- Recall is the fraction of all docs that belong to class c_p that were correctly assigned to class c_p

- Consider again the classifier illustrated below

	$\mathcal{T}(d_j, c_p) = 1$	$\mathcal{T}(d_j, c_p) = 0$	
$\mathcal{F}(d_j, c_p) = 1$	10	0	10
$\mathcal{F}(d_j, c_p) = 0$	10	980	990
all docs	20	980	1,000

- Precision and recall figures are given by

$$P(c_p) = \frac{10}{10} = 100\%$$

$$R(c_p) = \frac{10}{20} = 50\%$$

- Precision and recall
  - computed for every category in set $\mathcal{C}$
  - great number of values
    - makes tasks of comparing and evaluating algorithms more difficult
- Often convenient to combine precision and recall into a single quality measure
  - one of the most commonly used such metric: *F-measure*

F-measure

- F-measure is defined as

$$F_\alpha(c_p) = \frac{(\alpha^2 + 1)P(c_p)R(c_p)}{\alpha^2 P(c_p) + R(c_p)}$$

  - α: relative importance of precision and recall
  - when $\alpha = 0$, only precision is considered
  - when $\alpha = \infty$, only recall is considered
  - when $\alpha = 0.5$, recall is half as important as precision
  - when $\alpha = 1$, common metric called F_1-measure

$$F_1(c_p) = \frac{2P(c_p)R(c_p)}{P(c_p) + R(c_p)}$$

- Consider again the the classifier illustrated below

	$\mathcal{T}(d_j, c_p) = 1$	$\mathcal{T}(d_j, c_p) = 0$	
$\mathcal{F}(d_j, c_p) = 1$	10	0	10
$\mathcal{F}(d_j, c_p) = 0$	10	980	990
all docs	20	980	1,000

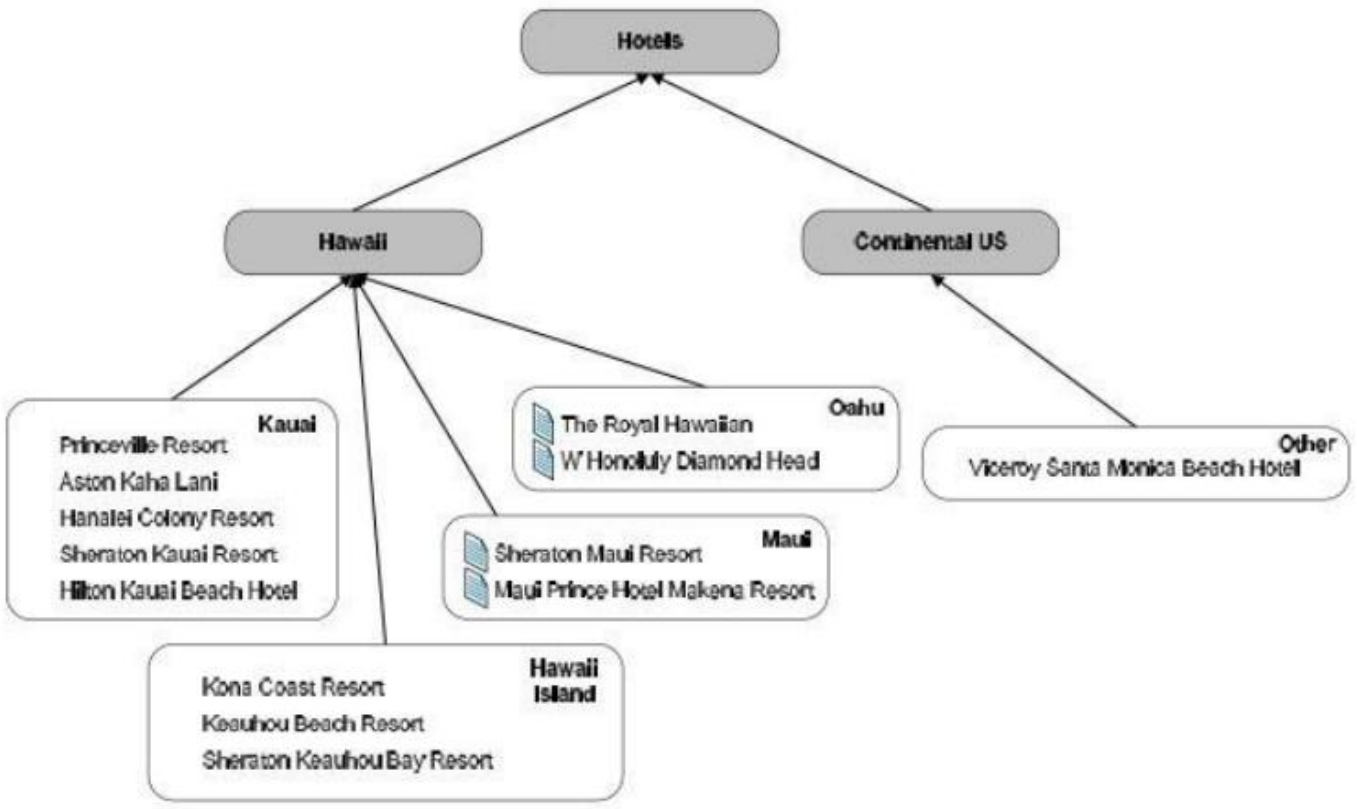
- For this example, we write

$$F_1(c_p) = \frac{2 * 1 * 0.5}{1 + 0.5} \sim 67\%$$

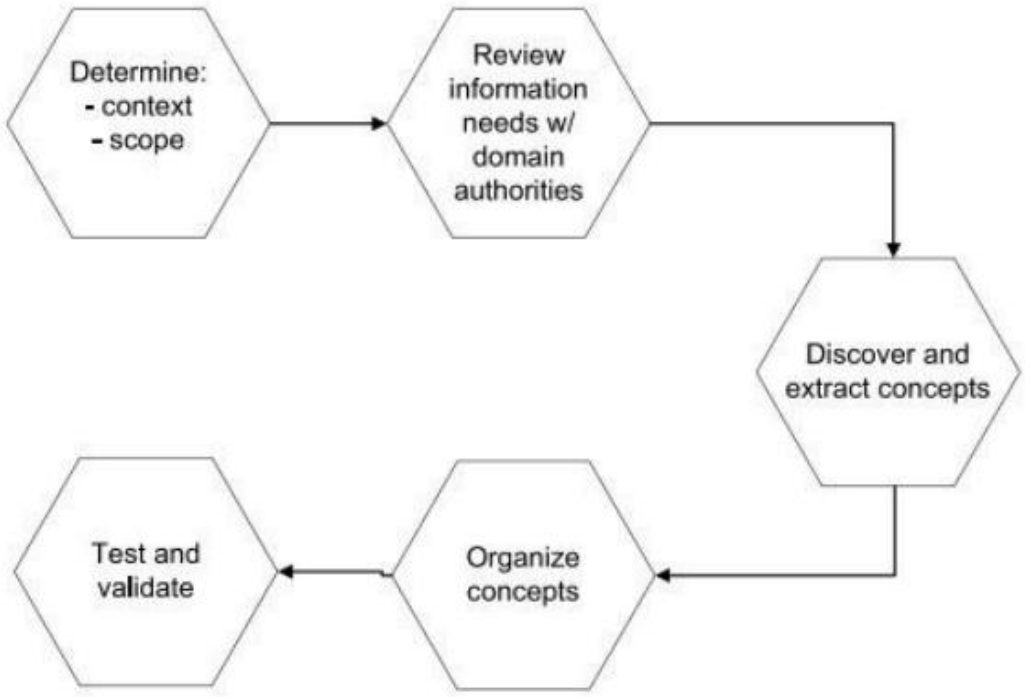
ORGANIZING THE CLASSES TAXONOMIES

Taxonomies

- Labels provide information on semantics of each class
- Lack of organization of classes restricts comprehension and reasoning
- Hierarchical organization of classes
  - most appealing to humans
  - hierarchies allow reasoning with more generic concepts
  - also provide for specialization, which allows breaking up a larger set of entities into subsets
- To organize classes hierarchically use
  - specialization
  - generalization
  - sibling relations
- Classes organized hierarchically compose a **taxonomy**
  - relations among classes can be used to fine tune the classifier
  - taxonomies make more sense when built for a specific domain of knowledge



- Taxonomies are built **manually** or **semi-automatically**
- Process of building a taxonomy:



- Manual taxonomies tend to be of superior quality
  - better reflect the information needs of the users
- Automatic construction of taxonomies
  - needs more research and development
- Once a taxonomy has been built
  - documents can be classified according to its concepts
  - can be done manually or automatically
  - automatic classification is advanced enough to work well in practice

INDEXING AND SEARCHING

Introduction

- Although **efficiency** might seem a secondary issue compared to **effectiveness**, it can rarely be neglected in the design of an IR system
- **Efficiency in IR systems**: to process user queries with minimal requirements of computational resources
- As we move to larger-scale applications, efficiency becomes more and more important
  - For example, in Web search engines that index terabytes of data and serve hundreds or thousands of queries per second

- **Index**: a data structure built from the text to speed up the searches
- In the context of an IR system that uses an index, the efficiency of the system can be measured by:
  - **Indexing time:** Time needed to build the index
  - **Indexing space:** Space used during the generation of the index
  - **Index storage:** Space required to store the index
  - **Query latency:** Time interval between the arrival of the query and the generation of the answer
  - **Query throughput:** Average number of queries processed per second

- When a text is updated, any index built on it must be updated as well
- Current indexing technology is not well prepared to support very frequent changes to the text collection
- **Semi-static collections:** collections which are updated at reasonable regular intervals (say, daily)
- Most real text collections, including the Web, are indeed semi-static
  - For example, although the Web changes very fast, the crawls of a search engine are relatively slow
- For maintaining freshness, incremental indexing is used

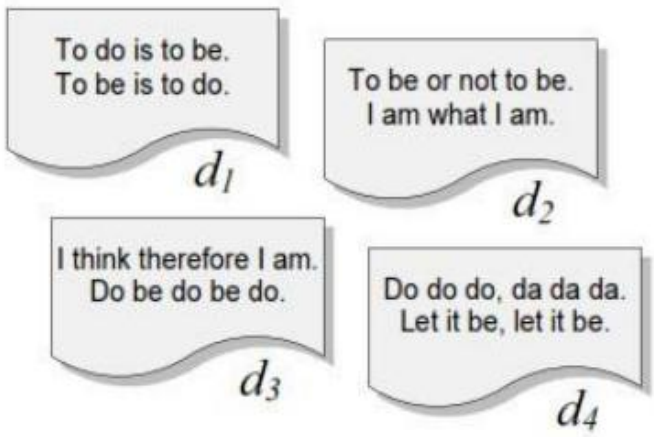
INVERTED INDEXES

Basic Concepts

- **Inverted index**: a word-oriented mechanism for indexing a text collection to speed up the searching task
- The inverted index structure is composed of two elements: the **vocabulary** and the **occurrences**
- The vocabulary is the set of all different words in the text
- For each word in the vocabulary the index stores the documents which contain that word (inverted index)

- **Term-document matrix**: the simplest way to represent the documents that contain each word of the vocabulary

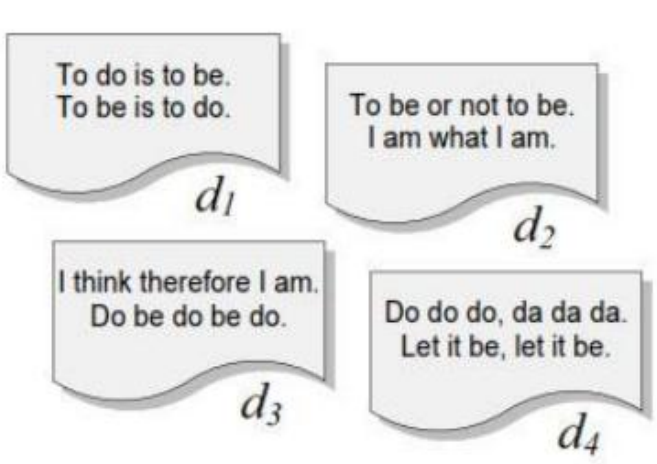
Vocabulary	n_i	d_1	d_2	d_3	d_4
to	2	4	2	-	-
do	3	2	-	3	3
is	1	2	-	-	-
be	4	2	2	2	2
or	1	-	1	-	-
not	1	-	1	-	-
I	2	-	2	2	-
am	2	-	2	1	-
what	1	-	1	-	-
think	1	-	-	1	-
therefore	1	-	-	1	-
da	1	-	-	-	3
let	1	-	-	-	2
it	1	-	-	-	2



- The main problem of this simple solution is that it requires too much space
- As this is a sparse matrix, the solution is to associate a list of documents with each word
- The set of all those lists is called the **occurrences**

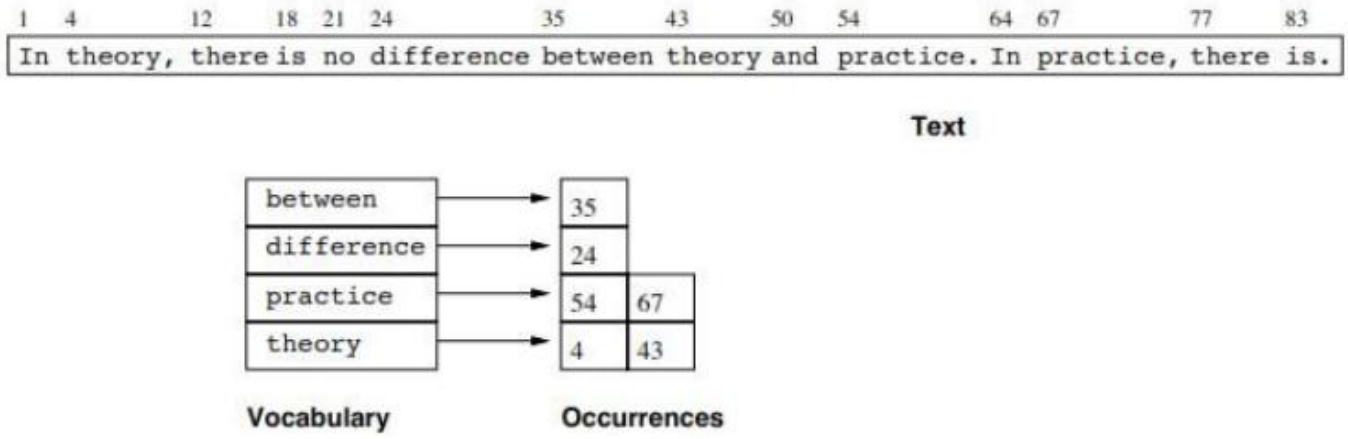
- Basic inverted index

Vocabulary	n_i	Occurrences as inverted lists
to	2	[1,4],[2,2]
do	3	[1,2],[3,3],[4,3]
is	1	[1,2]
be	4	[1,2],[2,2],[3,2],[4,2]
or	1	[2,1]
not	1	[2,1]
I	2	[2,2],[3,2]
am	2	[2,2],[3,1]
what	1	[2,1]
think	1	[3,1]
therefore	1	[3,1]
da	1	[4,3]
let	1	[4,2]
it	1	[4,2]



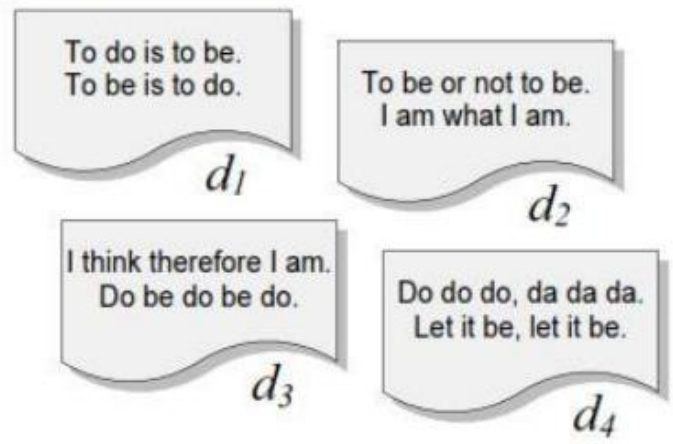
Full Inverted Indexes

- The basic index is not suitable for answering phrase or proximity queries
- Hence, we need to add the positions of each word in each document to the index (full inverted index)

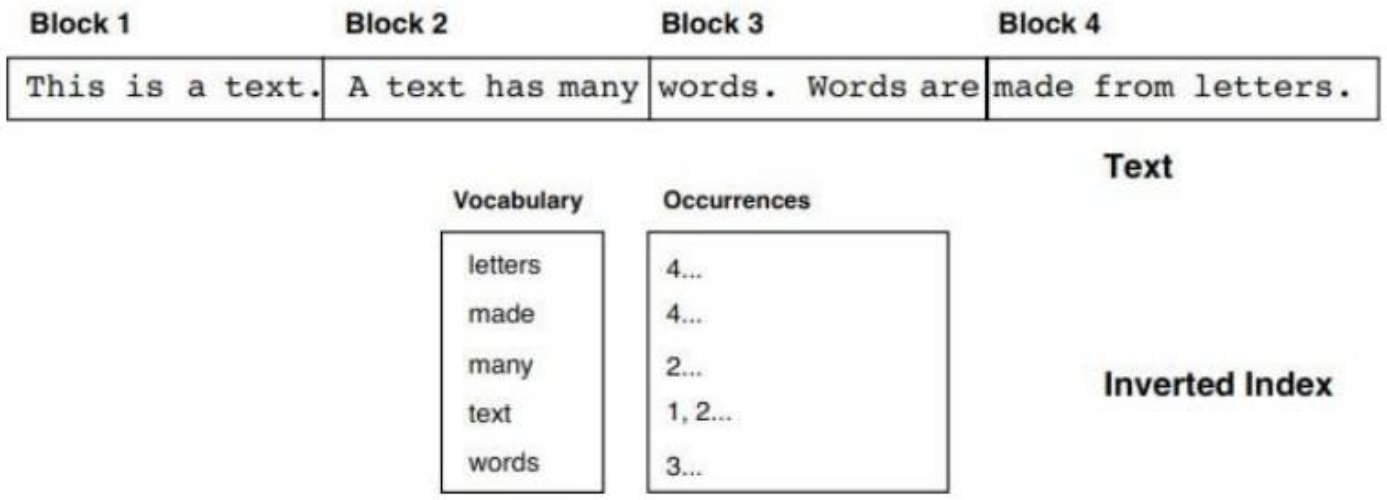


- In the case of multiple documents, we need to store one occurrence list per term-document pair

Vocabulary	n_i	Occurrences as full inverted lists
to	2	[1,4,[1,4,6,9]],[2,2,[1,5]]
do	3	[1,2,[2,10]],[3,3,[6,8,10]],[4,3,[1,2,3]]
is	1	[1,2,[3,8]]
be	4	[1,2,[5,7]],[2,2,[2,6]],[3,2,[7,9]],[4,2,[9,12]]
or	1	[2,1,[3]]
not	1	[2,1,[4]]
I	2	[2,2,[7,10]],[3,2,[1,4]]
am	2	[2,2,[8,11]],[3,1,[5]]
what	1	[2,1,[9]]
think	1	[3,1,[2]]
therefore	1	[3,1,[3]]
da	1	[4,3,[4,5,6]]
let	1	[4,2,[7,10]]
it	1	[4,2,[8,11]]



- The space required for the vocabulary is rather small
- Heaps' law: the vocabulary grows as $O(n^{\beta})$, where
  - n is the collection size
  - β is a collection-dependent constant between 0.4 and 0.6
- For instance, in the TREC-3 collection, the vocabulary of 1 gigabyte of text occupies only 5 megabytes
- This may be further reduced by stemming and other normalization techniques
- The occurrences demand much more space
- The extra space will be $O(n)$ and is around
  - 40% of the text size if stopwords are omitted
  - 80% when stopwords are indexed
- Document-addressing indexes are smaller, because only one occurrence per file must be recorded, for a given word
- Depending on the document (file) size, document-addressing indexes typically require 20% to 40% of the text size
- To reduce space requirements, a technique called **block addressing** is used
- The documents are divided into blocks, and the occurrences point to the blocks where the word appears



- The Table below presents the projected space taken by inverted indexes for texts of different sizes

Index granularity	Single document (1 MB)		Small collection (200 MB)		Medium collection (2 GB)	
Addressing words	45%	73%	36%	64%	35%	63%
Addressing documents	19%	26%	18%	32%	26%	47%
Addressing 64K blocks	27%	41%	18%	32%	5%	9%
Addressing 256 blocks	18%	25%	1.7%	2.4%	0.5%	0.7%

- The blocks can be of fixed size or they can be defined using the division of the text collection into documents
- The division into blocks of fixed size improves efficiency at retrieval time
  - This is because larger blocks match queries more frequently and are more expensive to traverse
- This technique also profits from *locality of reference*
  - That is, the same word will be used many times in the same context and all the references to that word will be collapsed in just one reference

SEARCHING

Searching

- Searching a single word is made by comparing its bit mask W with the bit masks B_i of all the text blocks
- Whenever $(W \ \& \ B_i = W)$, where $\&$ is the bit-wise AND, the text block **may** contain the word
- Hence, an online traversal must be performed to verify if the word is actually there
- This traversal cannot be avoided as in inverted indexes (except if the risk of a false match is accepted)

- This scheme is more efficient to search phrases and reasonable proximity queries
  - This is because **all** the words must be present in a block in order for that block to hold the phrase or the proximity query
- Hence, the bit-wise OR of all the query masks is searched, so that **all** their bits must be present
  - This reduces the probability of false drops
- Some care has to be exercised at block boundaries, to avoid missing a phrase which crosses a block limit
- To search phrases of j words or proximities of up to j words, consecutive blocks must overlap in $j-1$ words
- This is the only indexing scheme which improves in phrase searching

---

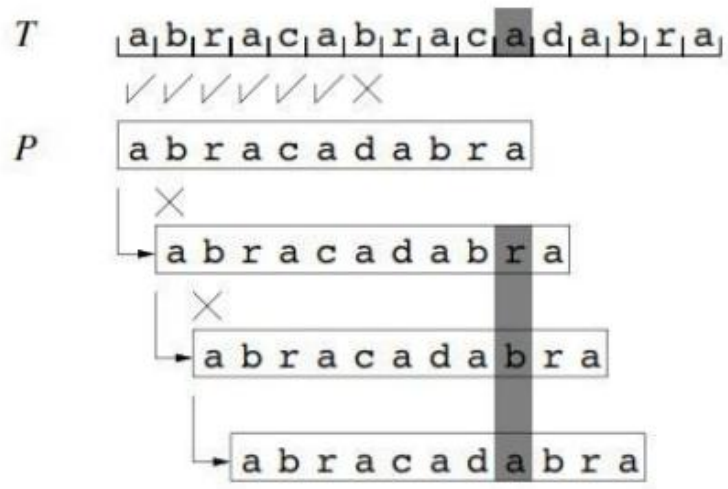
SEQUENTIAL SEARCHING

Sequential Searching

- In general the sequential search problem is:
  - Given a text $T = t_1 t_2 \ldots t_n$ and a pattern denoting a set of strings $\mathcal{P}$, find all the occurrences of the strings of $\mathcal{P}$ in T
- **Exact string matching:** the simplest case, where the pattern denotes just a single string $P = p_1 p_2 \ldots p_m$
- This problem subsumes many of the basic queries, such as word, prefix, suffix, and substring search
- We assume that the strings are sequences of characters drawn from an alphabet Σ of size σ

Simple Strings: Brute Force

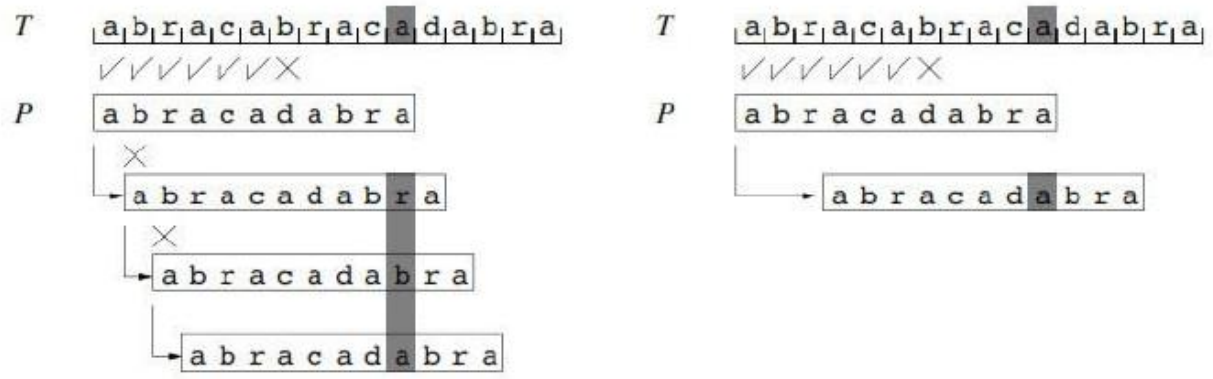
- The **brute force** algorithm:
  - Try out all the possible pattern positions in the text and checks them one by one
- More precisely, the algorithm slides a **window** of length m across the text, $t_{i+1}t_{i+2}\ldots t_{i+m}$ for $0 \leq i \leq n-m$
- Each window denotes a potential pattern occurrence that must be verified
- Once verified, the algorithm slides the window to the next position
- A sample text and pattern searched for using brute force
  - The first text window is `abracabraca`
  - After verifying that it does not match P, the window is shifted by one position



Simple Strings: Horspool

- **Horspool's algorithm** is in the fortunate position of being very simple to understand and program
- It is the fastest algorithm in many situations, especially when searching natural language texts
- Horspool's algorithm uses the previous idea to shift the window in a smarter way
- A table d indexed by the characters of the alphabet is precomputed:
  - $d[c]$ tells how many positions can the window be shifted if the final character of the window is c
- In other words, $d[c]$ is the distance from the end of the pattern to the last occurrence of c in P, excluding the occurrence of p_m

- The Figure repeats the previous example, now also applying Horspool's shift



- Pseudocode for Horspool's string matching algorithm

Horspool $(T = t_1 t_2 \dots t_n,\ P = p_1 p_2 \dots p_m)$

(1) **for** $c \in \Sigma$ **do** $d[c] \leftarrow m$
(2) **for** $j \leftarrow 1 \dots m-1$ **do** $d[p_j] \leftarrow m - j$
(3) $i \leftarrow 0$
(4) **while** $i \leq n - m$ **do**
(5) $\quad j \leftarrow 1$
(6) $\quad$ **while** $j \leq m \ \wedge\ t_{i+j} = p_j$ **do** $j \leftarrow j + 1$
(7) $\quad$ **if** $j > m$ **then** report an occurrence at text position $i + 1$
(8) $\quad i \leftarrow i + d[t_{i+m}]$

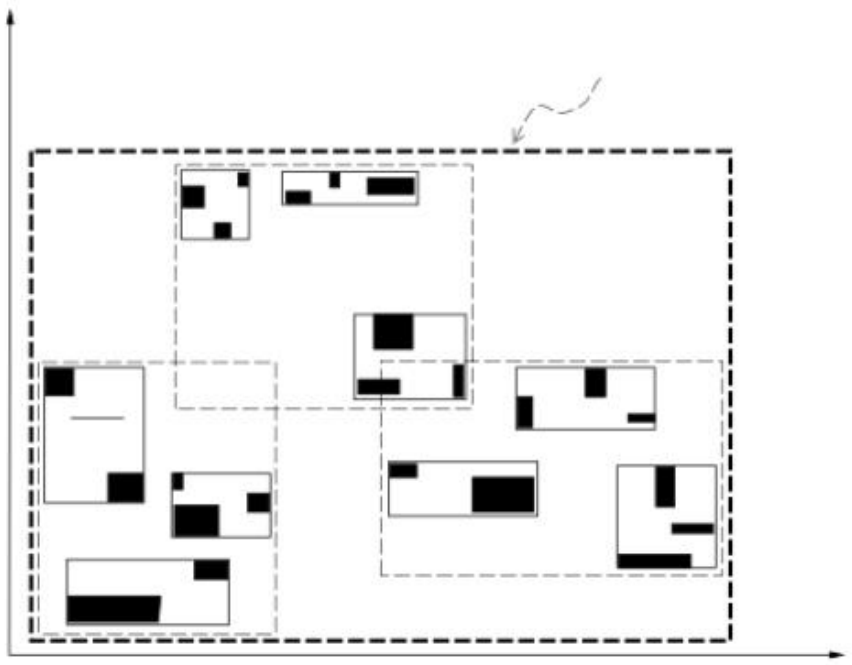
MULTI-DIMENSIONAL INDEXING

Multi-dimensional Indexing

- In multimedia data, we can represent every object by several numerical features
- For example, imagine an image from where we can extract a color histogram, edge positions, etc
- One way to search in this case is to map these object features into points in a multi-dimensional space
- Another approach is to have a distance function for objects and then use a distance based index
- The main mapping methods form three main classes:
  - R^*-trees and the rest of the R-tree family,
  - linear quadtrees,
  - grid-files

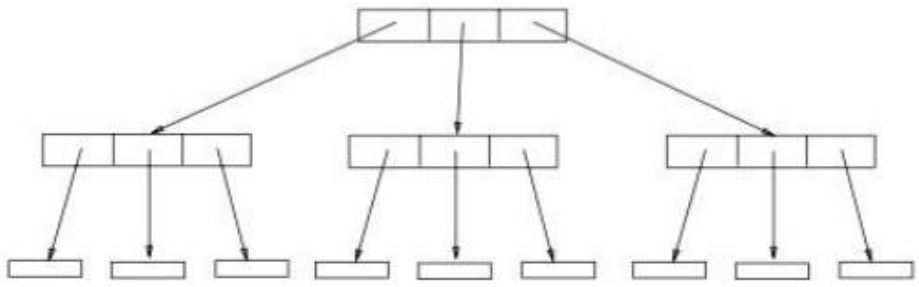
- The R-tree-based methods seem to be most robust for higher dimensions
- The R-tree represents a spatial object by its minimum bounding rectangle (MBR)
- Data rectangles are grouped to form parent nodes, which are recursively grouped, to form grandparent nodes and, eventually, a tree hierarchy
- **Disk pages** are consecutive byte positions on the surface of the disk that are fetched with one disk access
- The goal of the insertion, split, and deletion routines is to give trees that will have good clustering

- Figure below illustrates data rectangles (in black), organized in an R-tree with fanout 3



Multi-dimensional Search

- A range query specifies a region of interest, requiring all the data regions that intersect it
- To answer this query, we first retrieve a superset of the qualifying data regions:
  - We compute the MBR of the query region, and then we recursively descend the R-tree, excluding the branches whose MBRs do not intersect the query MBR
  - Thus, the R-tree will give us quickly the data regions whose MBR intersects the MBR of the query region
  - The retrieved data regions will be further examined for intersection with the query region
- The data structure of the R-tree for the previous figure is (fanout = 3)



PART-A

1 Integrate the problems of k-means method
2 Define the characterization of text classification.
3 Summarize Evaluation metrics with example.
4 What are the types of data in clustering analysis?
5 Point out the advantages and disadvantages of Decision Tree algorithm.
6 Show the applications SVM Classifier
7 Illustrate the advantages of Naive Bayes.
8 Assess how to measure distance of clusters?
9 Distinguish Supervised learning and Unsupervised Learning.
10 Summarize the good clustering approaches.
11 Define Supervised and Unsupervised algorithm
12 What is Hash-based Dictionary in Indexing?
13 Describe two types of index in detail.
14 Explain Sequential search in detail
15 Describe Brute Force in Sequential Search.
16 Distinguish Suffix Trees and Suffix arrays
17 What is Range queries and Nearest-Neighbour Queries?
18 Differentiate two types of multi-dimensional indexing.
19 Classify Dimension reduction in detail.
20 Discuss between clustering and classification.

PART-B

1 (i) Define Topic detection and tracking, Clustering in TDT. (4)
(ii) Examine in detail about Cluster Analysis in Text Clustering.(9)
2 Define Clustering in Metric space with application to Information Retrieval (13)
3 (i) Evaluate the Agglomerative Clustering and HAC in detail. (7)
(ii) Discuss the Types of data and evaluate it using any one clustering techniques. (6)
4 i) Summarize on Clustering Algorithms. (6)
ii) Evaluate on the various classification methods of Text. (7)
5 i) Analyze the working of Nearest Neighbor algorithm along with one representation. (7)
ii) Analyze the K-Means Clustering method and the problems in it. (6)
6 Analyze about Decision Tree Algorithm with illustration. (13)
7 Examine Inverted index and Forward index (13)
8 i) Discuss in detail about Text Classification. (7)
ii) Explain B Tree Index in detail (6)
9 i) Apply Naïve Bayes Algorithm for an example. (7)
ii) Demonstrate its working in detail. (6)
10 Analyse single Dimension Index in detail (13)
11 Define Knuth Morris Pratt algorithm in detail (13)
12 i) Construct B+ Tree Index in detail (6)

ii) Summarize the significance of SVM classifier in detail. (7)

13 Examine Single dimensional and multi-dimensional index. (13)

14 Explain Sequential search in detail (13)

PART-C

1 Rank the impacts of Categorization and clustering of text in the mining with the suitable examples. (8)

Detailed about KNN Classifier (7)

2 Design a Plan to overcome the gap in decision theoretic approach for evaluation in text mining. (15)

3 Compare two types of Dimensional Index in detail with example (15)

4 Estimate R Tree index and R+ Tree index (15)