REVA UNIVERSITY
Bengaluru, India

M23DE0201 – Machine Learning

II Semester MCA D Academic Year : 2024 - 2025

School of Computer Science and Applications

Dr.P SREE LAKSHMI
Assistant Professor

UNIT 2: **[14 Hours]**

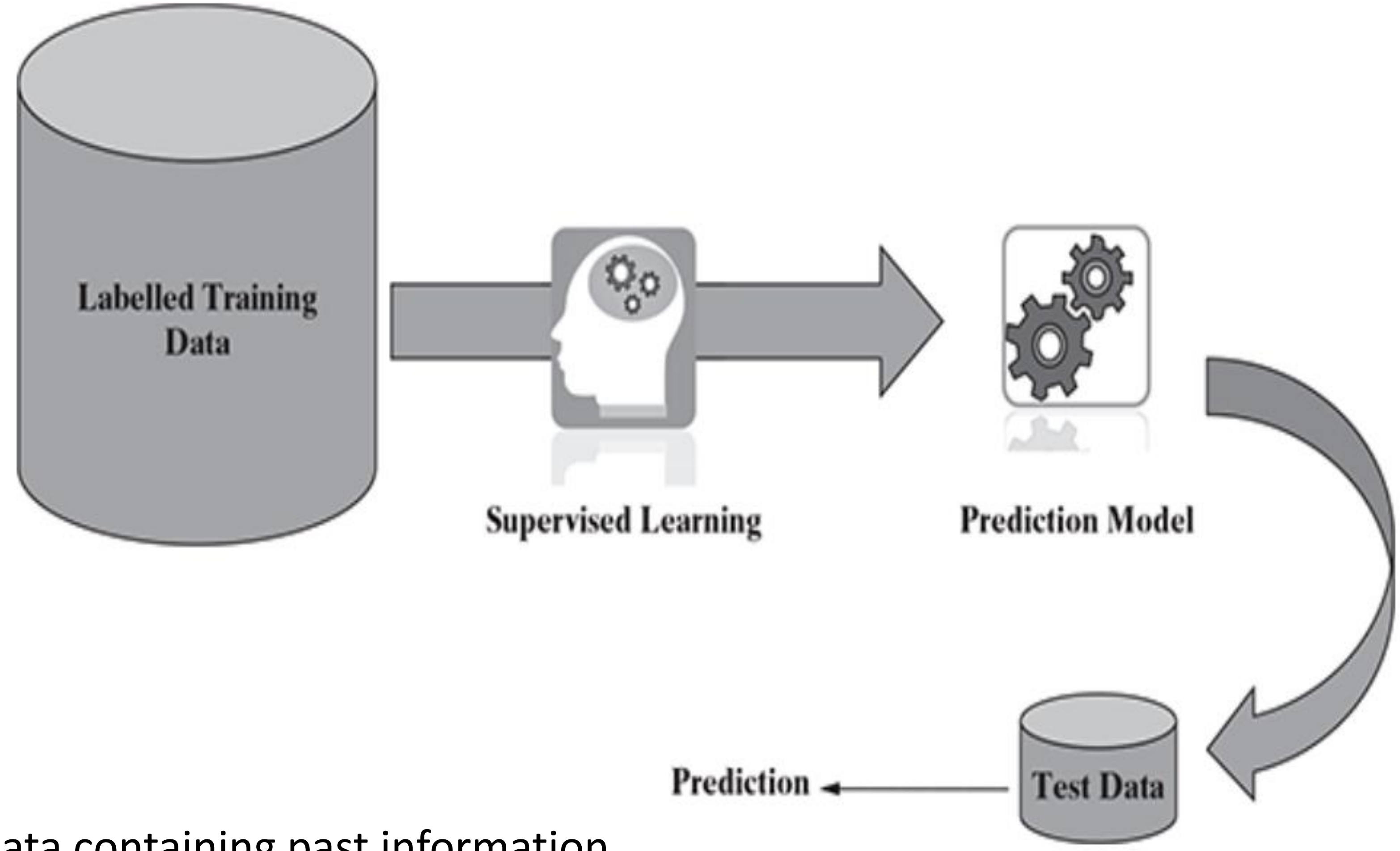
Supervised Learning:

- Introduction
- Classification
- Linear Regression
- k-Nearest Neighbor
- Linear models
- Decision Trees
- Naive Bayes Classifiers
- Support Vector Machine – Soft Margin and Non-Linear SVM classification
- Neural Networks - The Perceptron, MLP and Backpropagation, Train a DNN, Construction and Execution phase, How to use the Neural Network, Fine-tuning the Hyperparameters, The Number of Hidden Layers, Activation Functions.
- Visual Cortex Architecture, Convolutional Layers, Filters, Common CNN architectures, LexNet, AlexNet, GoogleNet, and ResNet

SUPERVISED LEARNING

- Learning from examples/ Labelled data
- Training set is given which acts as an example/ experience for the classes.
- System finds a description for each class (Classification Rule)
- Once the description has been formulated, it is used to predict the class for the new object.

SUPERVISED LEARNING



- Labeled training data containing past information comes as an input.
- Based on the training data, the machine builds a predictive model that can be used on test data to assign a label for each record in the test data.

- It is a machine learning task of learning a function that maps an input to an output based on example input-output pairs.
- It infers a function from *labeled training data* consisting of a set of *training examples*.
- In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*).
- A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples.

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

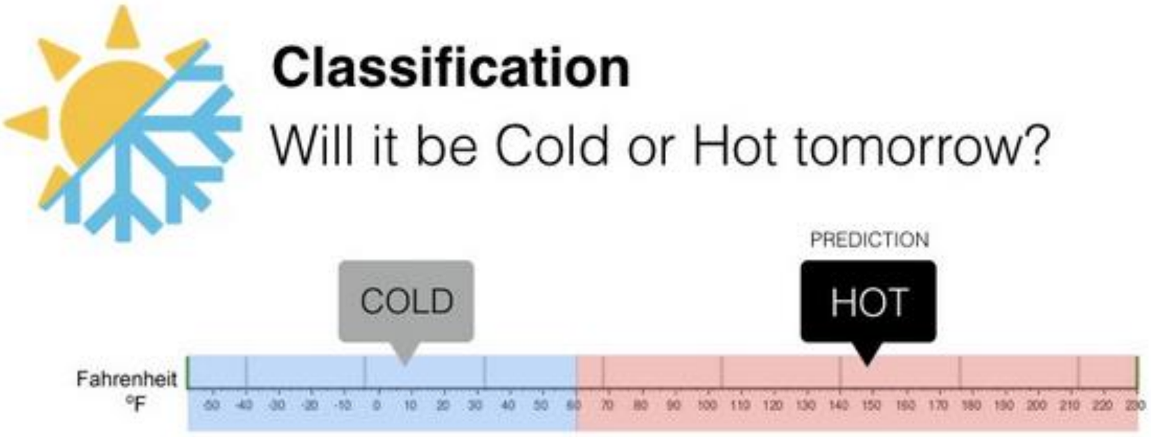
$$Y = f(X)$$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

- Supervised learning is the most popular paradigm for performing machine learning operations.
- It is widely used for datasets where there is a precise mapping between input-output data.
- The dataset, in this case, is **labeled,** meaning that the algorithm identifies the features explicitly and carries out predictions or classification accordingly.
- As the training period progresses, the algorithm can identify the relationships between the two variables such that we can predict a new outcome.

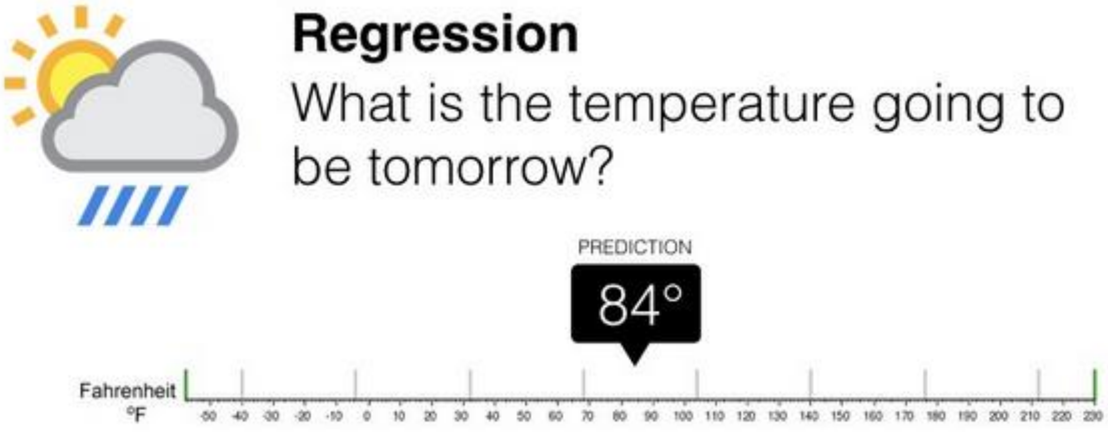
SUPERVISED LEARNING METHODS

1. **Classification**
2. **Regression**



Classification

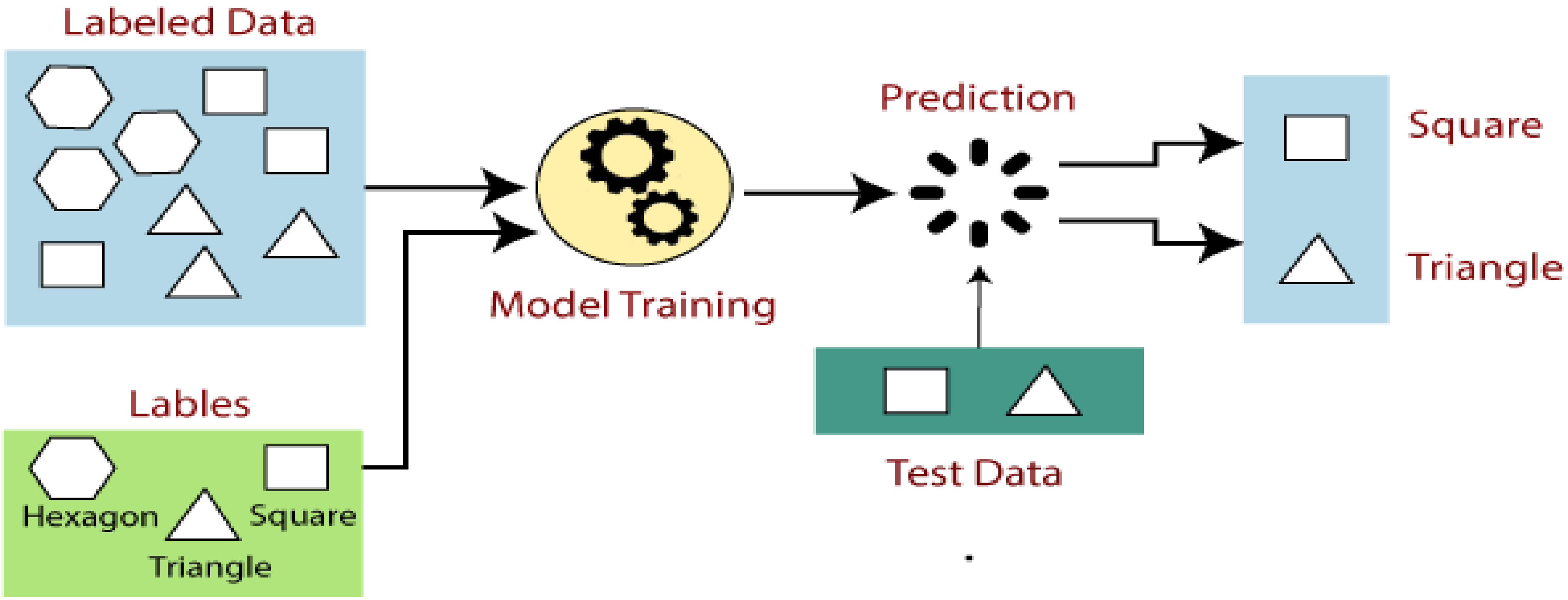
Will it be Cold or Hot tomorrow?



Regression

What is the temperature going to be tomorrow?

CLASSIFICATION- SUPERVISED LEARNING



CLASSIFICATION

- **Classification**
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data

Classification is the process of finding a **model** (or function) that describes and distinguishes data classes or concepts.

The model are derived based on the analysis of a set of

- **training data** (i.e., data objects for which the class labels are known).
- The model is used to predict the class label of objects for which the class label is unknown.

"How is the derived model presented?" \

The derived model may be represented in various forms, such as

- *classification rules* (i.e., *IF-THEN rules*),
- *decision trees*
- *Mathematical Formulae*, or
- *neural networks*

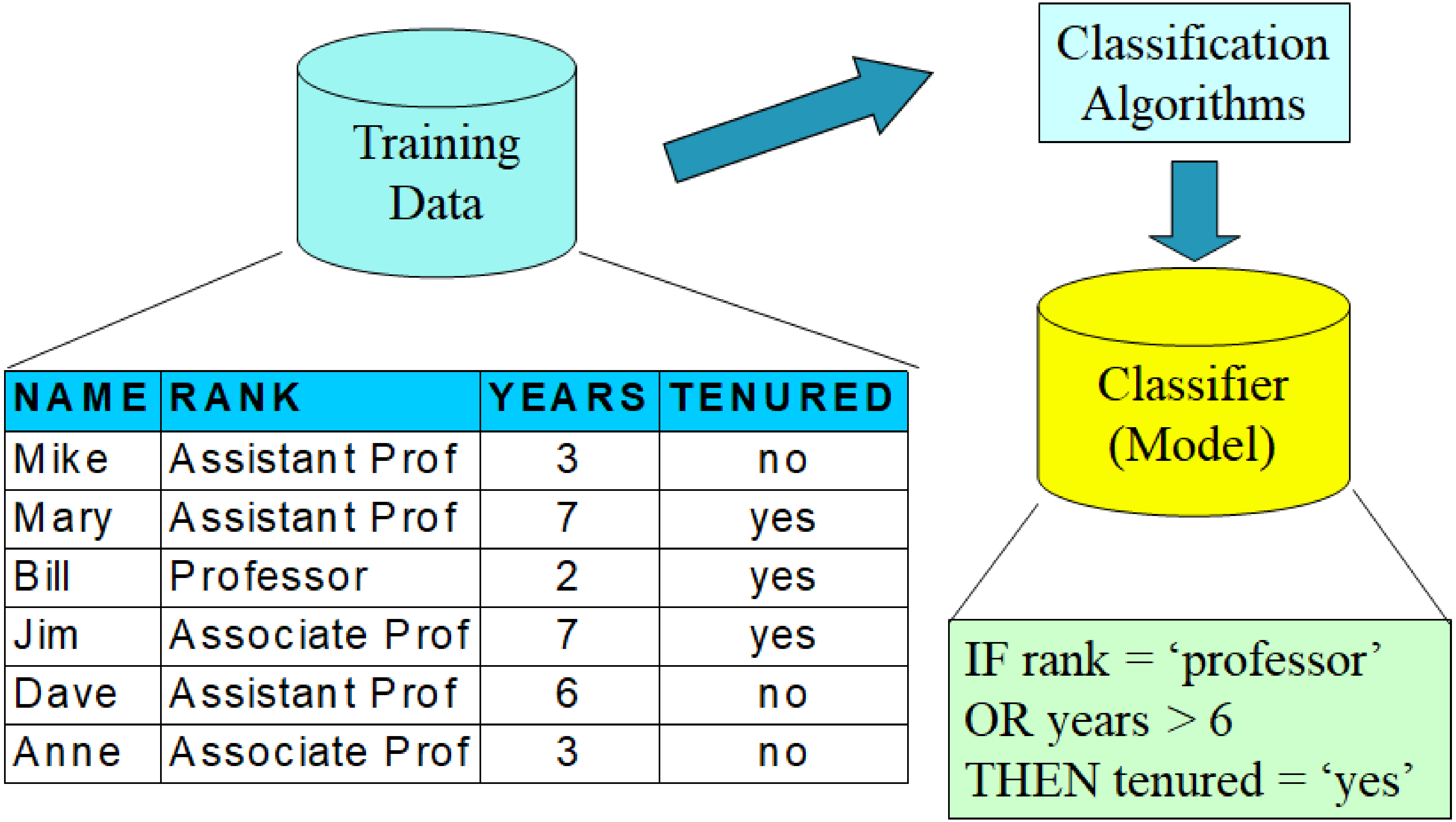
Other methods for constructing classification models, such as

- Naïve Bayesian classification
- support vector machines, and
- *k*-nearest-neighbor classification.

CLASSIFICATION—A TWO-STEP PROCESS

- **Model construction:** describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the **class label attribute**
  - The set of tuples used for model construction is a **training set**
  - The model is represented as classification rules, decision trees, or mathematical formulae
- **Model usage:** for classifying future or unknown objects
  - **Estimate the accuracy** of the model
    - The known label of the test sample is compared with the classified result from the model
    - **Accuracy** rate is the percentage of test set samples that are correctly classified by the model
    - **Test set** is independent of the training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to **classify new data**
- Note: If *the test set* is used to select models, it is called a **validation (test) set**

PROCESS (1): MODEL CONSTRUCTION



Training Data

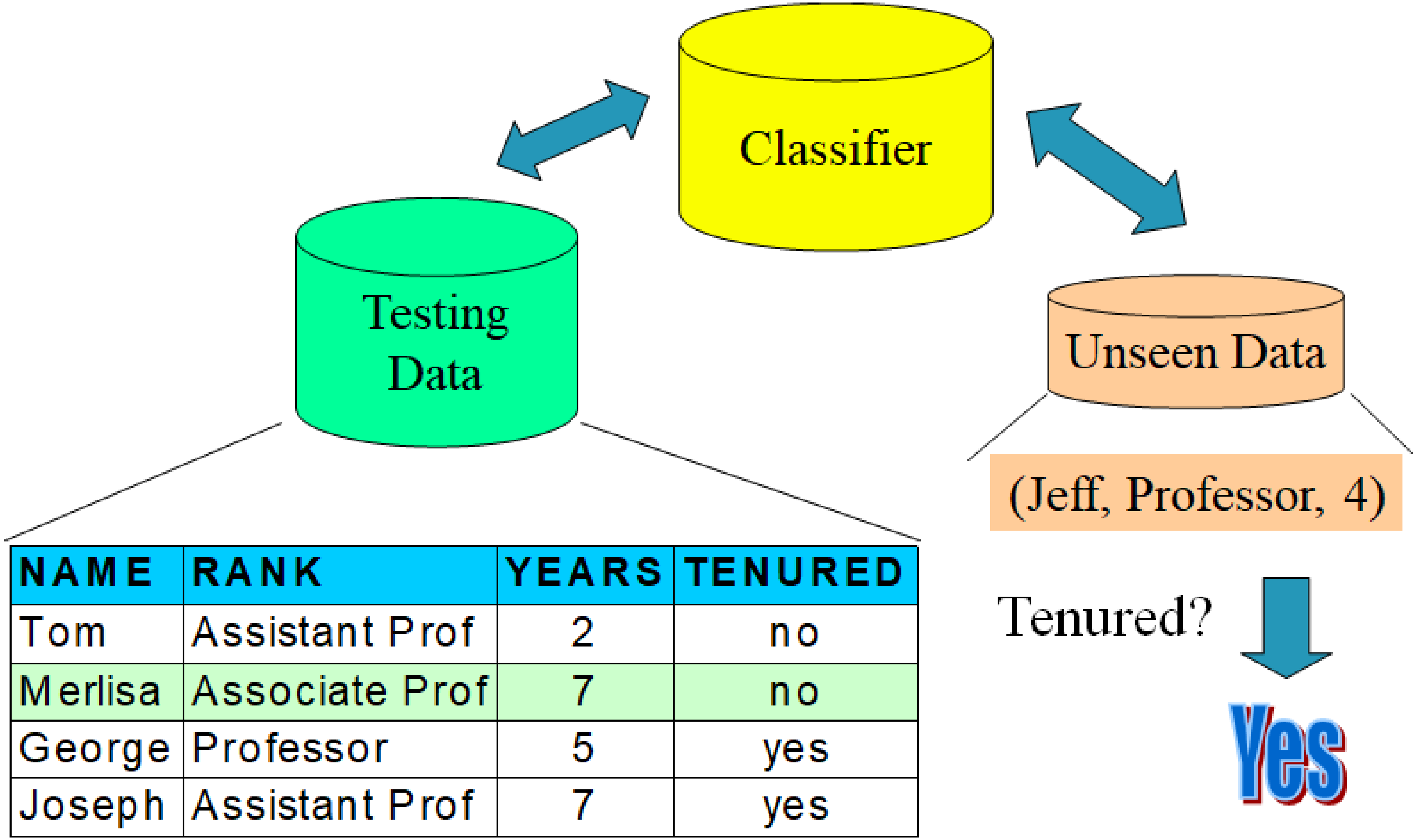
Classification Algorithms

Classifier (Model)

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

PROCESS (2): USING THE MODEL IN PREDICTION



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

CLASSIFICATION ALGORITHMS CATEGORIES:

1) Statistical based algorithms: Based directly on the use of statistical information

Ex: Regression (Division and Prediction), Bayesian classification

2) Distance based algorithms: Use similarity or distance measure to perform classification

Ex: Simple approach, KNN(K Nearest Neighbor)

3) Decision tree based algorithms: Uses Decision tree structure perform classification

Ex: ID3 ,C4.5, CART, Scalable DT Techniques SPRINT

K-NEAREST NEIGHBOR (KNN)

- Simplest Classification algorithms based on Supervised Learning technique.
- Assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- Stores all the available data and classifies a new data point based on the similarity.
- can be used for Regression as well as for Classification but mostly it is used for the Classification problems

KNN FEATURES

1. K-NN is a **non-parametric algorithm**

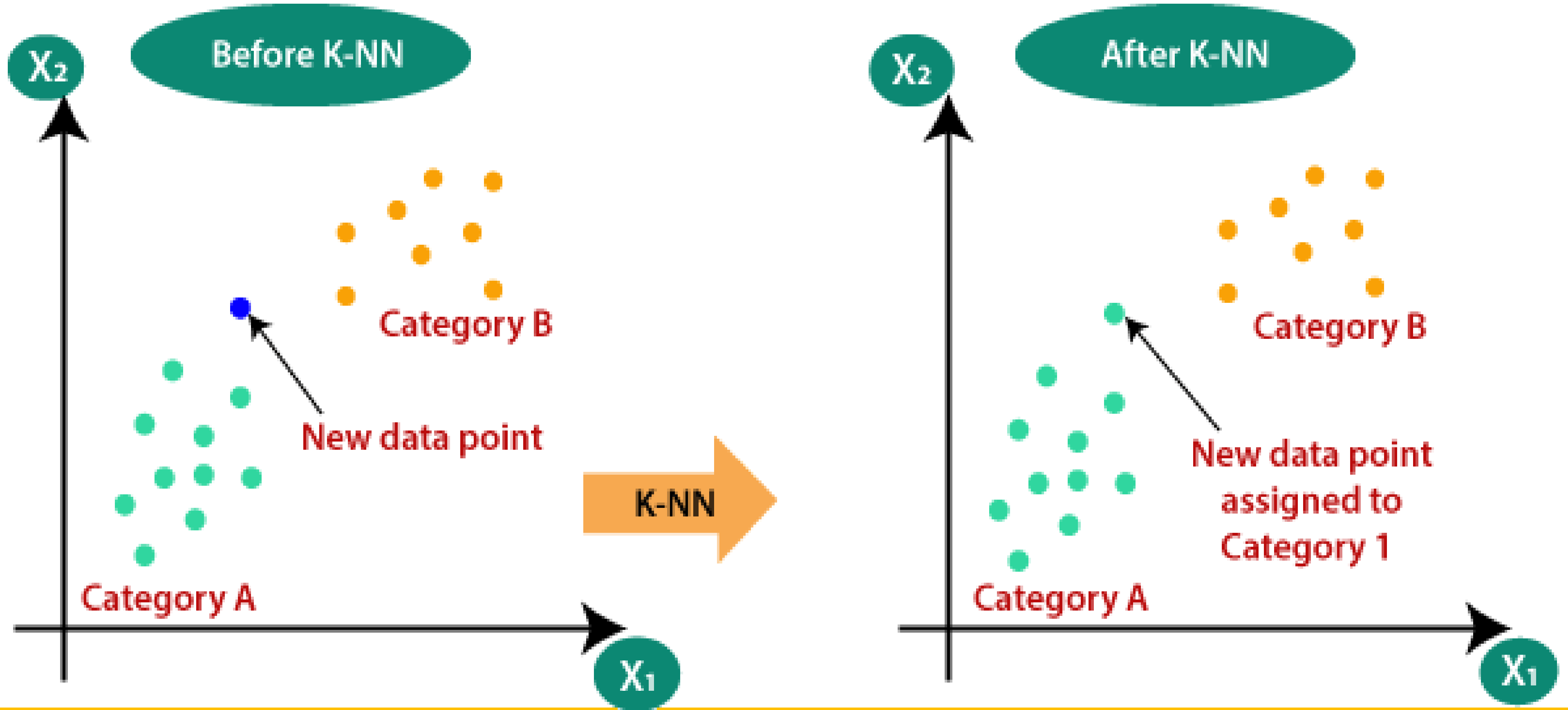
- **What does *non-parametric* mean?**
- Unlike parametric models (like linear regression) that learn a fixed number of parameters (e.g., slope and intercept), K-NN **does not learn a model** during training.
- Instead, **it memorizes the training data** and makes decisions **at the time of prediction**.
- **Why K-NN is non-parametric:**
  - **No Training Step**:
    K-NN simply stores the dataset and uses it during classification or regression. It doesn't estimate any coefficients or parameters.
  - **Flexible Decision Boundaries**:
    The decision boundary in K-NN depends entirely on the data distribution and the value of **K**. This allows K-NN to adapt to complex data shapes.
  - **Instance-Based Learning**:
    K-NN is a lazy learner — it defers computation until prediction time. It **relies on comparing the input with stored instances**.

2. A **lazy learner algorithm** because it does not learn from the training set immediately

- instead it stores the dataset and at the time of classification, it performs an action on the dataset.

WHY DO WE NEED A K-NN ALGORITHM?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm



WHERE TO USE KNN

Used in simple

- recommendation systems,
- image recognition technology, and
- decision-making models.

It is the algorithm companies like Netflix or Amazon use in order to recommend different movies to watch or books to buy.

HOW DOES K-NN WORK?

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of **K number of neighbors**

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

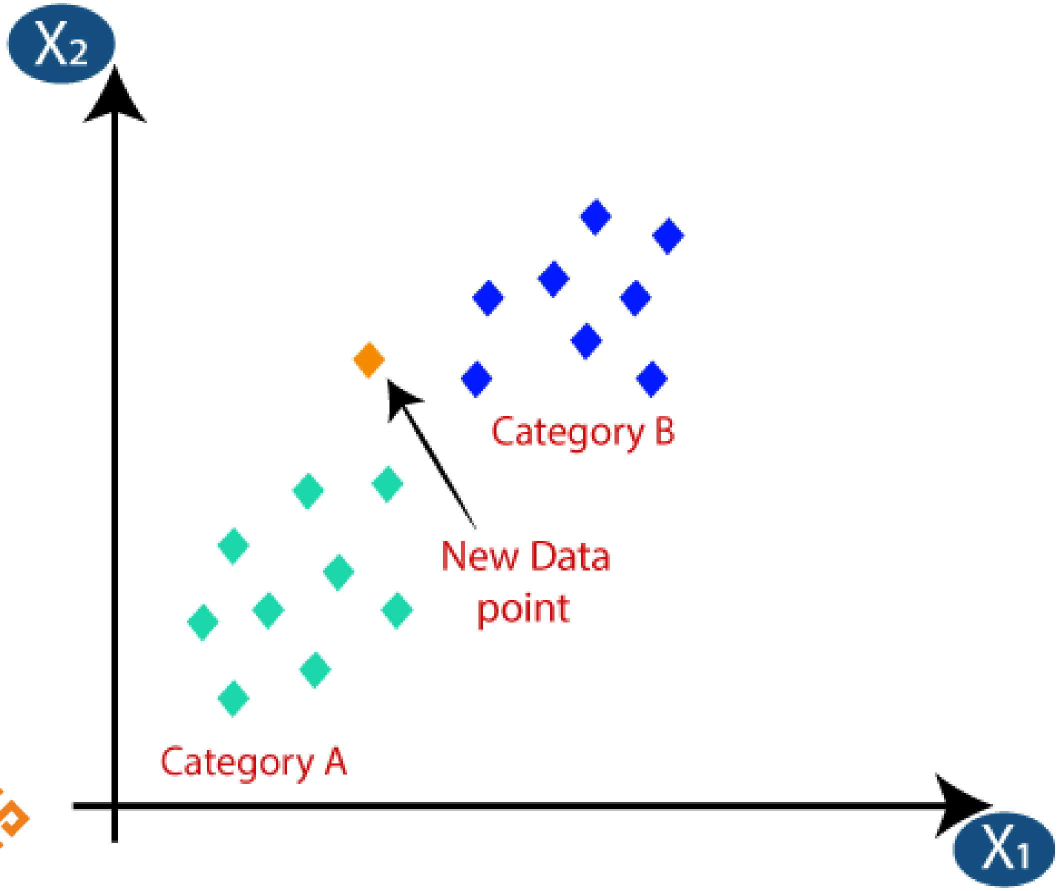
Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready.

EXAMPLE-KNN

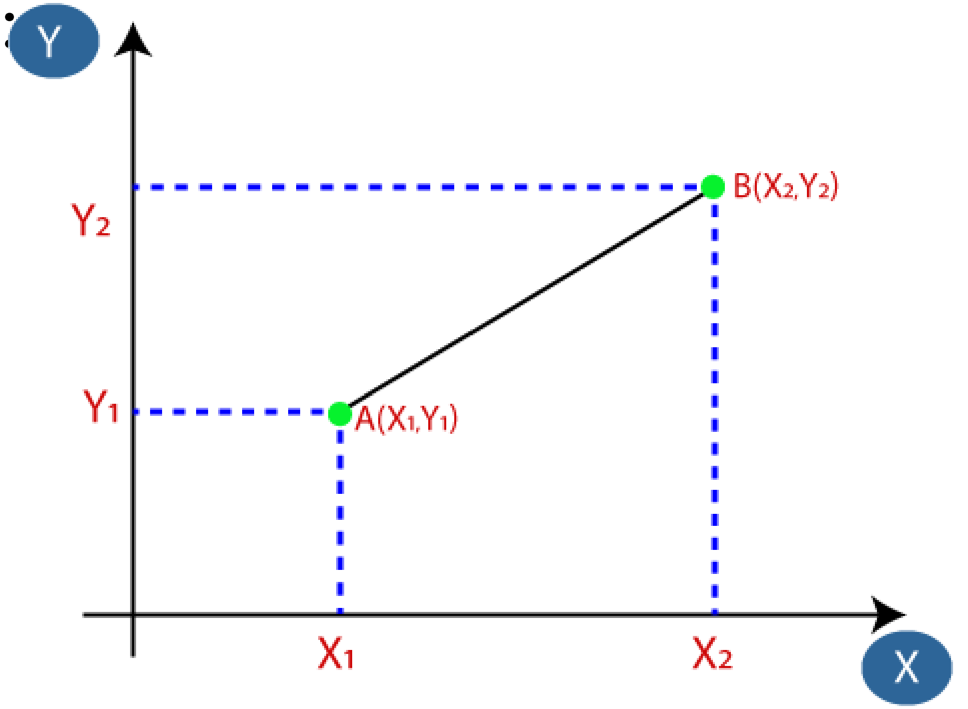
Suppose we have a new data point and we need to put it in the required category. Consider the below image



EXAMPLE-KNN

1. choose the number of neighbors, so we will choose the k=5.

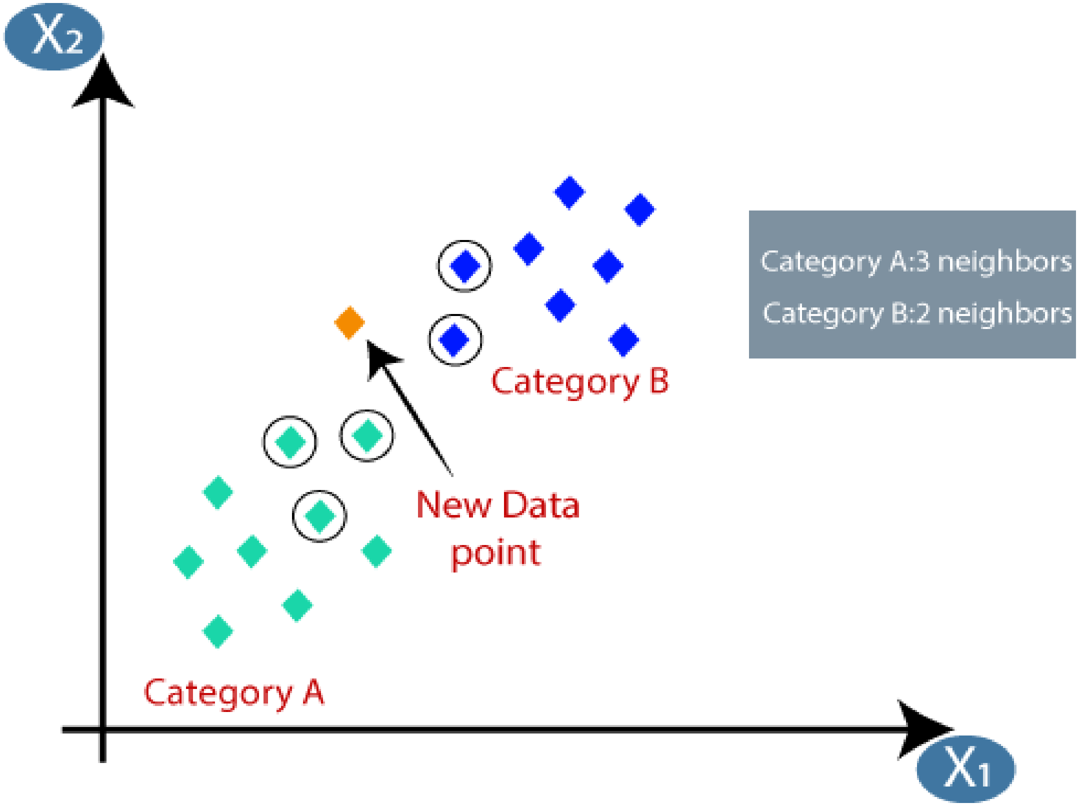
2. calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.



HOW TO SELECT THE VALUE OF K IN THE K-NN ALGORITHM?

- no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.

- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.

- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

It is simple to implement.

It is robust to the noisy training data

It can be more effective if the training data is large.

Quick calculation time.

Does not make assumptions about the data.

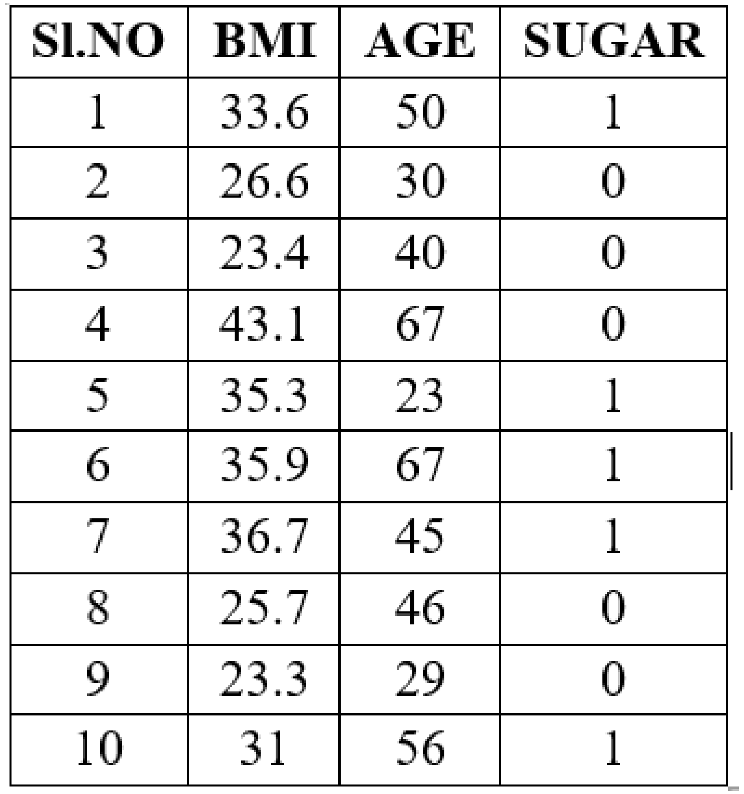
Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples
- Accuracy depends on the quality of the data.
- Must find an optimal k value (number of nearest neighbors).
- Poor at classifying data points in a boundary where they can be classified one way or another.

KNN EXAMPLE 1

Apply K Nearest Neighbor (K-NN) classifier to predict the diabetic patient with the given features BMI, age.

Consider Test example BMI=43.6, age=40, Sugar=? And assume k=3.



Sl.NO	BMI	AGE	SUGAR
1	33.6	50	1
2	26.6	30	0
3	23.4	40	0
4	43.1	67	0
5	35.3	23	1
6	35.9	67	1
7	36.7	45	1
8	25.7	46	0
9	23.3	29	0
10	31	56	1

KNN EXAMPLE 1

The given training dataset has 10 instances with two features BMI (Body Mass Index) and Age. Sugar is the target label.

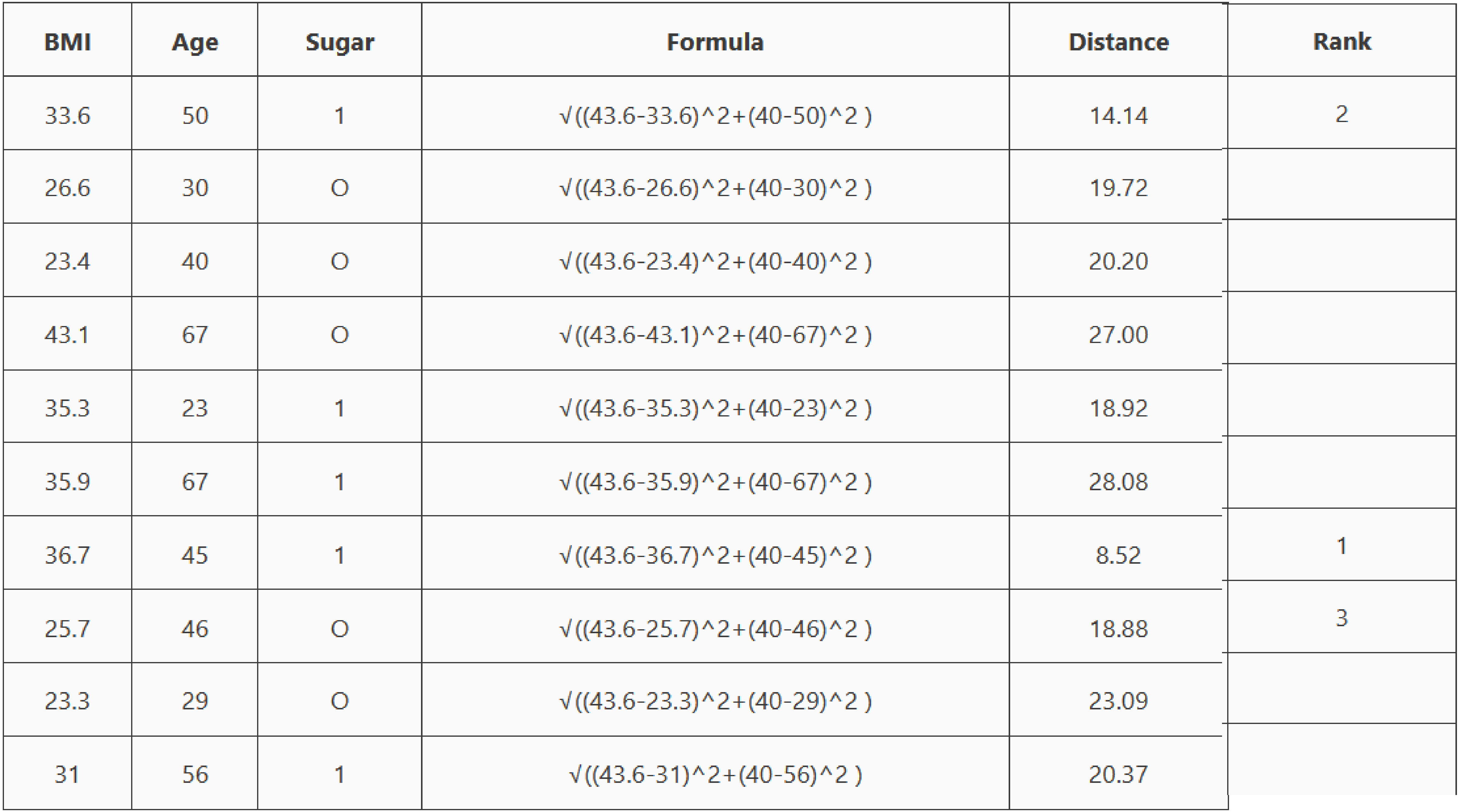
The target label has two possibilities 0 and 1. 0 means the diabetic patient has no sugar and 1 means the diabetic patient has sugar.

Given the dataset and new test instance, we need to find the distance from the new test instance to every training example.

Here we use the euclidean distance formula to find the distance.

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

Once you calculate the distance, the next step is to find the nearest neighbors based on the value of k. In this case, the value of k is 3. Hence we need to find 3 nearest neighbors.



BMI	Age	Sugar	Formula	Distance	Rank
33.6	50	1	√((43.6-33.6)^2+(40-50)^2)	14.14	2
26.6	30	O	√((43.6-26.6)^2+(40-30)^2)	19.72	
23.4	40	O	√((43.6-23.4)^2+(40-40)^2)	20.20	
43.1	67	O	√((43.6-43.1)^2+(40-67)^2)	27.00	
35.3	23	1	√((43.6-35.3)^2+(40-23)^2)	18.92	
35.9	67	1	√((43.6-35.9)^2+(40-67)^2)	28.08	
36.7	45	1	√((43.6-36.7)^2+(40-45)^2)	8.52	1
25.7	46	O	√((43.6-25.7)^2+(40-46)^2)	18.88	3
23.3	29	O	√((43.6-23.3)^2+(40-29)^2)	23.09	
31	56	1	√((43.6-31)^2+(40-56)^2)	20.37	

- Now, we need to apply the majority voting technique to decide the resulting label for the new example.
- Here the 1st and 2nd nearest neighbors have target label 1 and the 3rd nearest neighbor has target label 0. Target label 1 has the majority.
- Hence the new example is classified as 1, That is the diabetic patient has **Sugar**.

KNN EXAMPLE 2

Apply K Nearest Neighbor (K-NN) classifier to predict the class of sports with the given features

Name= Angelina, age =5, gender = 1

Class of sports = ?

And assume k=3.

NAME	AGE	GENDER	CLASS OF SPORTS
Ajay	32	0	Football
Mark	40	0	Neither
Sara	16	1	Cricket
Zaira	34	1	Cricket
Sachin	55	0	Neither
Rahul	40	0	Cricket
Pooja	20	1	Neither
Smith	15	0	Cricket
Laxmi	55	1	Football
Michael	15	0	Football

EXAMPLE 3

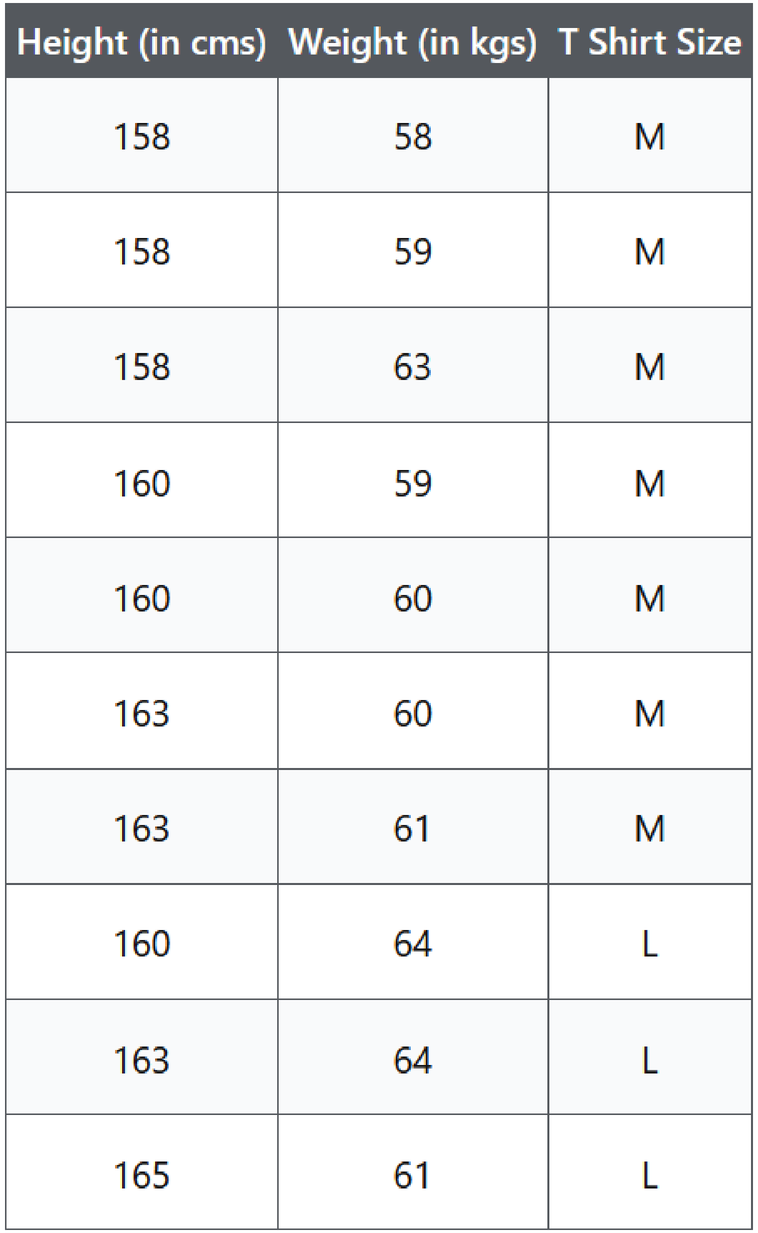
Suppose we have height, weight and T-shirt size of some customers and we need to predict the T-shirt size of a new customer given only height and weight information we have.

New customer named 'Monica' has height 161cm and weight 61kg.

Data including height, weight and T-shirt size information is shown below

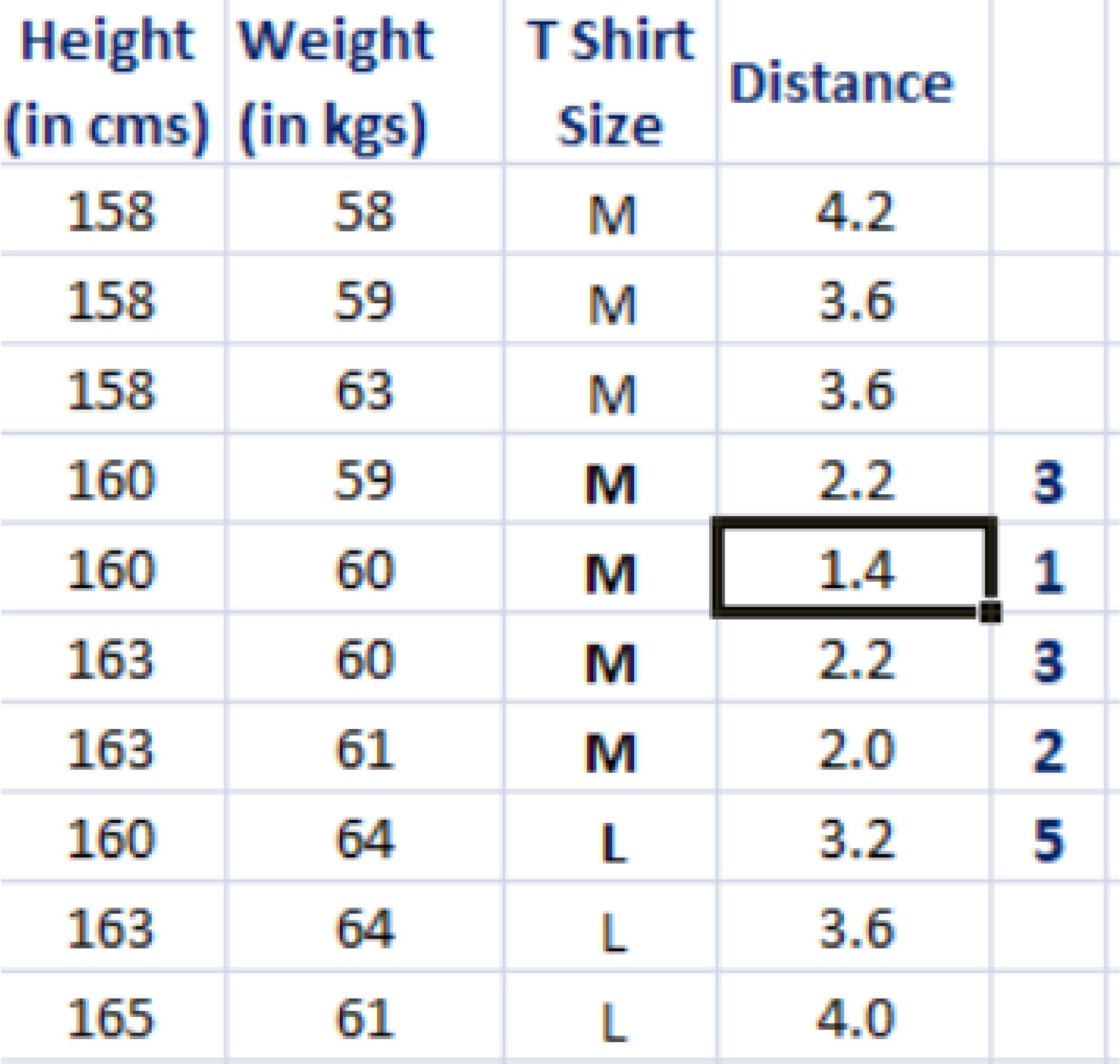
- Apply K Nearest Neighbor (K-NN) classifier to

Predict the T shirt size.



Height (in cms)	Weight (in kgs)	T Shirt Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L

Solution:



Height (in cms)	Weight (in kgs)	T Shirt Size	Distance	
158	58	M	4.2	
158	59	M	3.6	
158	63	M	3.6	
160	59	**M**	2.2	**3**
160	60	**M**	1.4	**1**
163	60	**M**	2.2	**3**
163	61	**M**	2.0	**2**
160	64	**L**	3.2	**5**
163	64	L	3.6	
165	61	L	4.0	

K-NEAREST NEIGHBORS

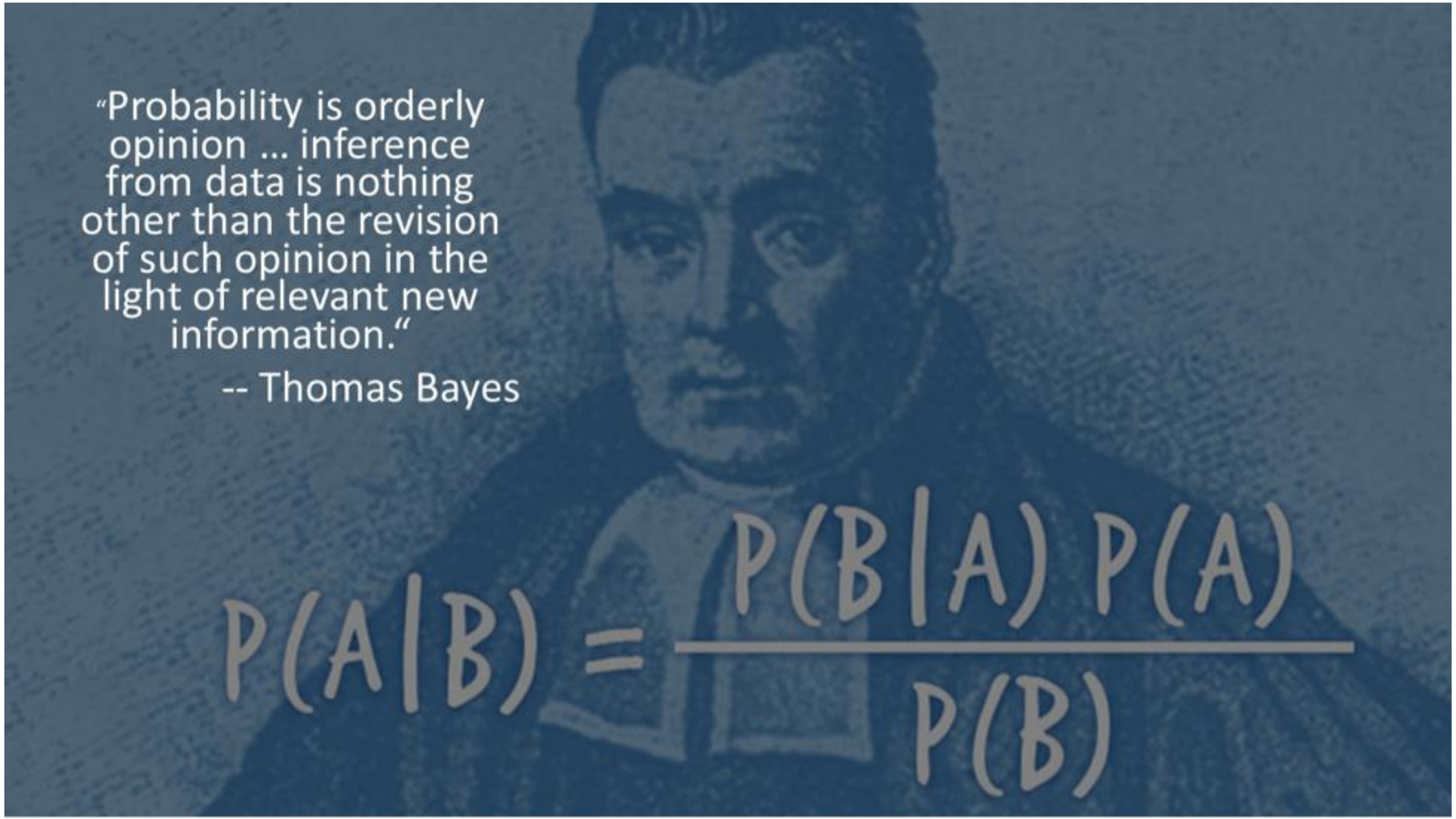
$$Euclidean\ distance:\ d(x,y) = \sqrt{\sum (x_i - yi)^2}$$

$$Squared\ Euclidean\ distance:\ d(x,y) = \sum (x_i - yi)^2$$

$$Manhattan\ distance:\ d(x,y) = \sum |(xi - yi)|$$

Lab program: **Write a program to implement Classification Algorithm. Calculate the accuracy, precision, recall.**

BAYES THEOREM

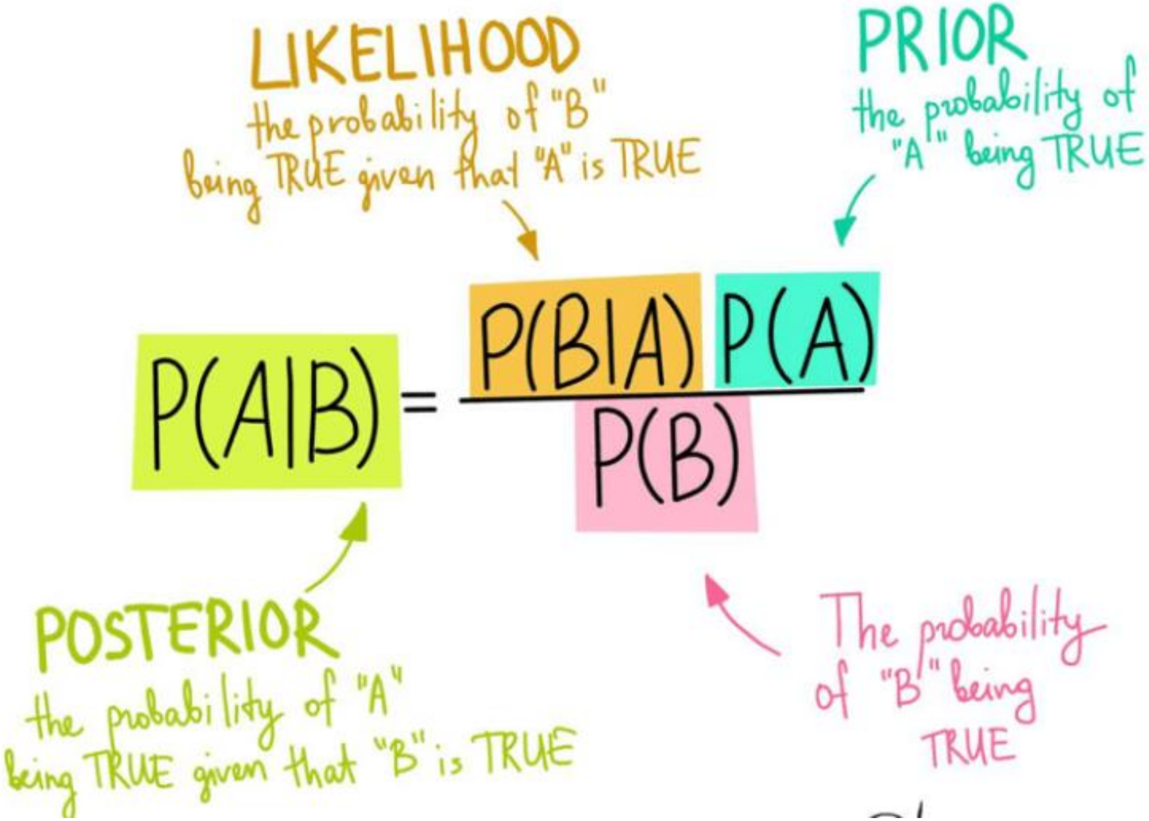


“Probability is orderly opinion … inference from data is nothing other than the revision of such opinion in the light of relevant new information.“
-- Thomas Bayes
$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$

Have you ever seen the popular TV show ‘Sherlock’ (or any crime thriller show)? Think about it – our beliefs about the culprit change throughout the episode. We process new evidence and refine our hypothesis at each step. This is Bayes’ Theorem in real life!

BAYES THEOREM

- P(A|B) = the **posterior** probability the probability of A to occur giv event B already occurred
- P(B|A) is the **likelihood** or the probability of B given A
- P(A), P(B) is the **prior** probability event A and B to occur



LIKELIHOOD
the probability of "B" being TRUE given that "A" is TRUE
PRIOR
the probability of "A" being TRUE
P(A|B) = P(B|A) P(A) / P(B)
POSTERIOR
the probability of "A" being TRUE given that "B" is TRUE
The probability of "B" being TRUE

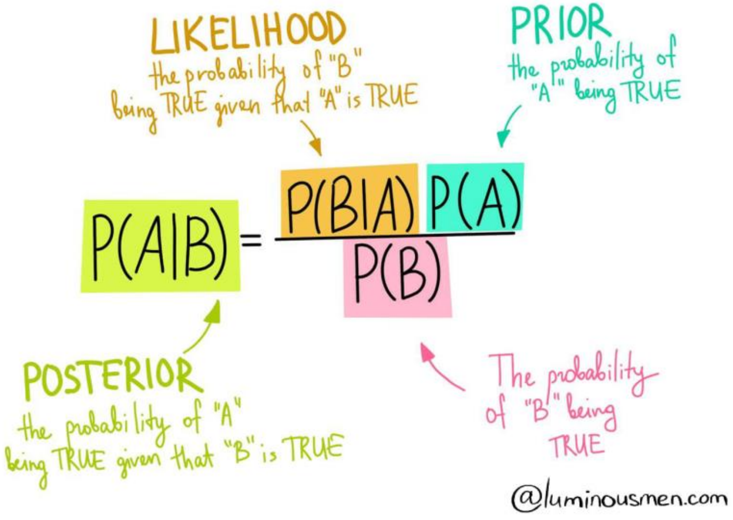
NAÏVE BAYE’S CLASSIFIER

- It is a classification technique based on Bayes’ Theorem with an independence assumption among predictors.
- Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identifying that it is an apple without depending on each other.
- Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

2) BAYESIAN CLASSIFICATION: WHY?

1. <u>A statistical classifier</u>: performs *probabilistic prediction, i.e.,* predicts class membership probabilities

2. <u>Foundation:</u> Based on Bayes’ Theorem.



LIKELIHOOD
the probability of "B" being TRUE given that "A" is TRUE
PRIOR
the probability of "A" being TRUE
$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$
POSTERIOR
the probability of "A" being TRUE given that "B" is TRUE
The probability of "B" being TRUE
@luminousmen.com

<u>3. Performance:</u> A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers

BAYESIAN CLASSIFICATION: WHY?

4. Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data.

5. Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

BAYES' THEOREM: BASICS

1. Total probability Theorem:

$$P(B) = \sum_{i=1}^{M} P(B|A_i)P(A_i)$$

2. Bayes' Theorem:

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

- Let **X** be a data sample (“*evidence*”): class label is unknown
- Let H be a *hypothesis* that X belongs to class C

BAYES' THEOREM: BASICS

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P($$

- Classification is to determine P(H|X), (i.e., posteriori probability): the probability that the hypothesis holds given the observed data sample X.
- Ex: X = is a 35-year-old customer with an income of $40,000.
- Suppose that H = the hypothesis that our customer will buy a computer.
- Then P(H|X) reflects the probability that customer X will buy a computer given that we know the customer's age and income.
- P(H) (prior probability): the initial probability
- E.g., X will buy computer, regardless of age, income,
- P(X): probability that sample data is observed
- Ex: it is the probability that a person from our set of customers is 35 years old and earns $40,000.
- P(X|H) (likelihood): the probability of observing the sample X, given that the hypothesis holds
- E.g., it is the probability that a customer, X, is 35 years old and earns $40,000, given that we know the customer will buy a computer

NAIVE BAYES CLASSIFIERS ALGORITHM

Step 1 – Collect raw data
Step **2 – Convert data to a frequency table(s)**
Step 3 – Calculate prior probability and evidence
Step 4 – Apply probabilities to Bayes’ Theorem equation

NAIVE BAYES CLASSIFIERS

Problem: If the Outlook is sunny, then the Player should play or not?

Apply Bayes'theorem:

1. P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)
2. P(No|Sunny)= P(Sunny|No)*P(No)

Step1: first consider the below dataset

Total observations - 14

Sunny – total - 5

Outlook	**Play**
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Sunny	No
Rainy	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Sunny	No
Sunny	Yes
Rainy	No
Overcast	Yes
Overcast	Yes

NAIVE BAYES CLASSIFIERS

Step 2: Frequency table:

Outlook	No	Yes	
Overcast	0	5	
Rainy	2	2	
Sunny	2	3	
Total	**4**	**10**	**= 14**

Outlook	Play
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Sunny	No
Rainy	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Sunny	No
Sunny	Yes
Rainy	No
Overcast	Yes
Overcast	Yes

NAIVE BAYES CLASSIFIERS

Step 3: (Prior prob/Likelihood table):

Outlook	No	Yes
Overcast	0	5
Rainy	2	2
Sunny	2	3

P(No)= 4/14=0.29 **P(Yes)=10/14=0.71**

P(Sunny|NO)= 2/4=0.5 **P(Sunny|Yes)= 3/10= 0.3**

Outlook	**Play**
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Sunny	No
Rainy	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Sunny	No
Sunny	Yes
Rainy	No
Overcast	Yes
Overcast	Yes

NAIVE BAYES CLASSIFIERS

Step 4: Applying Bayes'theorem:

1. P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)
2. P(No|Sunny)= P(Sunny|No)*P(No)

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)

P(Sunny|Yes)= 3/10= 0.3

P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71 = 0.213

P(No|Sunny)= P(Sunny|No)*P(No)

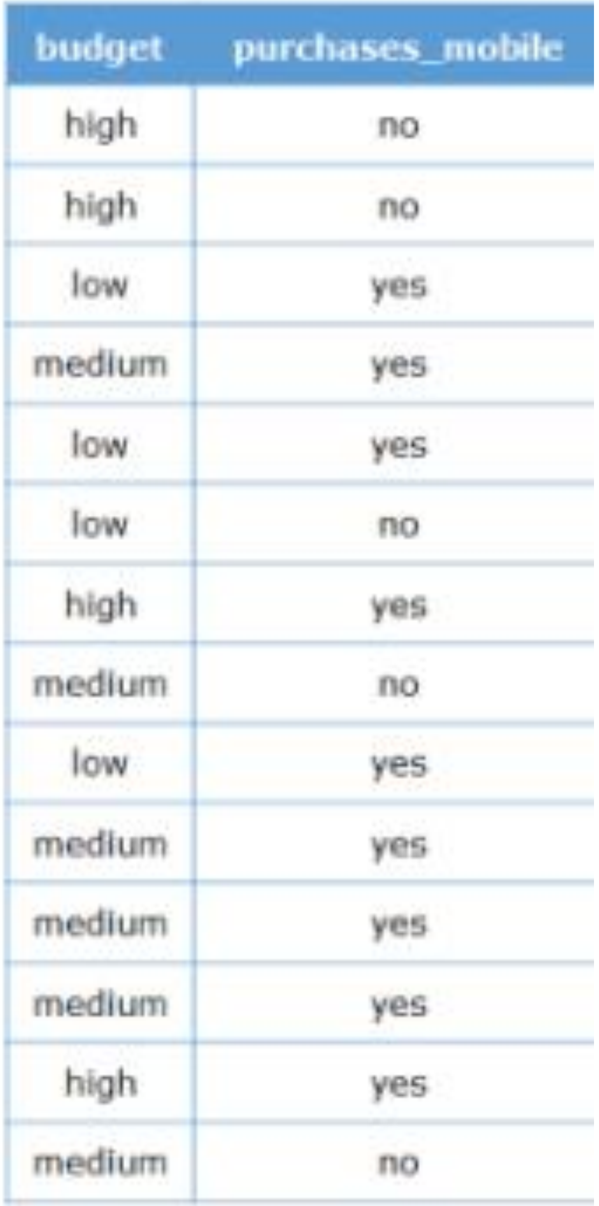
P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

So P(No|Sunny)= 0.5*0.29 = 0.145

Outlook	Play
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Sunny	No
Rainy	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Sunny	No
Sunny	Yes
Rainy	No
Overcast	Yes
Overcast	Yes

NAIVE BAYES CLASSIFIERS – EXE 1



budget	purchases_mobile
high	no
high	no
low	yes
medium	yes
low	yes
low	no
high	yes
medium	no
low	yes
medium	yes
medium	yes
medium	yes
high	yes
medium	no

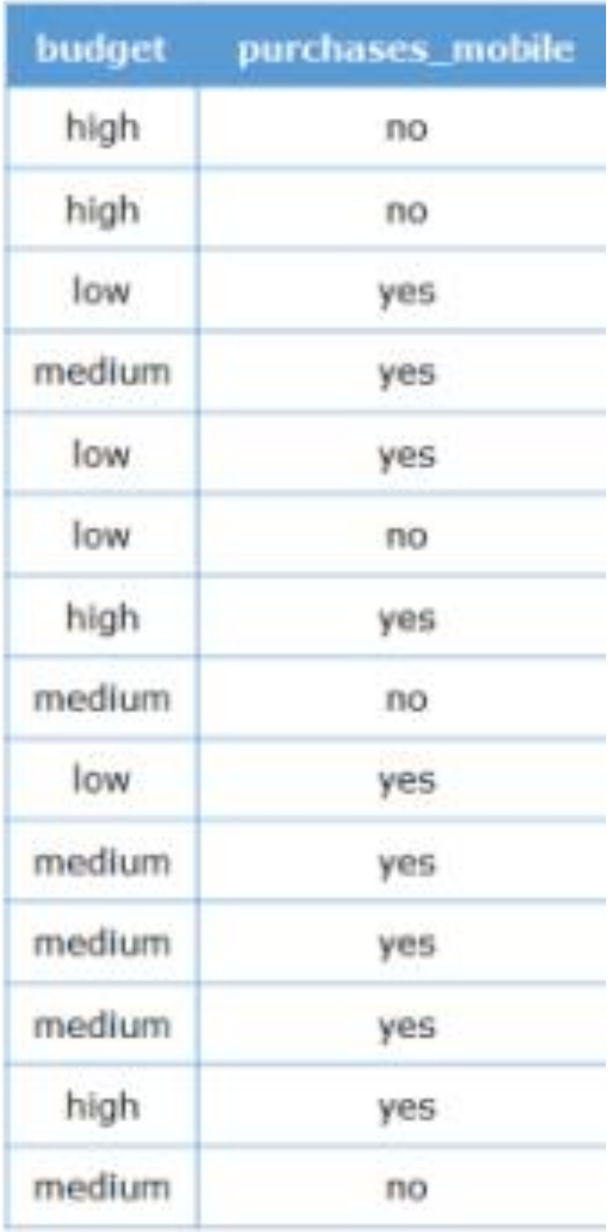
For the given test data {budget = medium}, predict the chance of purchasing mobile.

NAIVE BAYES CLASSIFIERS

Problem: For the given test data {budget = medium}, predict the chance of purchasing mobile.

Step1: first consider the below dataset

Total observations - 14
Medium – total - 6

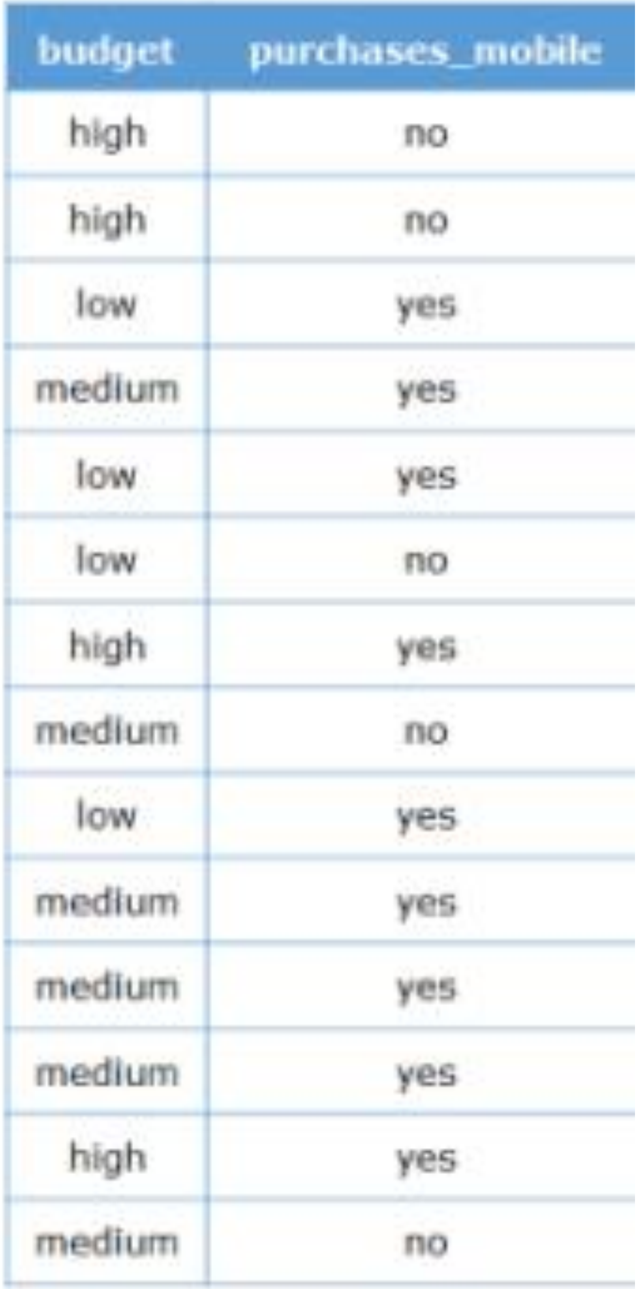


budget	purchases_mobile
high	no
high	no
low	yes
medium	yes
low	yes
low	no
high	yes
medium	no
low	yes
medium	yes
medium	yes
medium	yes
high	yes
medium	no

NAIVE BAYES CLASSIFIERS

Step 2: Frequency table:

Budget	**No**	**Yes**	
Low			
Medium			
High			
Total			**= 14**



budget	purchases_mobile
high	no
high	no
low	yes
medium	yes
low	yes
low	no
high	yes
medium	no
low	yes
medium	yes
medium	yes
medium	yes
high	yes
medium	no

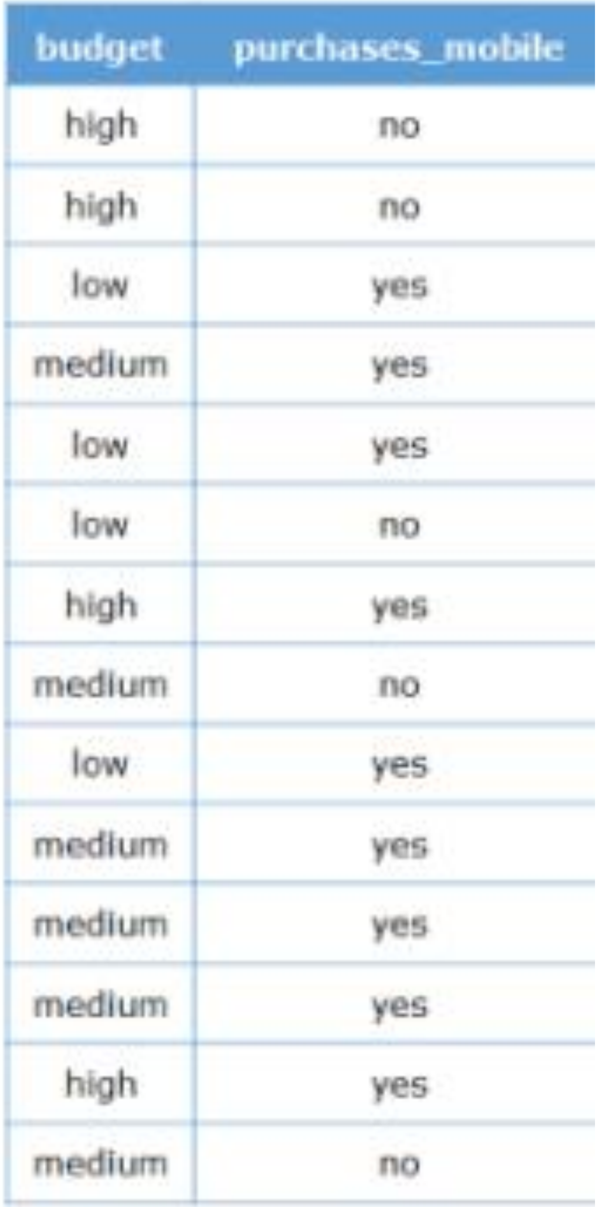
NAIVE BAYES CLASSIFIERS

Step 3: (prior prob/Likelihood table):

	No	**Yes**
Low		
Medium		
High		

P(No)= /14= **P(Yes)= /14=**

P(Medium|NO)= /= **P(Medium/Yes)= / =**



budget	purchases_mobile
high	no
high	no
low	yes
medium	yes
low	yes
low	no
high	yes
medium	no
low	yes
medium	yes
medium	yes
medium	yes
high	yes
medium	no

NAIVE BAYES CLASSIFIERS

Step 4: Applying Bayes'theorem: - no denominator

1. P(Yes|Medium)= P(Medium|Yes)*P(Yes)
2. P(No|Medium)= P(Medium|No)*P(No)

P(Medium|Yes)=

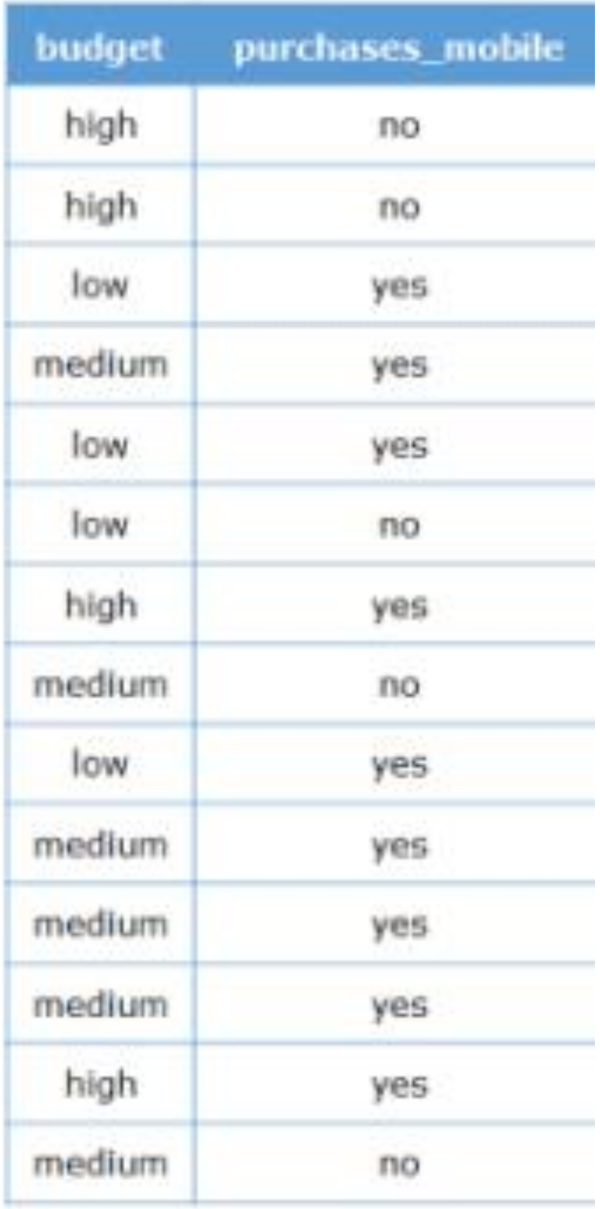
P(Yes)=

So P(Yes|Medium) **=**

P(Medium|No)=

P(No)=

So P(No|Medium**)=**

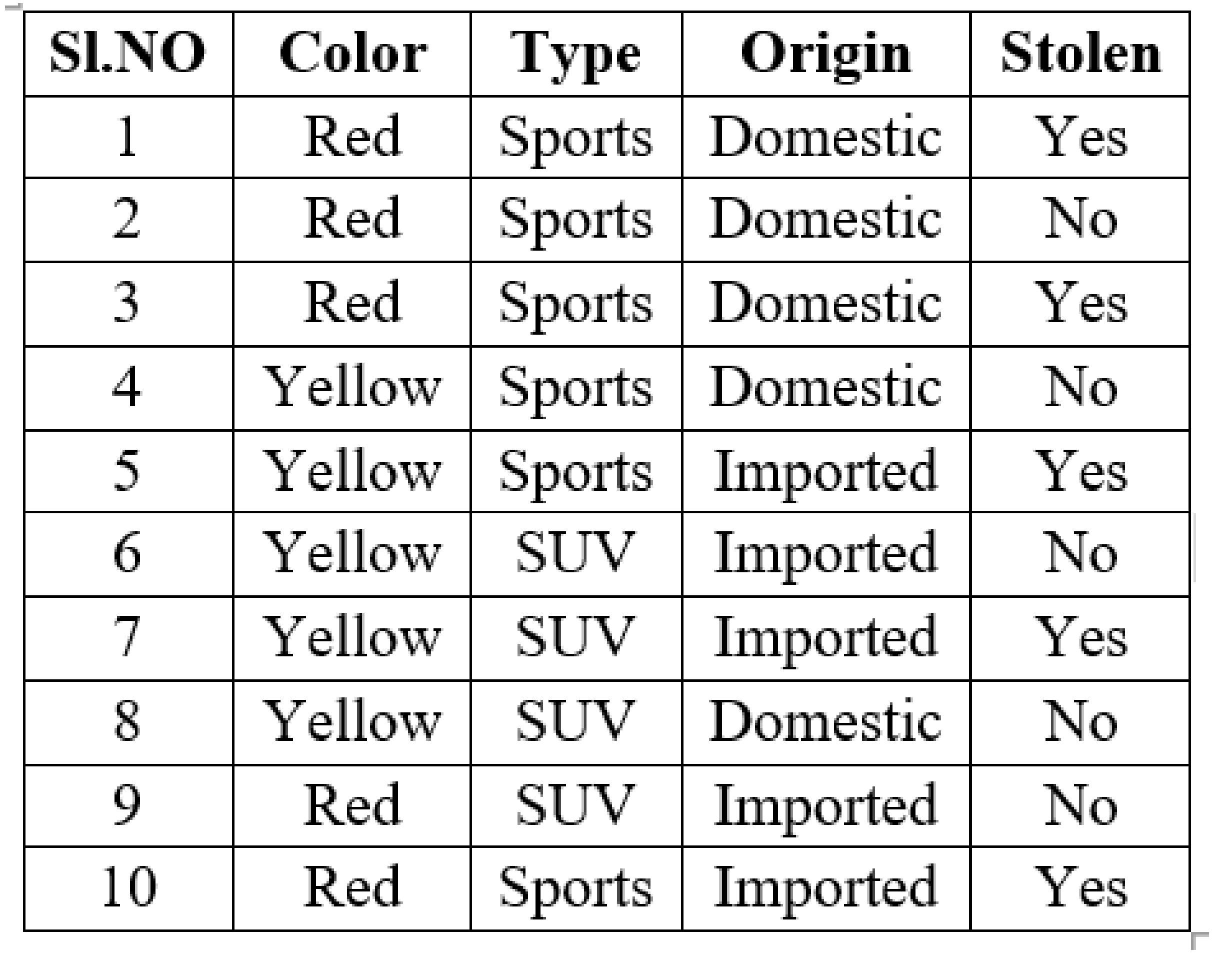


budget	purchases_mobile
high	no
high	no
low	yes
medium	yes
low	yes
low	no
high	yes
medium	no
low	yes
medium	yes
medium	yes
medium	yes
high	yes
medium	no

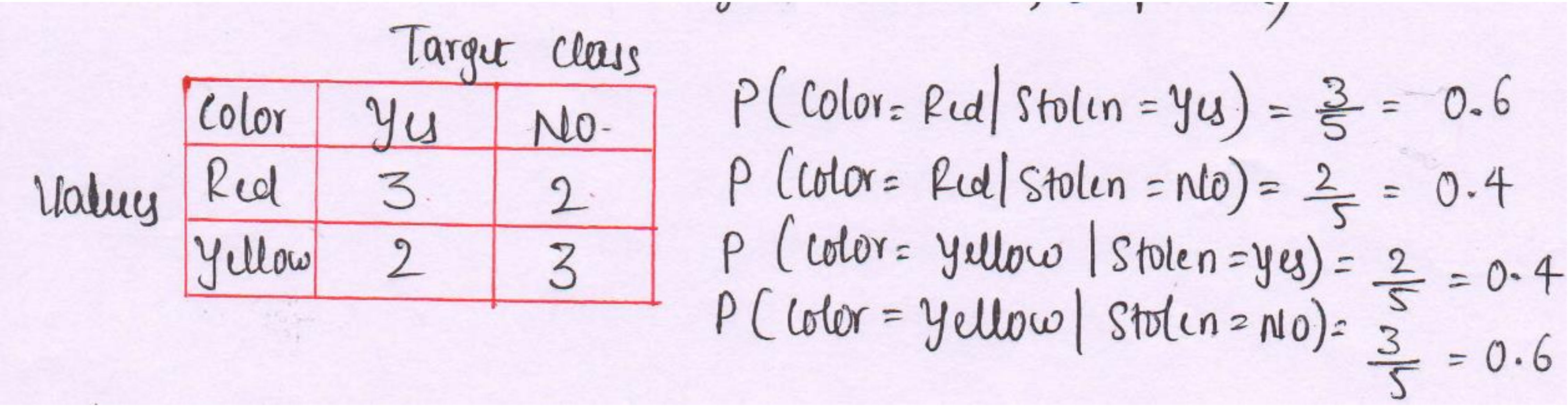
EXAMPLE 3

Apply the naïve Bayes classifier for the following dataset, and consider the new instance.

New Instance=(Red, SUV, Domestic)



Sl.NO	Color	Type	Origin	Stolen
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes



Target class

Values

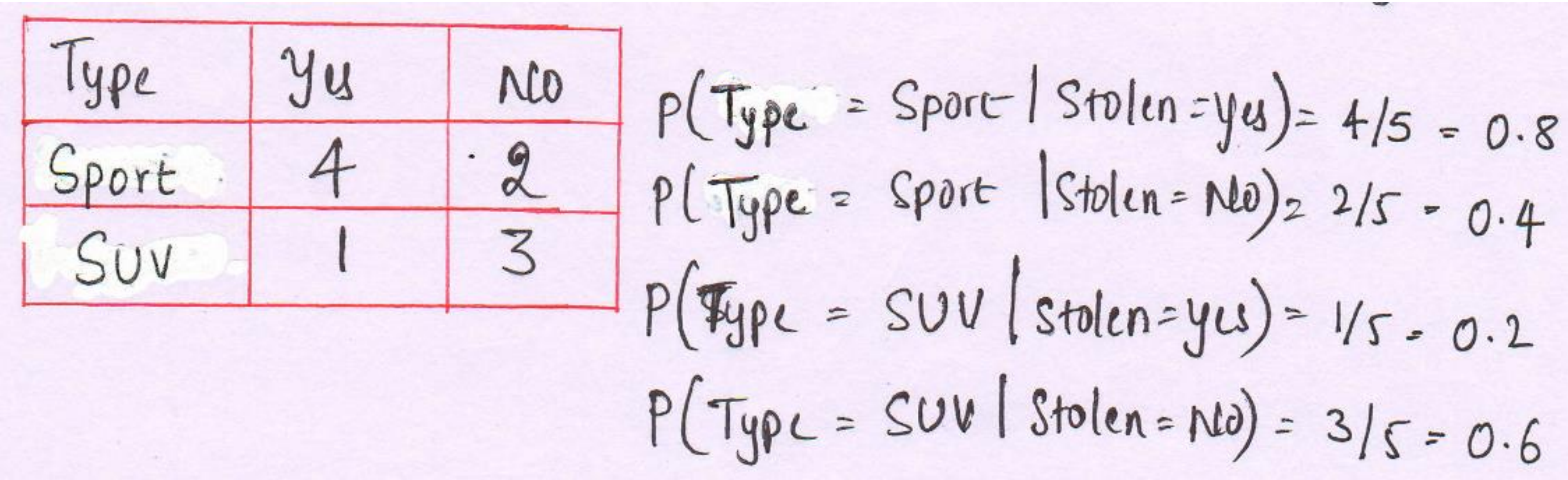
Color	Yes	No
Red	3	2
Yellow	2	3

$P(\text{Color}=\text{Red} \mid \text{Stolen}=\text{Yes}) = \frac{3}{5} = 0.6$

$P(\text{Color}=\text{Red} \mid \text{Stolen}=\text{No}) = \frac{2}{5} = 0.4$

$P(\text{Color}=\text{Yellow} \mid \text{Stolen}=\text{Yes}) = \frac{2}{5} = 0.4$

$P(\text{Color}=\text{Yellow} \mid \text{Stolen}=\text{No}) = \frac{3}{5} = 0.6$



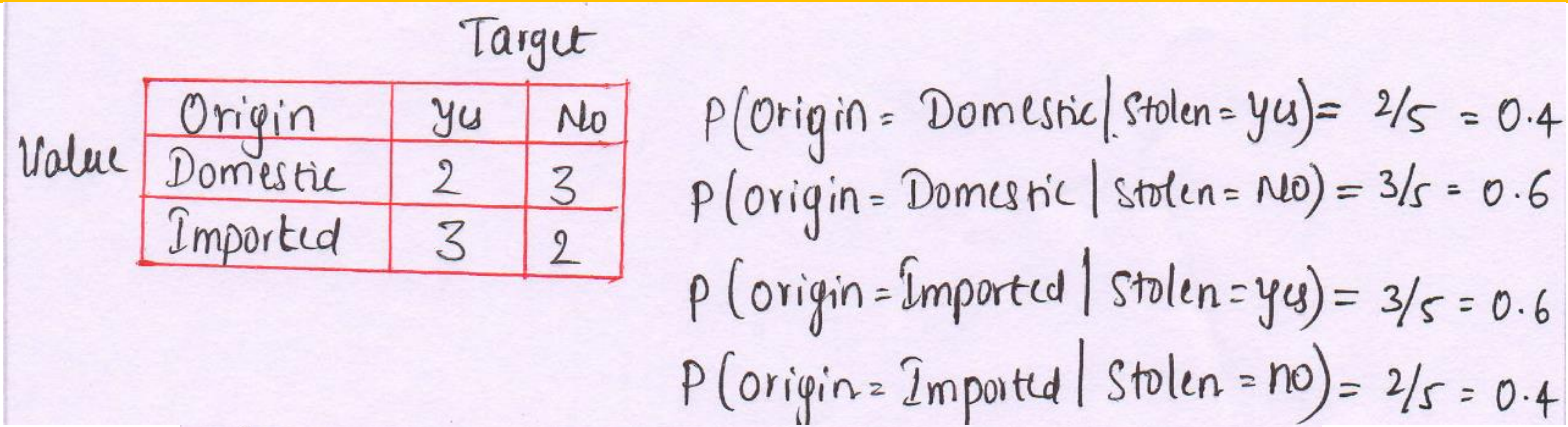
Type	Yes	No
Sport	4	2
SUV	1	3

$P(\text{Type}=\text{Sport} \mid \text{Stolen}=\text{Yes}) = 4/5 = 0.8$

$P(\text{Type}=\text{Sport} \mid \text{Stolen}=\text{No}) = 2/5 = 0.4$

$P(\text{Type}=\text{SUV} \mid \text{Stolen}=\text{Yes}) = 1/5 = 0.2$

$P(\text{Type}=\text{SUV} \mid \text{Stolen}=\text{No}) = 3/5 = 0.6$



Target

Value

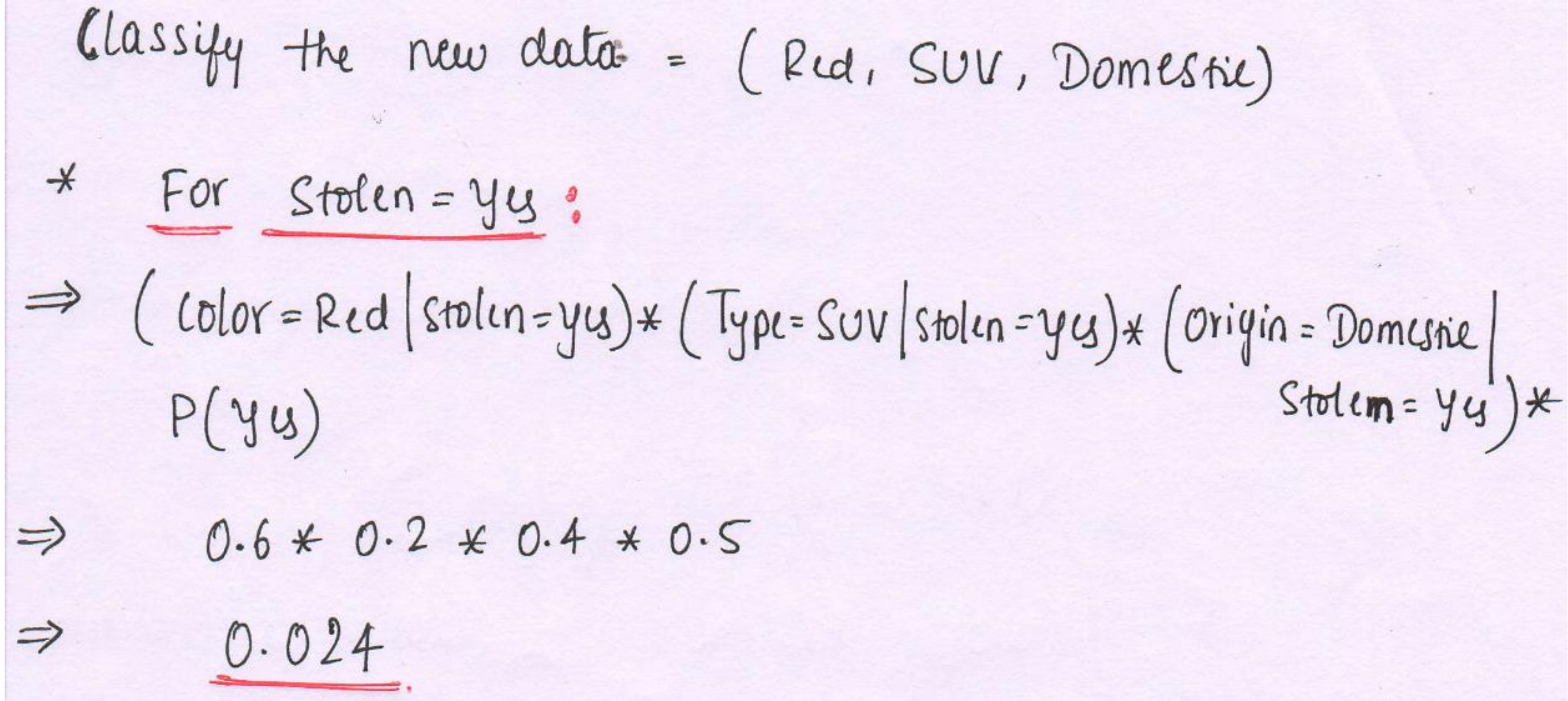
Origin	yes	No
Domestic	2	3
Imported	3	2

$P(\text{Origin} = \text{Domestic} \mid \text{Stolen} = \text{yes}) = 2/5 = 0.4$

$P(\text{origin} = \text{Domestic} \mid \text{stolen} = \text{NO}) = 3/5 = 0.6$

$P(\text{origin} = \text{Imported} \mid \text{stolen} = \text{yes}) = 3/5 = 0.6$

$P(\text{origin} = \text{Imported} \mid \text{Stolen} = \text{no}) = 2/5 = 0.4$



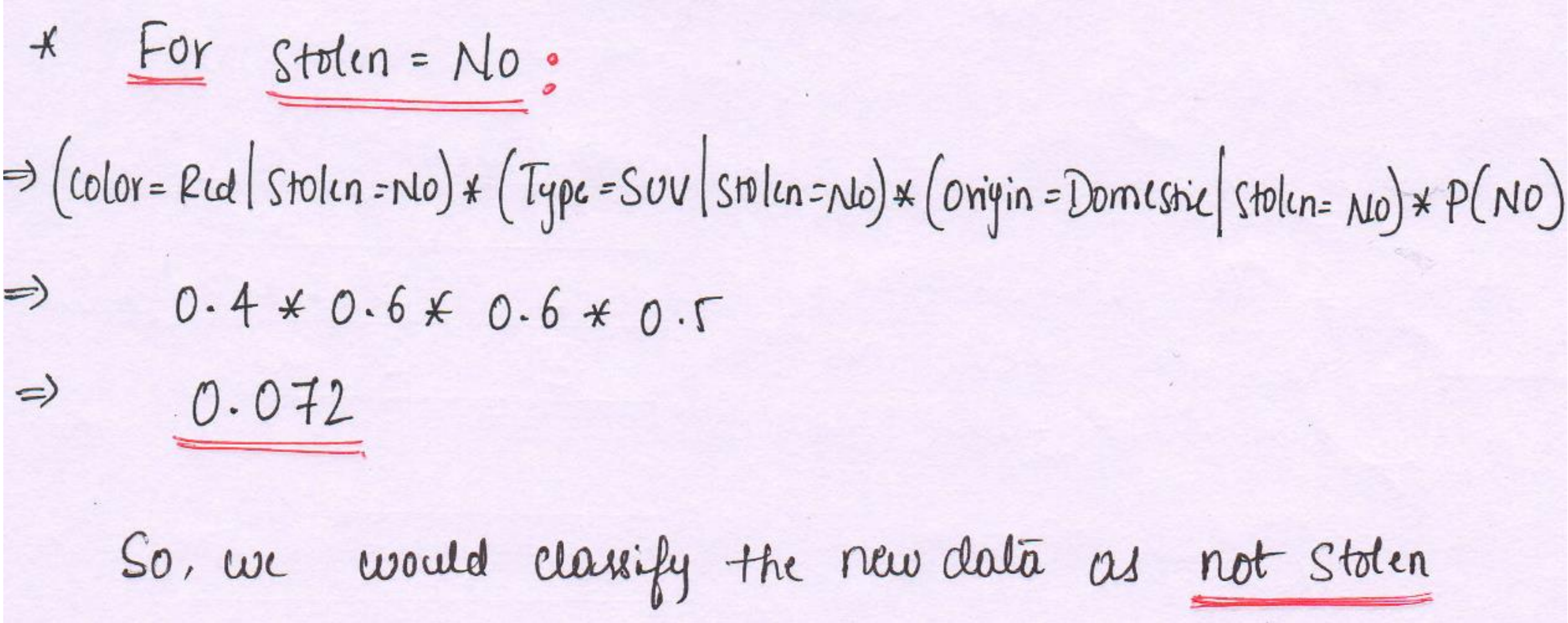
Classify the new data = (Red, SUV, Domestic)

* For Stolen = yes :

$\Rightarrow$ (Color = Red | stolen = yes) * (Type = SUV | stolen = yes) * (origin = Domestic | Stolem = yes) * P(yes)

$\Rightarrow$ $0.6 * 0.2 * 0.4 * 0.5$

$\Rightarrow$ 0.024



* For Stolen = No :

⇒ (Color = Red | Stolen = No) * (Type = SUV | Stolen = No) * (Origin = Domestic | Stolen = No) * P(No)

⇒ 0.4 * 0.6 * 0.6 * 0.5

⇒ 0.072

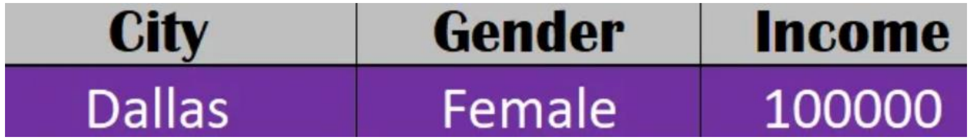
So, we would classify the new data as not Stolen

EXAMPLE 4

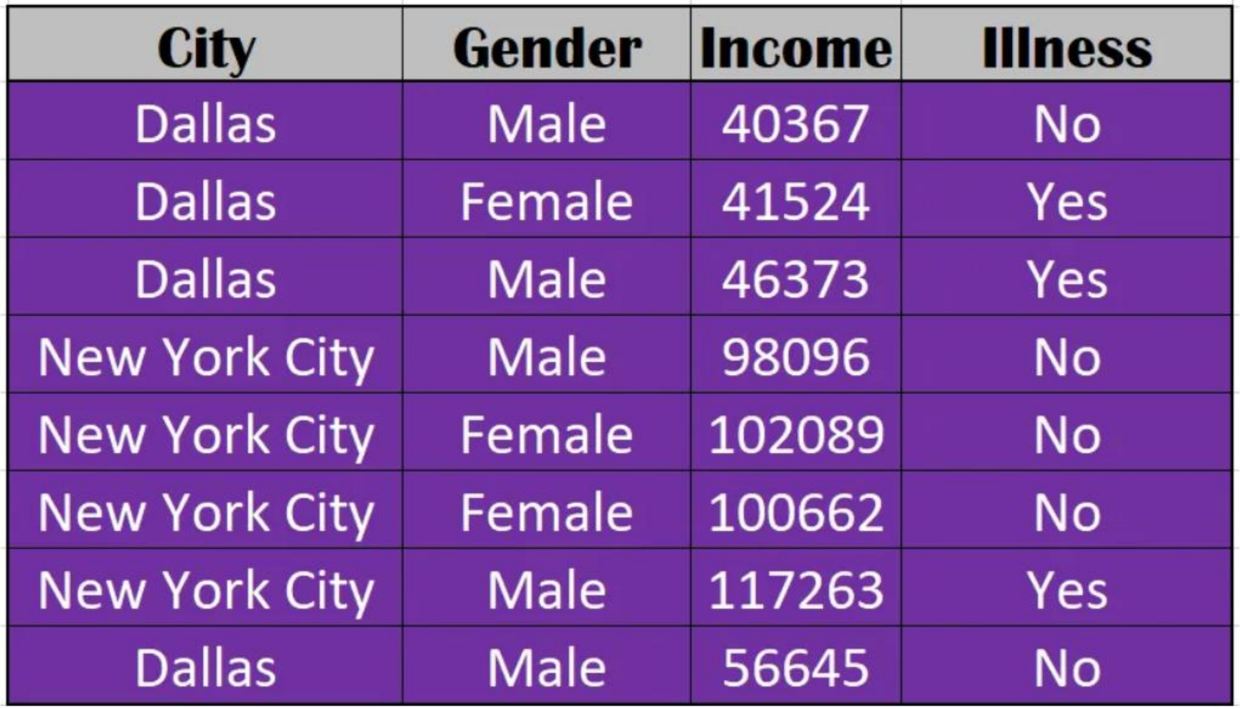
Consider the following dataset

Using Naïve bayes classifier classify

the below new sample



City	Gender	Income
Dallas	Female	100000



City	Gender	Income	Illness
Dallas	Male	40367	No
Dallas	Female	41524	Yes
Dallas	Male	46373	Yes
New York City	Male	98096	No
New York City	Female	102089	No
New York City	Female	100662	No
New York City	Male	117263	Yes
Dallas	Male	56645	No

Solution: **patient will not have a heart disease .**

NAÏVE BAYES CLASSIFIER: TRAINING DATASET

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

Class:

C1:buys_computer = ‘yes’

C2:buys_computer = ‘no’

Data to be classified:

X = (age <=30,

Income = medium,

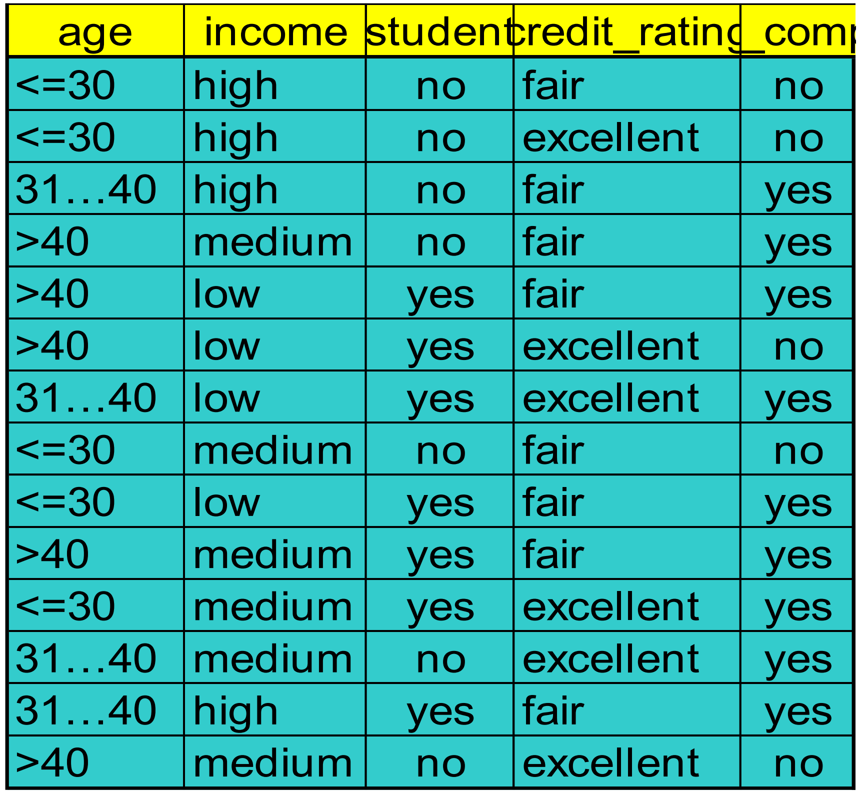
Student = yes

Credit_rating = Fair)

We need to maximize P(X|Ci)P(Ci), for i = 1, 2.

age	income	student	credit_rating	_comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

NAÏVE BAYES CLASSIFIER: AN EXAMPLE



age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$P(C_i)$: P(buys_computer = “yes”) = 9/14 = 0.643

P(buys_computer = “no”) = 5/14= 0.357

Compute $P(X|C_i)$ for each class

P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222

P(age = “<= 30” | buys_computer = “no”) = 3/5 = 0.6

P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444

P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4

P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667

P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2

P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667

P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

NAÏVE BAYES CLASSIFIER: AN EXAMPLE

age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

3. **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.044 *0.643= 0.028

P(X|buys_computer = "no") * P(buys_computer = "no") = 0.019* 0.357 =0.007

Therefore, X belongs to class ("buys_computer = yes")

Example 2:

Data to be classified:

X = (age>40,

Income = low,

Student = yes

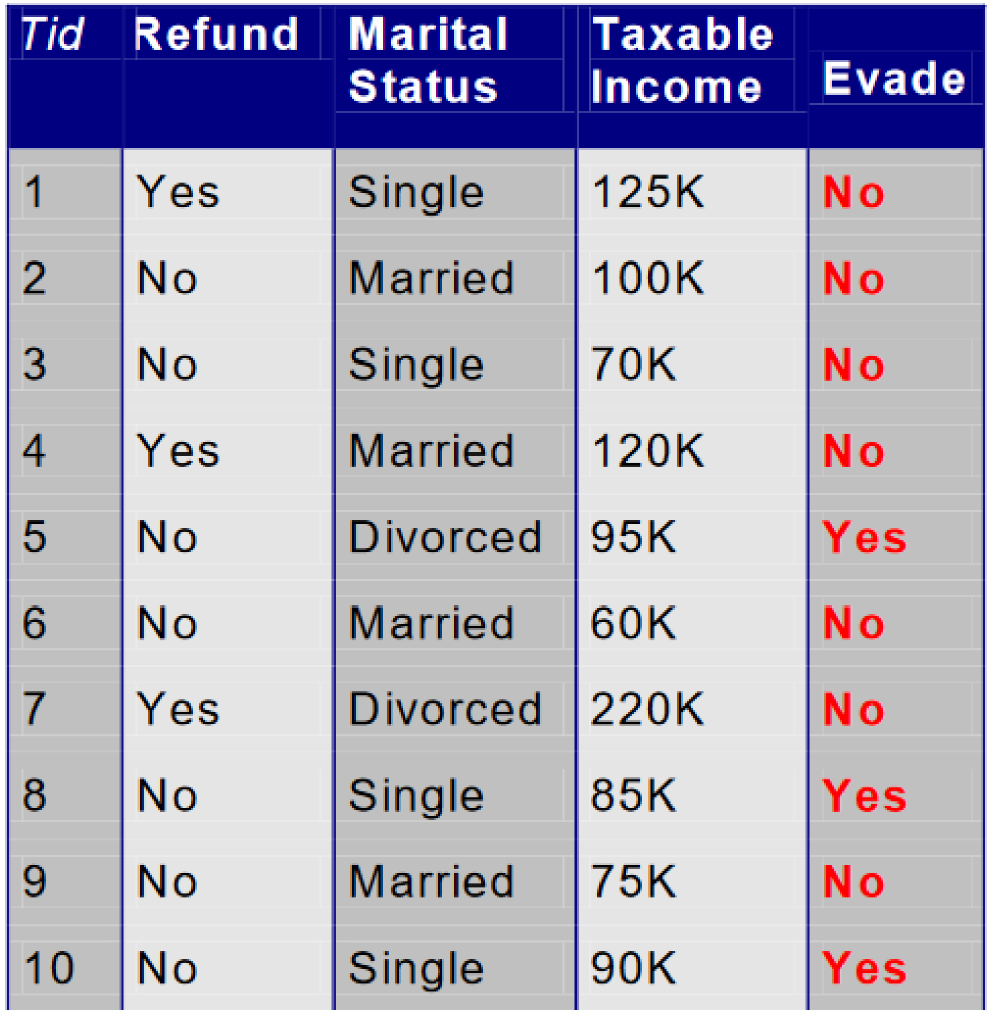
Credit_rating = Fair)

EXAMPLE 5

Consider the following dataset.

Classify the given test tuple using naïve Bayes classifier

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$



Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

EXAMPLE 5

Solution:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No) × P(Married| Class=No) × P(Income=120K| Class=No) = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes) × P(Married| Class=Yes) × P(Income=120K| Class=Yes) = $1 \times 0 \times 1.2 \times 10^{-9} = 0$

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)
=> Class = No

NAIVE BAYES CLASSIFIERS – EXE 2

predict the type of fruit from the observation given below for the sample X= {Yellow, Sweet, Long}

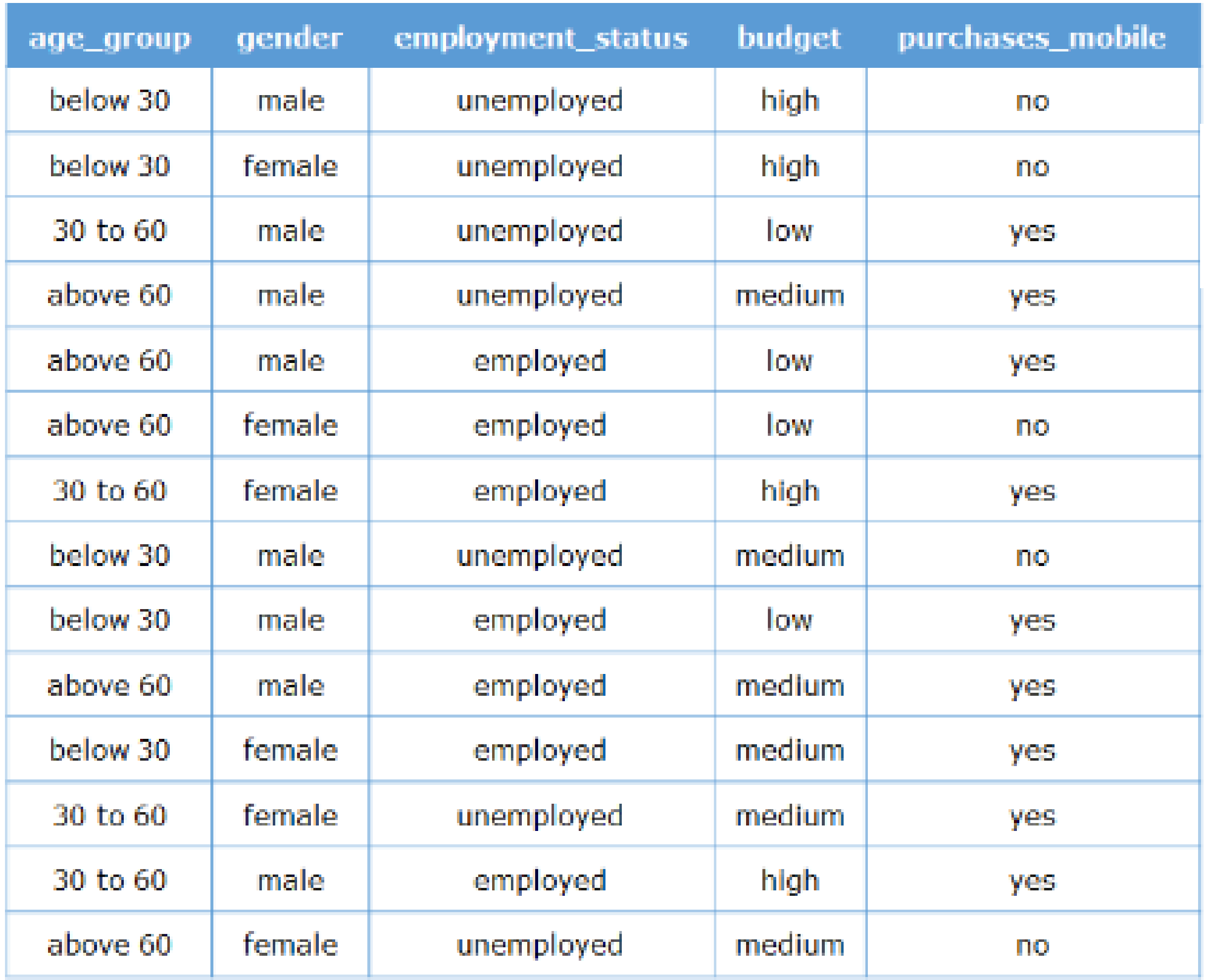
Fruit	Yellow	Sweet	Long	Total
Mango	**350**	450	0	650
Banana	400	300	350	400
Others	50	100	50	150
Total	800	850	400	1200

NAIVE BAYES CLASSIFIERS – EXE 3

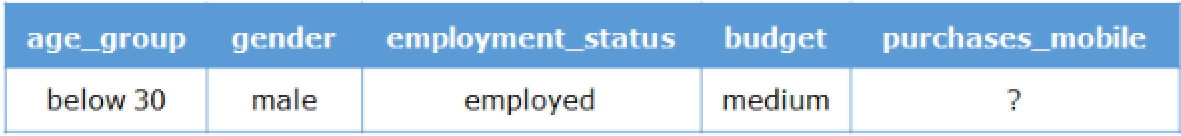
predict the type of fruit from the observation given below for the sample X= {Yellow, Sweet, Long}

Fruit	Yellow	Sweet	Long	Total
Banana	450	350	400	500
Orange	300	150	0	300
Others	50	150	100	200
Total	800	650	500	1000

NAIVE BAYES CLASSIFIERS – EXE 4



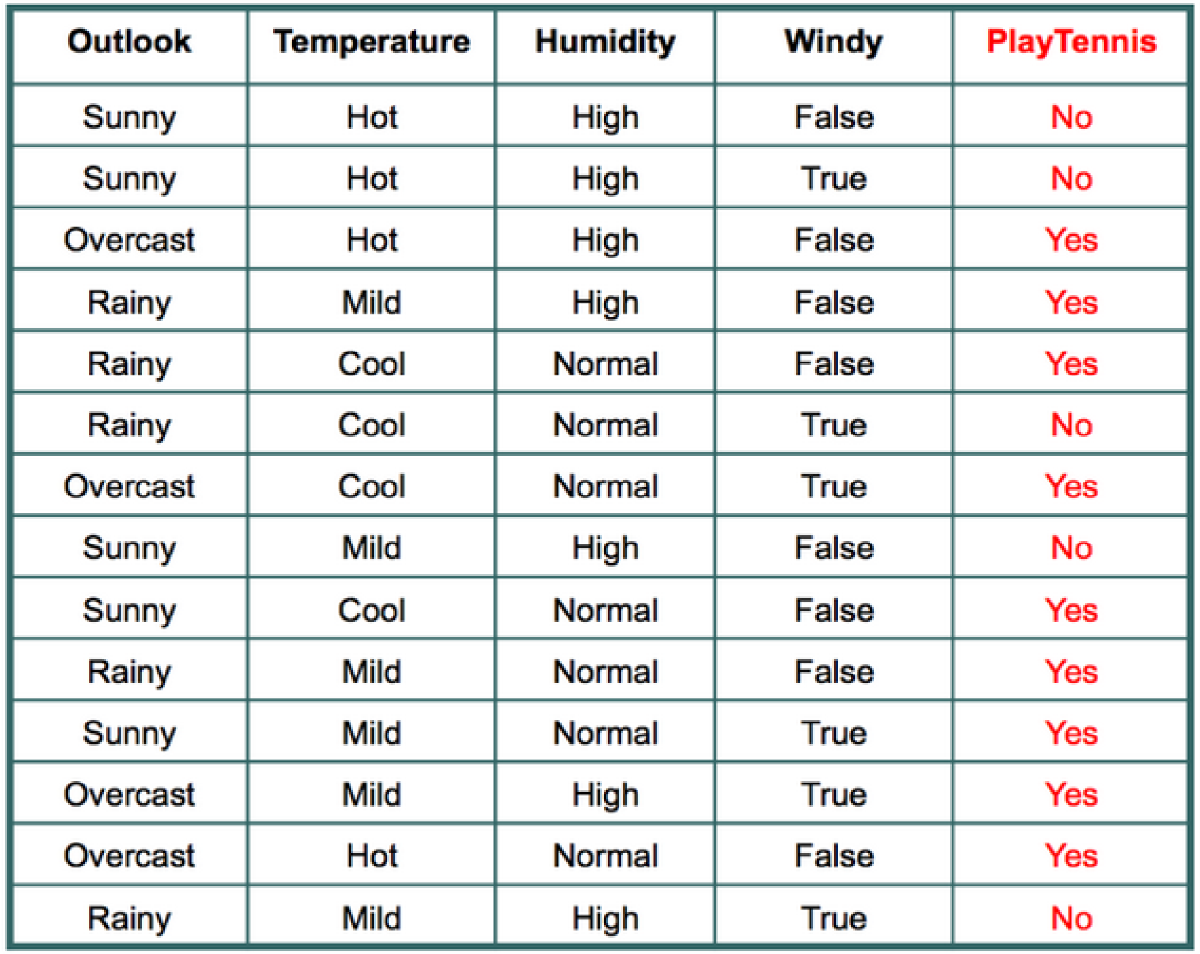
age_group	gender	employment_status	budget	purchases_mobile
below 30	male	unemployed	high	no
below 30	female	unemployed	high	no
30 to 60	male	unemployed	low	yes
above 60	male	unemployed	medium	yes
above 60	male	employed	low	yes
above 60	female	employed	low	no
30 to 60	female	employed	high	yes
below 30	male	unemployed	medium	no
below 30	male	employed	low	yes
above 60	male	employed	medium	yes
below 30	female	employed	medium	yes
30 to 60	female	unemployed	medium	yes
30 to 60	male	employed	high	yes
above 60	female	unemployed	medium	no



age_group	gender	employment_status	budget	purchases_mobile
below 30	male	employed	medium	?

NAIVE BAYES CLASSIFIERS – EXE 5

Predict whether the player can play based on the observation given below X= {Outlook=Sunny, Temp=Cool, Humdiity=High, Windy=True}



Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

NAIVE BAYES CLASSIFIERS

Advantages of Naïve Bayes Classifier:

- Easy to work with when using binary or categorical input values and handles both continuous and discrete data
- Fast and reliable for making real-time predictions.
- can be used for Binary as well as Multi-class Classifications and performs well in Multi-class predictions as compared to the other Algorithms.
- work very well with high-dimensional sparse data and are relatively robust to the parameters.
- the most popular choice for text classification problems.
- Require a small number of training data for estimating the parameters necessary for classification.

NAIVE BAYES CLASSIFIERS

Disadvantages of Naïve Bayes Classifier:

- NB models are great baseline models and are often used on very large datasets, where training even a linear model might take too long.
- Assumes that all the features are independent, which is highly unlikely in practical scenarios.
- Unsuitable for numerical data.
- The number of features must be equal to the number of attributes in the data for the algorithm to make correct predictions.
- Encounters ‘Zero Frequency’ problem if a categorical variable has a category in the test dataset that wasn’t included in the training dataset, the model will assign it a 0 probability and will be unable to make a prediction.
- Computationally expensive when used to classify a large number of items.

NAIVE BAYES CLASSIFIERS

Applications of Naïve Bayes Classifier:

- Credit Scoring
- medical data classification
- Text classification such as Spam filtering : excellent choice for spam filtering of your emails or news categorization on your smartphone.
- Sentiment analysis: identification of positive or negative sentiments of a target group (customers, audience, etc.), feedback forms and reviews.
- Recommendation Systems: Naïve Bayes is used with collaborative filtering to build recommendation systems for you. ‘*Because you watched* ____’ section on Netflix.

NAIVE BAYES CLASSIFIERS

Bernoulli Naïve Bayes

- Predictors are Boolean/Binary variables
- Used when data is as per a multivariate Bernoulli distribution
- Popular for discrete features
- used for sparse count data such as text.
- Used for text/document classification

Multinomial Naïve Bayes

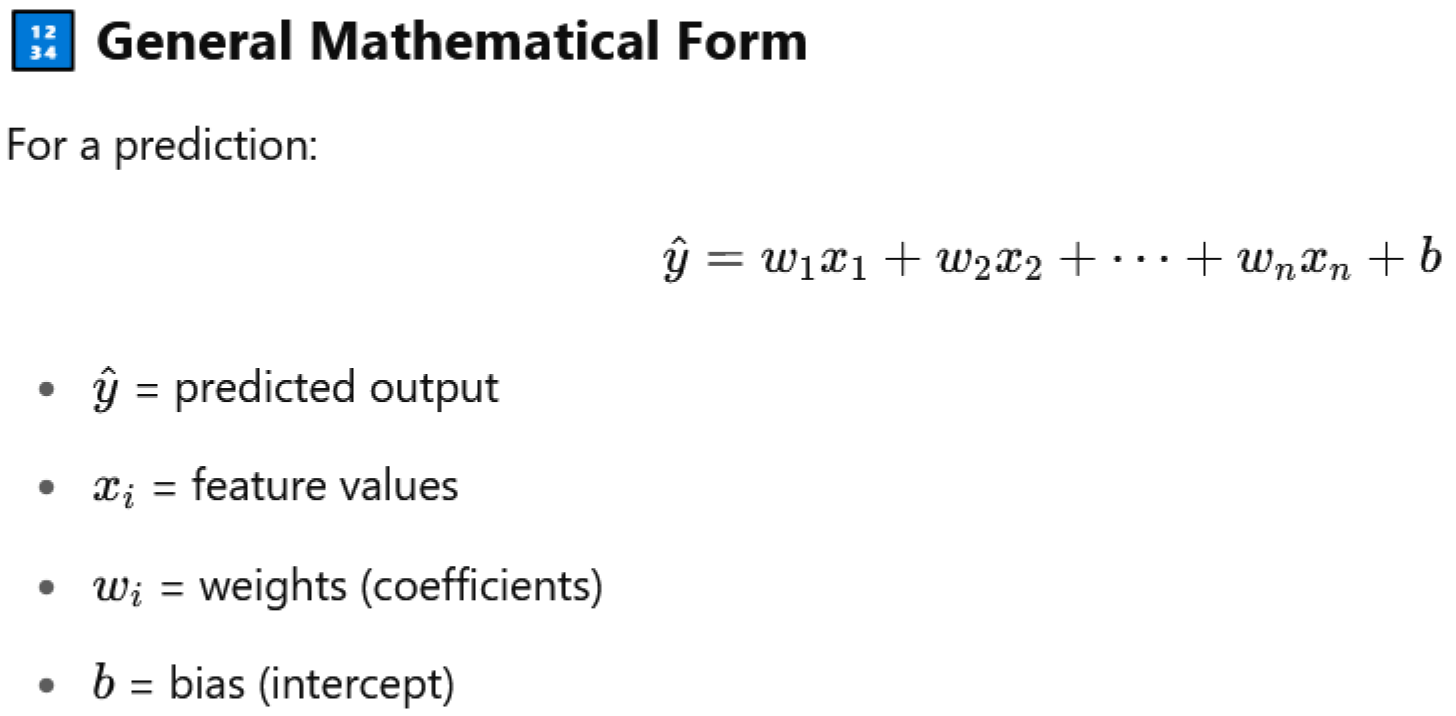
- Uses the frequency of present words as features
- Commonly used for document classification problems
- Popular for discrete features as well
- used for sparse count data such as text.
- performs better than BinaryNB, particularly on datasets with a relatively large number of nonzero features.
- Used for text/document classification

Gaussian Naïve Bayes

- Used when data is as per the Gaussian distribution
- Predictors are continuous variables
- used on very high-dimensional data

LINEAR MODELS

Linear models are machine learning algorithms that make predictions by computing a **linear combination of input features**. They assume a **linear relationship** between the input variables (features) and the output (target).



🔢 General Mathematical Form

For a prediction:

$$\hat{y} = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

- $\hat{y}$ = predicted output
- x_i = feature values
- w_i = weights (coefficients)
- b = bias (intercept)

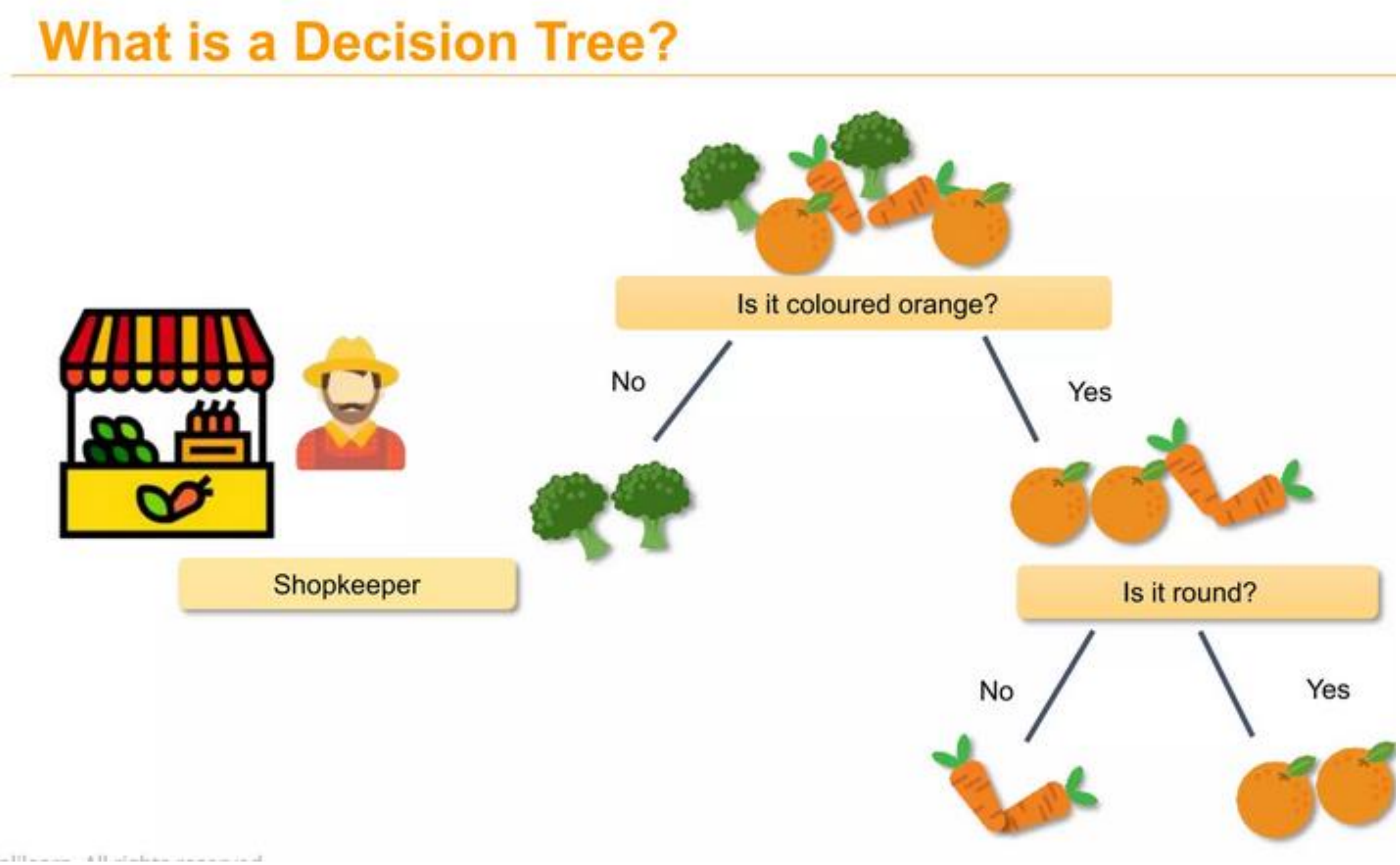
TYPES OF LINEAR MODELS

Model Name	Type	Use Case	Output Type
Linear Regression	Regression	Predict continuous value	Real number
Ridge Regression	Regression	Linear regression with L2 regularization	Real number
Lasso Regression	Regression	Linear regression with L1 regularization	Real number
Logistic Regression	Classification	Predict class labels	Probabilities / Classes
Linear Discriminant Analysis (LDA)	Classification	Linear decision boundaries	Classes
Support Vector Machine (Linear Kernel)	Classification	Linear separating hyperplane	Classes

When to Use Linear Models?

- When **interpretability** is important.
- When the data has a **linear trend**.
- When the dataset is **large** and needs **fast training**.
- As a **baseline** model before trying more complex ones.

DECISION TREE BASED ALGORITHM



What is a Decision Tree?

What problems can be solved using Decision Tree?

Classification:

Identifying to which set an object belongs

Example: Carrot is orange while broccoli is green

Regression:

Regression problems have continuous or numerical valued output variables

Example: Predicting the profits of a company

DECISION TREE

A Supervised Machine Learning Algorithm, used to build classification and regression models in the form of a tree structure.

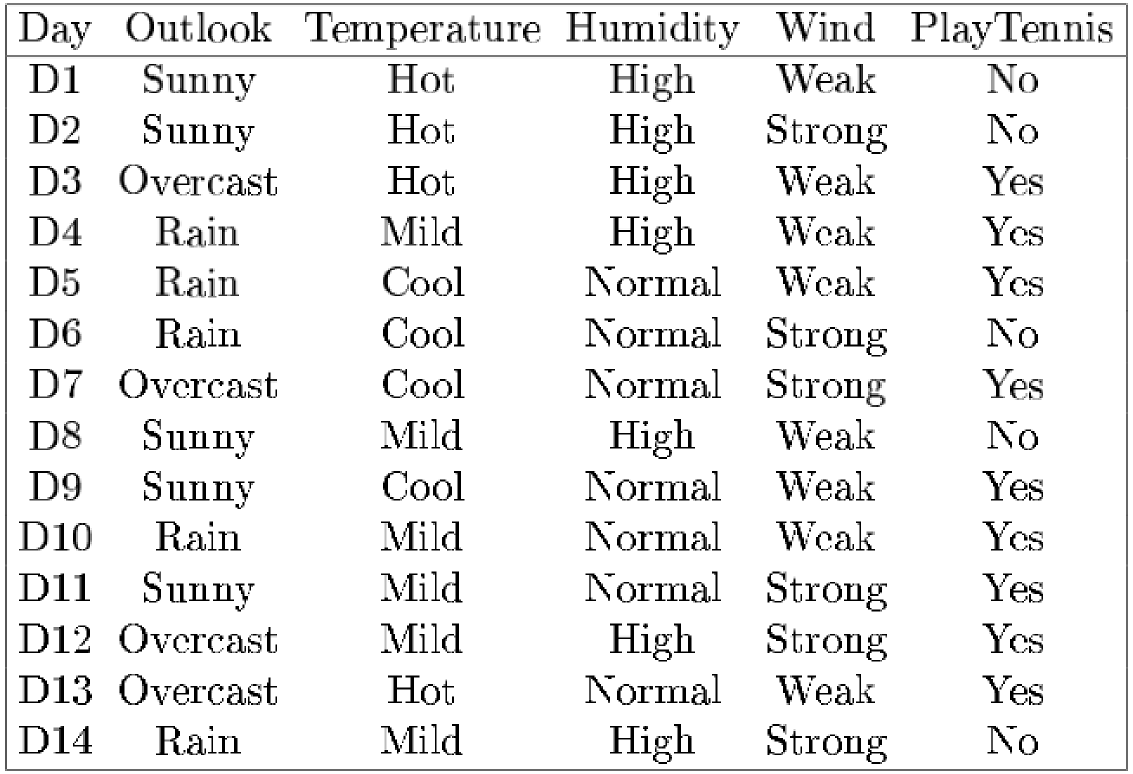
Decision tree induction is the learning of decision trees from class-labeled training tuples.

A decision tree is a flowchart-like tree structure, where

- each internal node (non leaf node) denotes a test on an attribute,
- each branch represents an outcome of the test, and
- each leaf node (or terminal node) holds a class label.
- The topmost node in a tree is the root node.

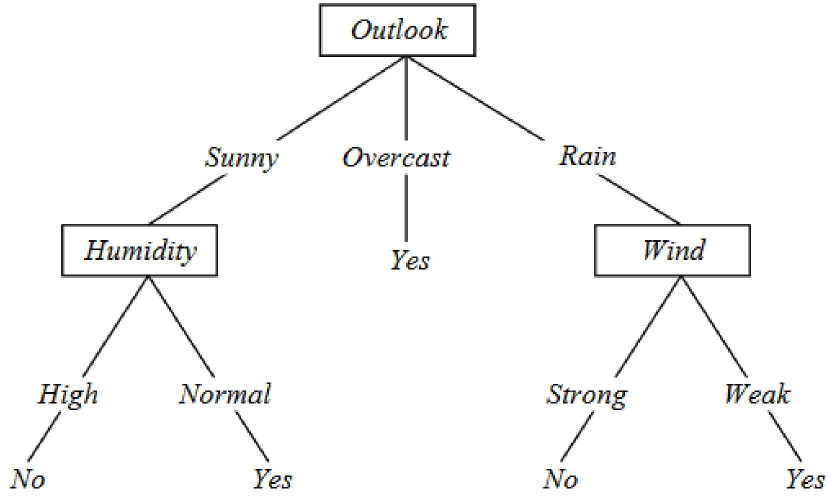
EXAMPLE- DECISION TREE

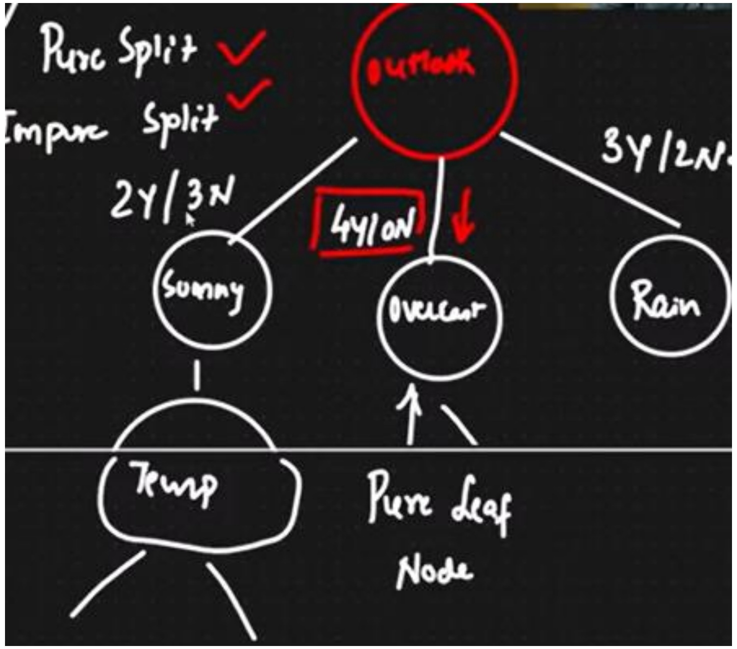
Dataset



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision Tree for PlayTennis dataset





2 Questions

1) How to check Purity? Pure Split or pure Node

Entropy

Gini Impurity/ Coefficient/Gini Index

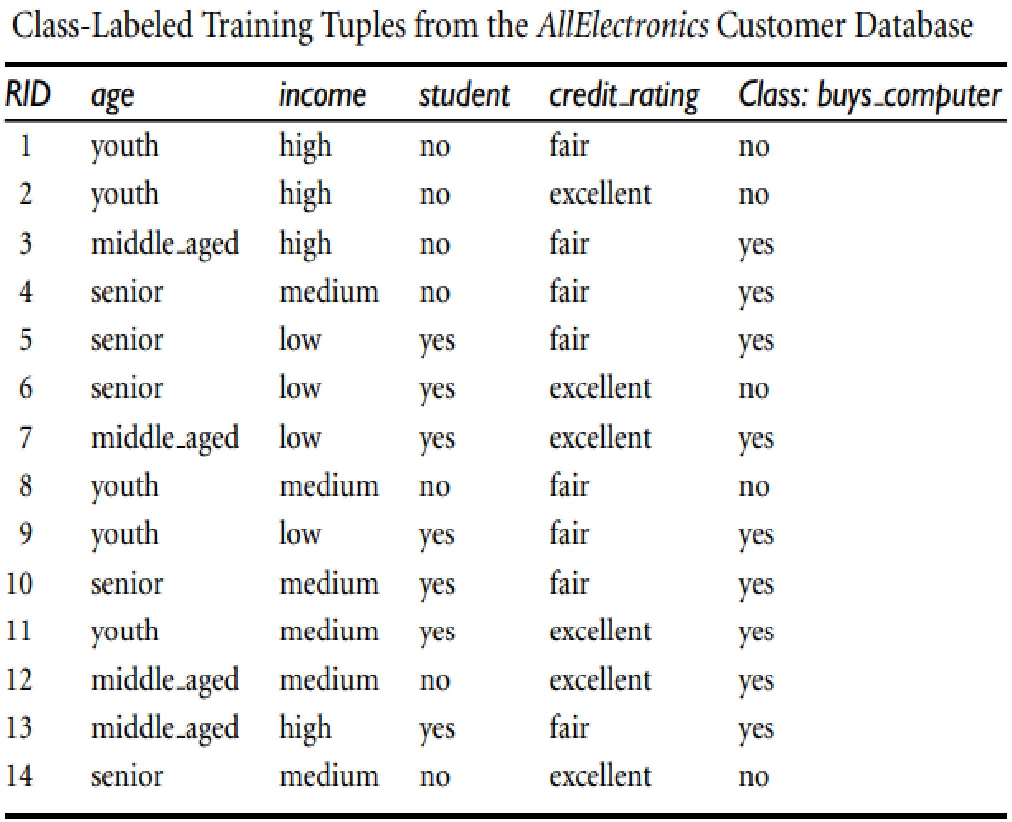
2) How the features are Selected?

Using Attribute Selection Measures

Example: Information Gain

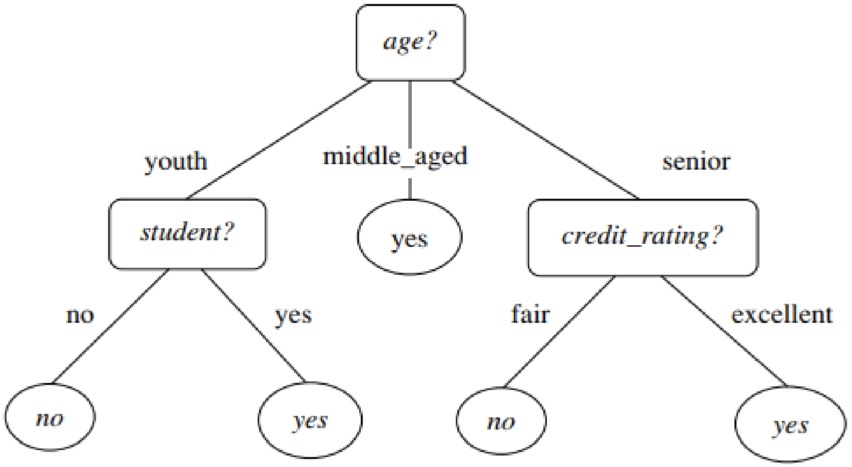
EXAMPLE 2

EXAMPLE DATASET:

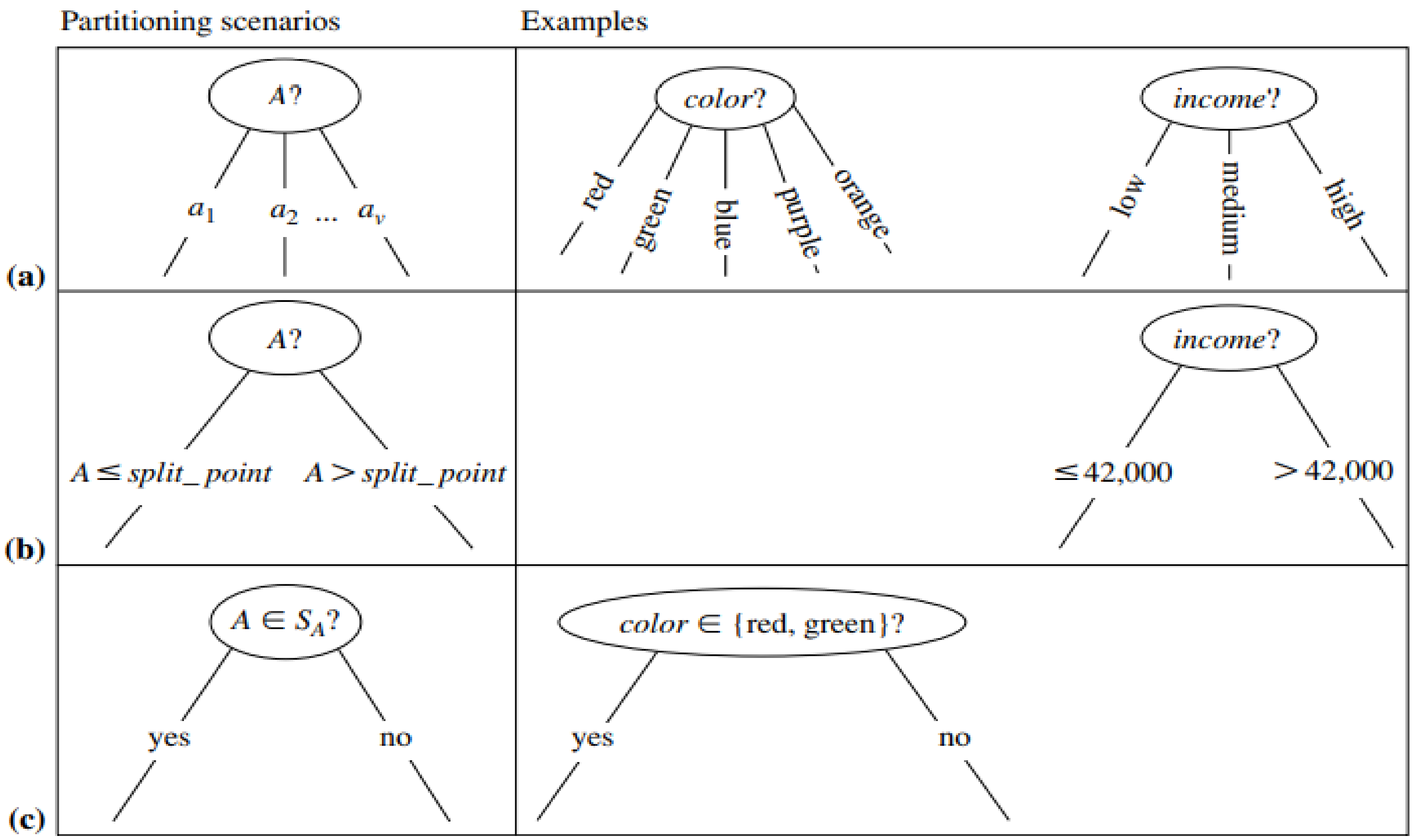


Class-Labeled Training Tuples from the *AllElectronics* Customer Database

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no



2 A decision tree for the concept *buys_computer*, indicating whether an *AllElectronics* customer is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer* = *yes* or *buys_computer* = *no*).



This figure shows three possibilities for partitioning tuples based on the splitting criterion, each with examples. Let A be the splitting attribute. (a) If A is discrete-valued, then one branch is grown for each known value of A. (b) If A is continuous-valued, then two branches are grown, corresponding to $A \leq split\_point$ and $A > split\_point$. (c) If A is discrete-valued and a binary tree must be produced, then the test is of the form $A \in S_A$, where S_A is the splitting subset for A.

Attribute selection measures :

During tree construction, attribute selection measures are used to select the attribute that best partitions the tuples into distinct classes.

Popular measures of attribute selection are

- Information Gain
- Gain ratio
- Gini index

Tree pruning

When decision trees are built, many of the branches may reflect noise or outliers in the training data. Tree pruning attempts to identify and remove such branches, with the goal of improving classification accuracy on unseen data.

DT ISSUES

- Choosing Splitting Attributes
  - Which attributes to use for splitting attributes impacts the performance applying the built DT.
- Ordering of Splitting Attributes
  - Order in which attributes are chosen are important.

DT ISSUES

- Splits
  - Associated with the ordering of attributes is the number of splits to take.
- Tree Structure
  - Balanced tree with the fewest levels is desirable.
- Stopping Criteria
  - Creation of tree definitely stops when the training data are perfectly classified.
  - There may be situations when stopping earlier would be desirable to prevent the creation of larger trees.
  - Stopping earlier may be performed to prevent over fitting.

DT ISSUES

- Training Data
  - Structure of the DT created depends on the training data.
  - If the training data set is too small, then the generated tree might not be specific enough to work properly with the more general data.
- Pruning
  - Removing redundant comparisons or remove subtrees to achieve better performance.

ADVANTAGES AND DISADVANTAGES

- Advantages:
  - DT’s are easy to use and efficient.
  - Rules can be generated that are easy to interpret and understand.
  - Scale well for large databases because the tree size is independent of the database size.
- Disadvantages:
  - Do not easily handle continuous data.
  - Overfitting may occur.

ALGORTIHM

1. **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
2. **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
3. **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
4. **Step-4:** Generate the decision tree node, which contains the best attribute.
5. **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

6.

DECISION TREE INDUCTION – HOW A DECISION TREE WORKS

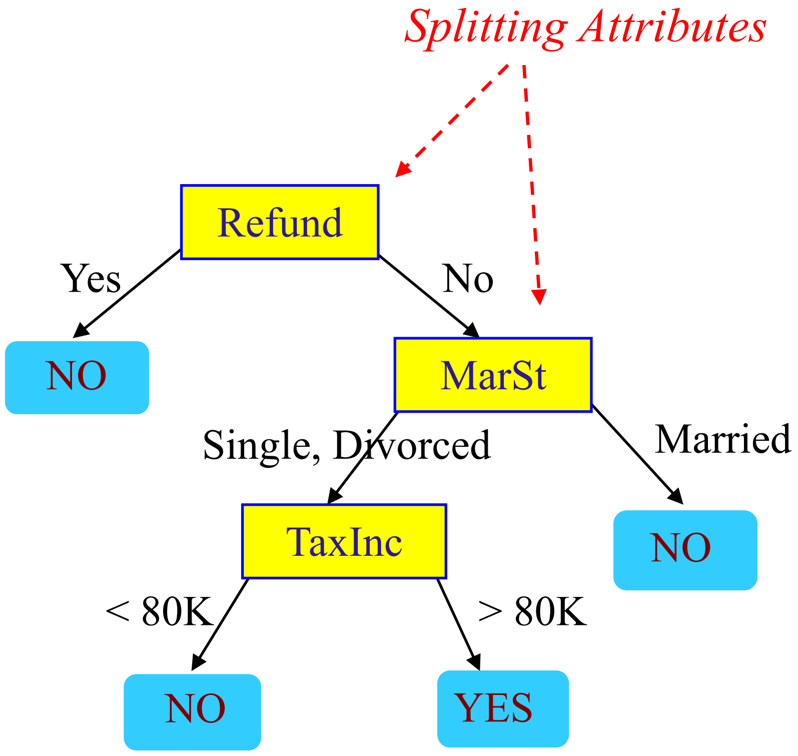
1. ***Decision Tree (DT):***
   - Tree where the root and each internal node is labeled with a question.
   - The arcs represent each possible answer to the associated question.
   - Each leaf node represents a prediction of a solution to the problem.
2. ***Root node:*** No incoming edges and zero or more outgoing edges.
3. ***Internal node:*** Exactly one incoming edges and two or more outgoing edges.
4. ***Leaf node:*** Exactly one incoming edge and no outgoing edges. Leaf node indicates class to which the corresponding tuple belongs.

HOW TO BUILD A DECISION TREE

Binary categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data

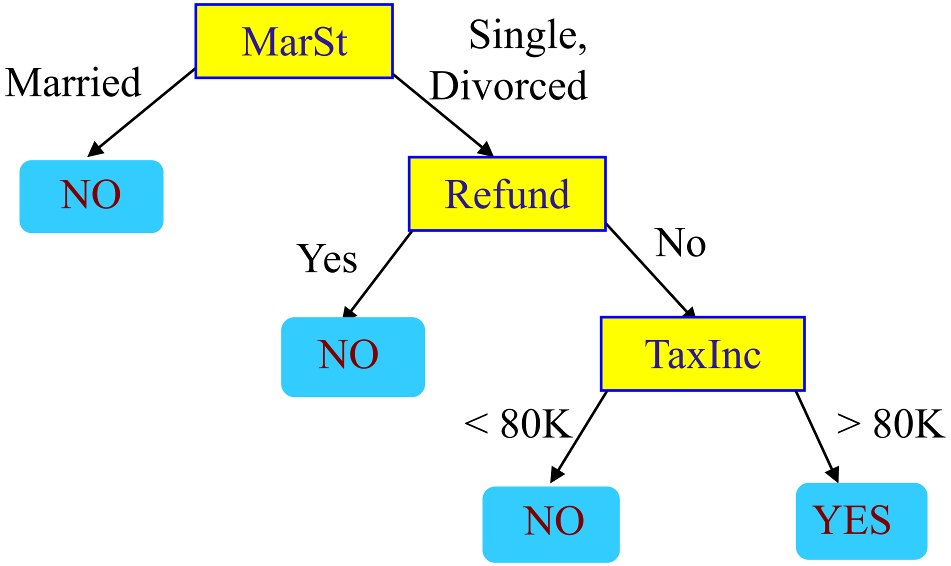


Model: Decision Tree

HOW TO BUILD A DECISION TREE

categorical categorical continuous class

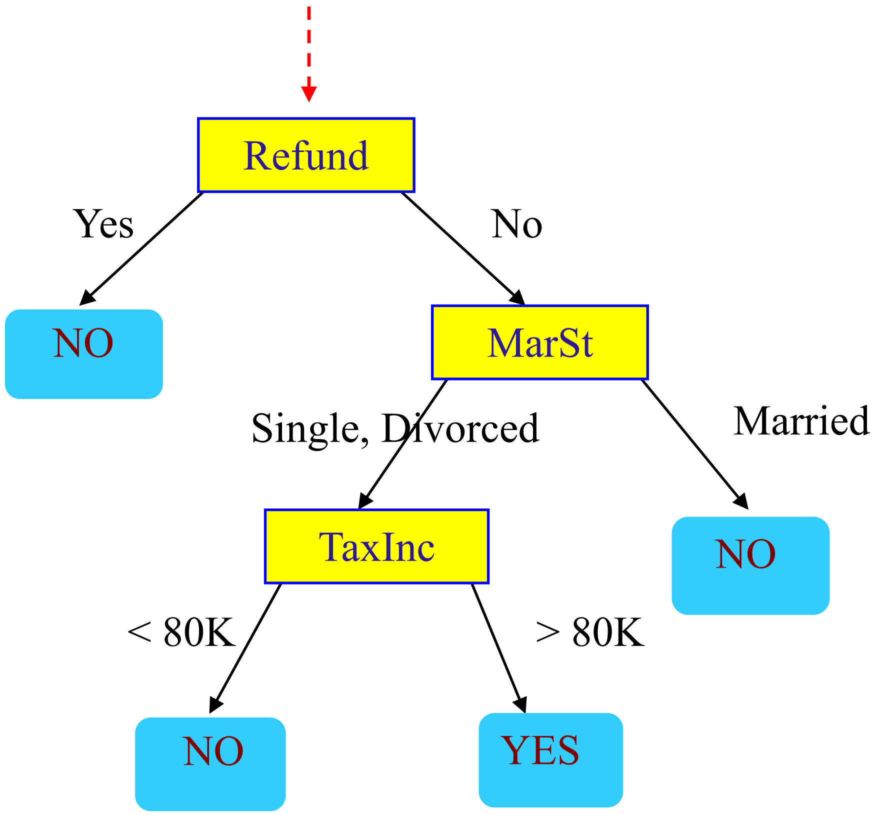
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

HOW TO BUILD A DECISION TREE

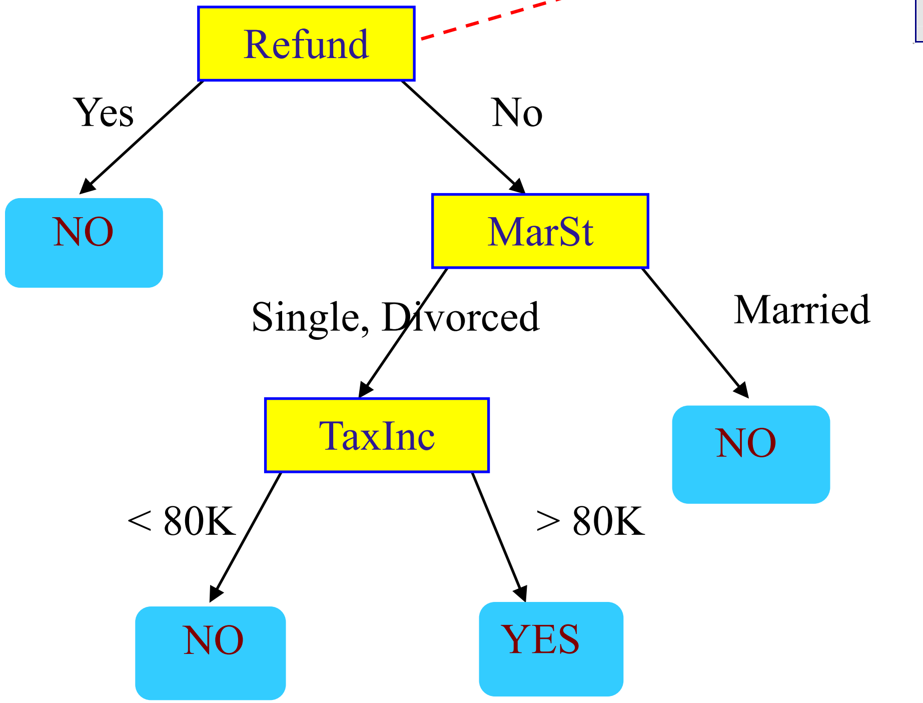
Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

HOW TO BUILD A DECISION TREE



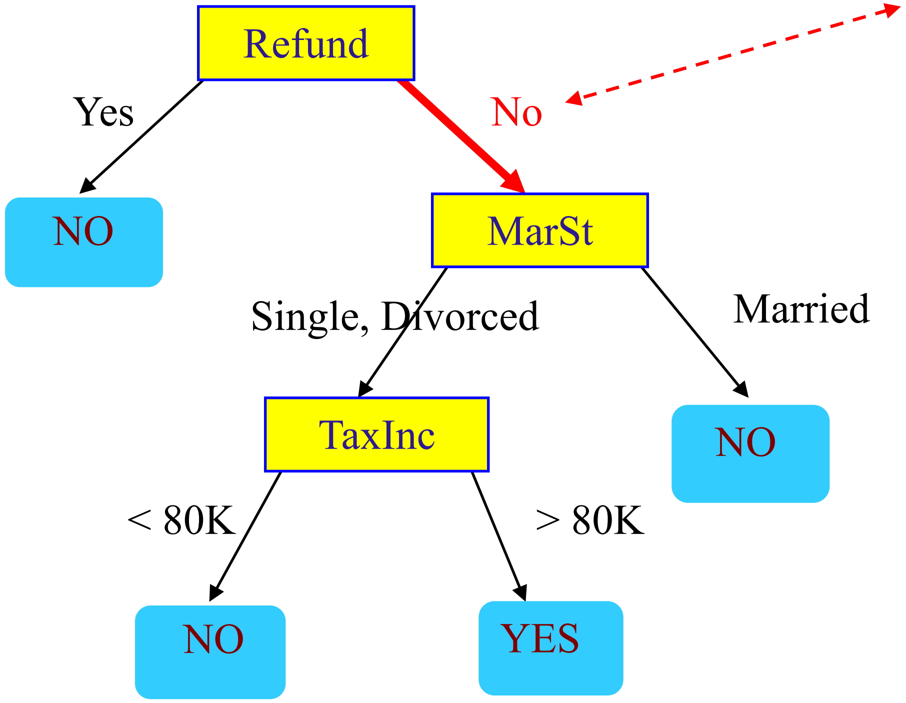
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

HOW TO BUILD A DECISION TREE

Test Data

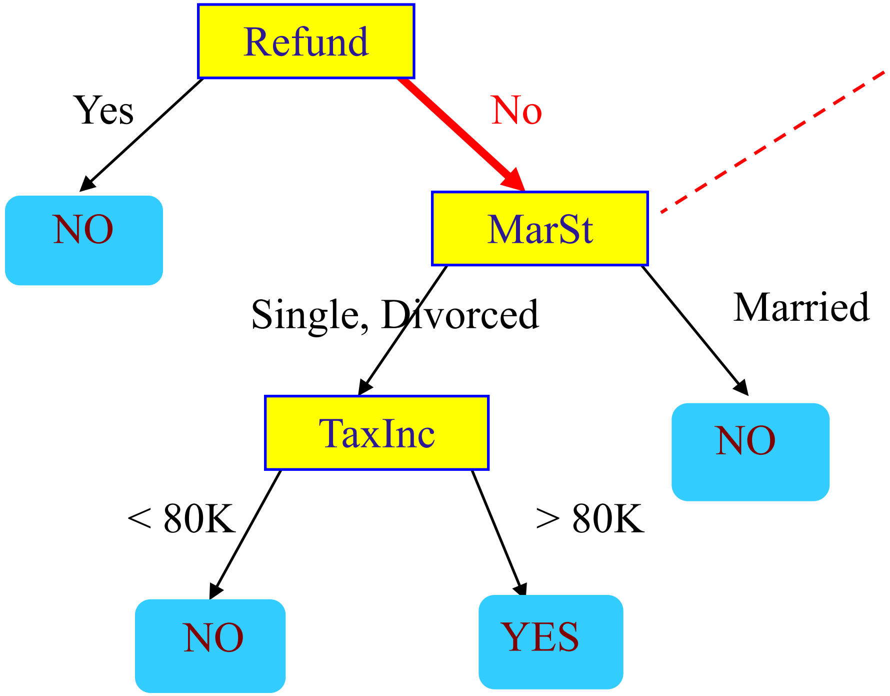
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



HOW TO BUILD A DECISION TREE

Test Data

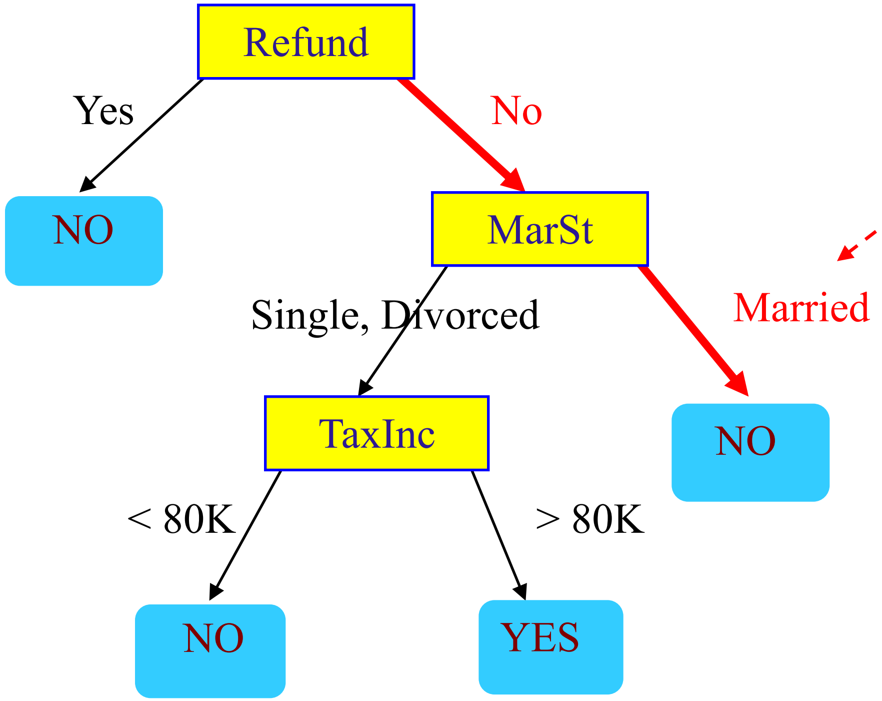
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



HOW TO BUILD A DECISION TREE

Test Data

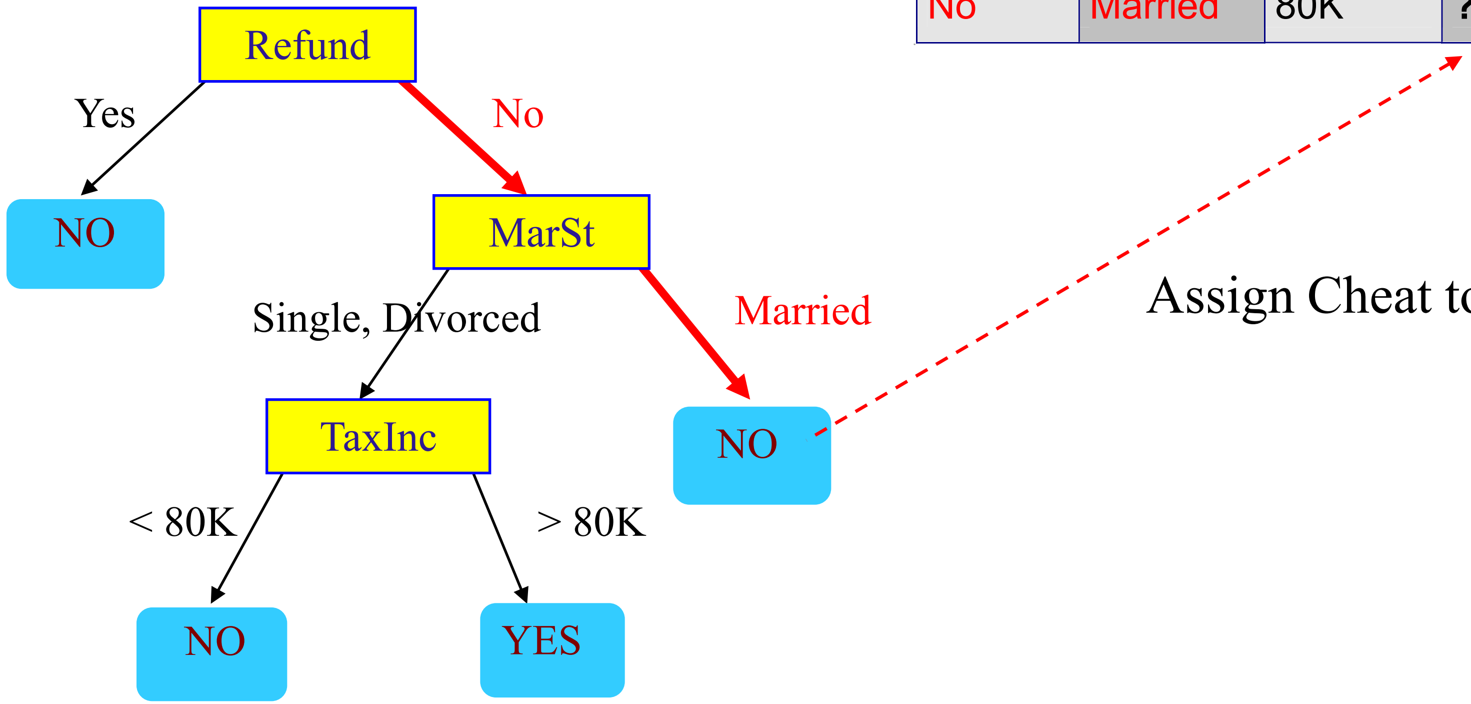
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



HOW TO BUILD A DECISION TREE

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to “No”

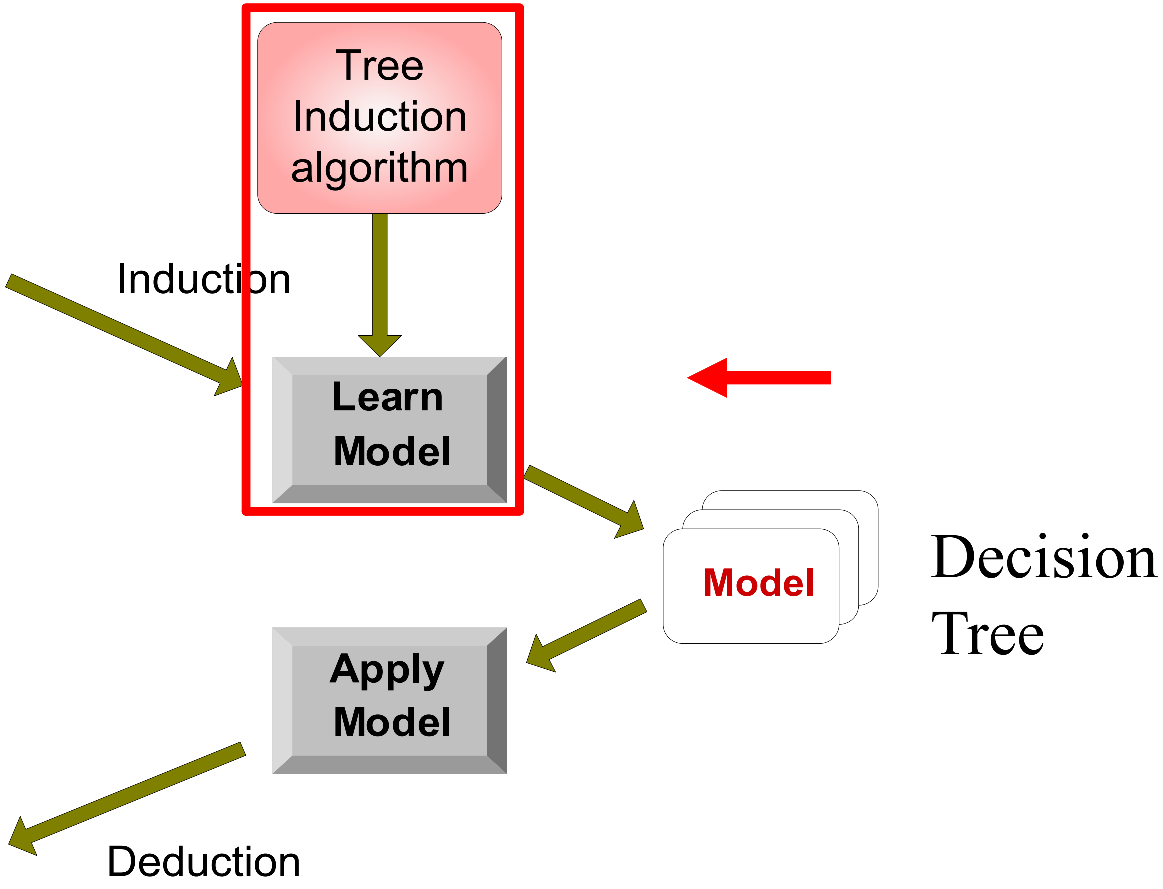
HOW TO BUILD A DECISION TREE

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



DIFFERENT TYPES OF DECISION TREE ALGORITHM

- **ID3**
- C4.5
- **CART(Classification and Regression Trees)**
- CHAID(Chi-Squared Automatic interaction Detector)
- MARS:
- Conditional Inference Trees

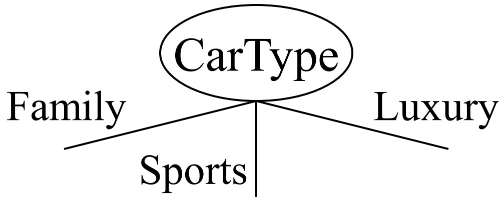
METHODS FOR SPECIFYING ATTRIBUTE TEST CONDITIONS

1. Depends on attribute types
   - Nominal
   - Ordinal
   - Continuous

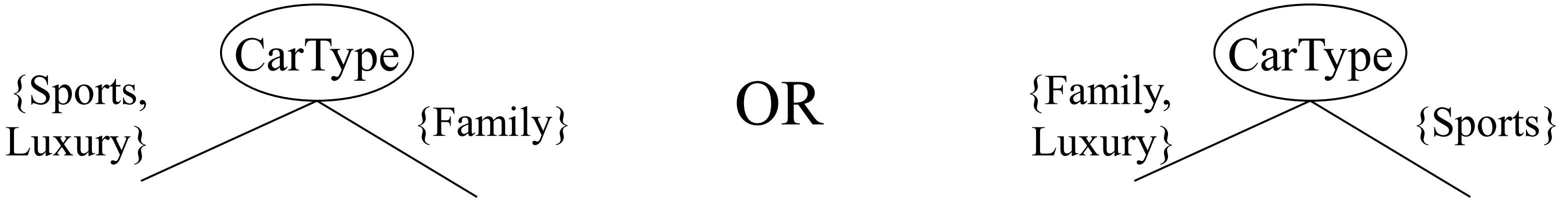
2. Depends on number of ways to split
   - 2-way split
   - Multi-way split

SPLITTING BASED ON NOMINAL ATTRIBUTES

- Multi-way split: Use as many partitions as distinct values.

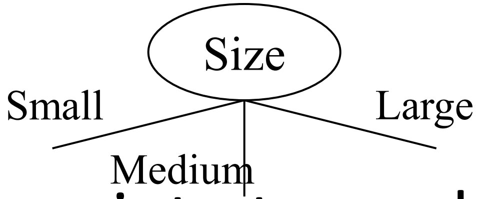


- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.

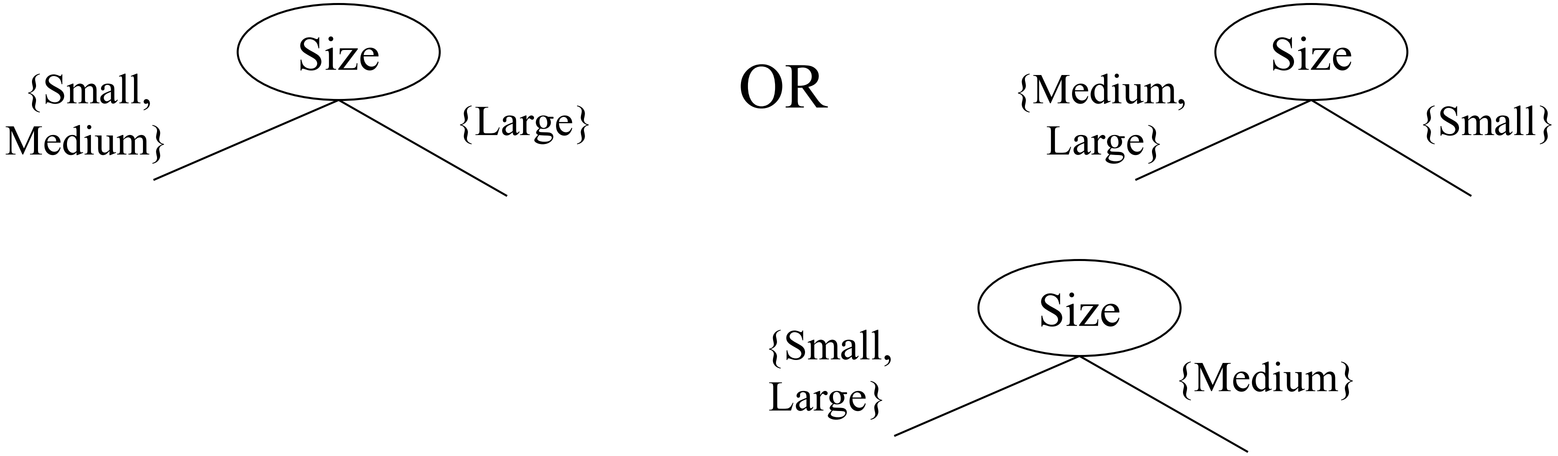


SPLITTING BASED ON ORDINAL ATTRIBUTES

- Multi-way split: Use as many partitions as distinct values.



- Binary split: Divides values into two subsets. Need to find optimal partitioning.



SPLITTING BASED ON CONTINUOUS ATTRIBUTES

The test condition can be expressed in terms of a binary decision *($A < v$?) or ($A >= v$?),* whose outcomes are Yes / No, or as a range query whose outcomes are *$v_i <= A <= v_i+1$, for i = 1, 2, … k.*

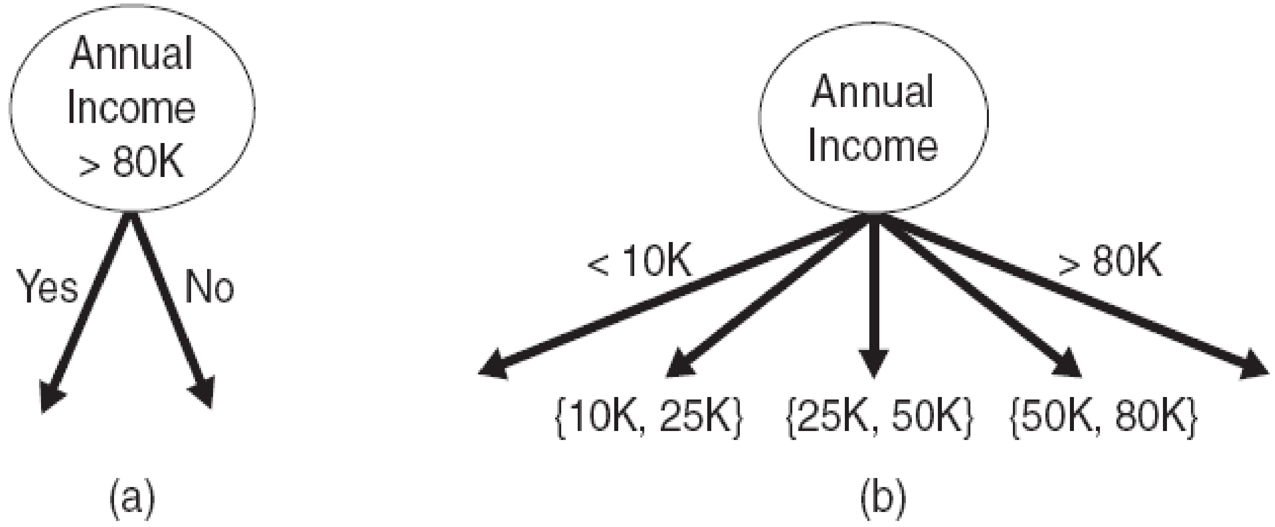


Figure 4.11. Test condition for continuous attributes.

MEASURES FOR SELECTING THE BEST SPLIT

1. **p(i|t)**: fraction of records belonging to class i at a given node t.
2. **Best split** is selected based on the degree of **impurity** of the child nodes
   - Class distribution **(0,1)** has **high purity**
   - Class distribution **(0.5,0.5)** has the **smallest purity (highest impurity)**
3. **Intuition:** high purity → small value of impurity measures → better split.

ID3 ALGORITHM

- ID3 stands for Iterative Dichotomiser 3
- is named such because the algorithm iteratively (repeatedly) dichotomizes(divides) features into two or more groups at each step.
- Invented by Ross Quinlan
- ID3 uses a **top-down greedy** approach to build a decision tree.
- **top-down:** we start building the tree from the top and
- **Greedy** : at each iteration we select the best feature at the present moment to create a node.
- Most generally ID3 is only used for classification problems with nominal features only.
- **Build a decision tree** by selecting the best attribute that yields **maximum Information Gain (IG) or minimum Entropy (H)**.

WHAT IS ENTROPY AND INFORMATION GAIN?

Entropy is a measure of the amount of uncertainty in the dataset S.

The expected information needed to classify a tuple in D or entropy is given by

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

In ID3, entropy is calculated for each remaining attribute. The attribute with the smallest entropy is used to split the set S on that particular iteration.

Entropy = 0 implies it is of pure class, that means all are of same category.

Information Gain IG(A)

tells us how much uncertainty in S was reduced after splitting set S on attribute

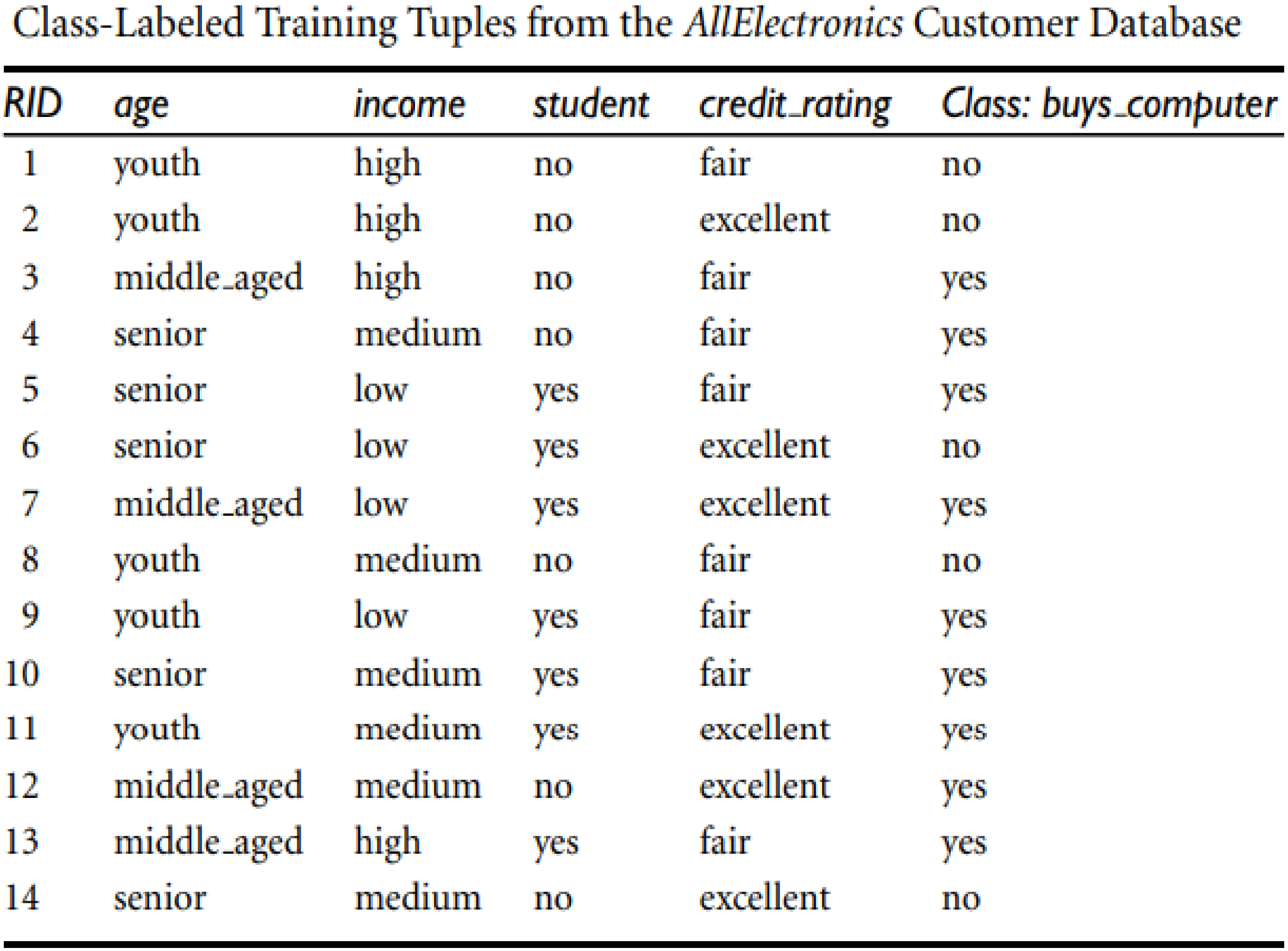
Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

ID3 EXAMPLE



Class-Labeled Training Tuples from the *AllElectronics* Customer Database

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- The class label attribute, buys computer, has two distinct values (namely, {yes, no}); therefore, there are two distinct classes (i.e., m = 2).
- Let class C1 correspond to yes and class C2 correspond to no.
- There are nine tuples of class yes and five tuples of class no.
- A (root) node N is created for the tuples in D.
- To find the splitting criterion for these tuples, we must compute the information gain of each attribute

1) To compute the expected information needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

2) Next, we need to compute the expected information requirement for each attribute.

Let’s start with the attribute age.

We need to look at the distribution of yes and no tuples for each category of age.

For the age category “youth,” there are two yes tuples and three no tuples. For the category “middle aged,” there are four yes tuples and zero no tuples. For the category “senior,” there are three yes tuples and two no tuples

the expected information needed to classify a tuple in D if the tuples are partitioned according to age is

$$
\begin{aligned}
Info_{age}(D) &= \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\
&+ \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4}\right) \\
&+ \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) \\
&= 0.694 \text{ bits.}
\end{aligned}
$$

- Information needed (after using A to split D into v partitions) to classify D:

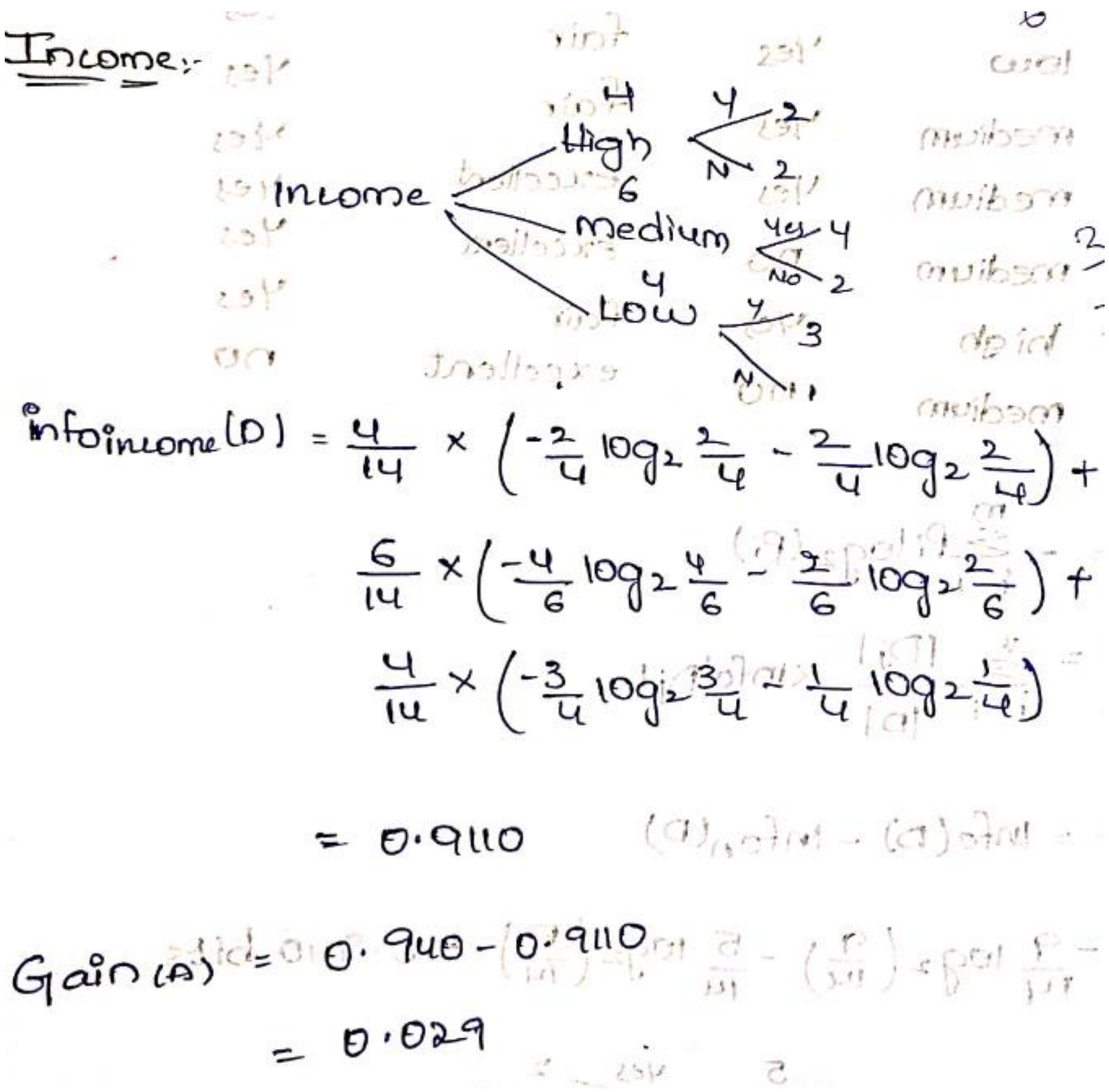
$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

the gain in information from such a partitioning would be

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$



Income:-

Income

4 High 6 — Y 2, N 2

Medium — Yes 4, No 2

4 Low — Y 3, N 1

$$\text{info}_{income}(D) = \frac{4}{14}\times\left(-\frac{2}{4}\log_2\frac{2}{4}-\frac{2}{4}\log_2\frac{2}{4}\right)+$$

$$\frac{6}{14}\times\left(-\frac{4}{6}\log_2\frac{4}{6}-\frac{2}{6}\log_2\frac{2}{6}\right)+$$

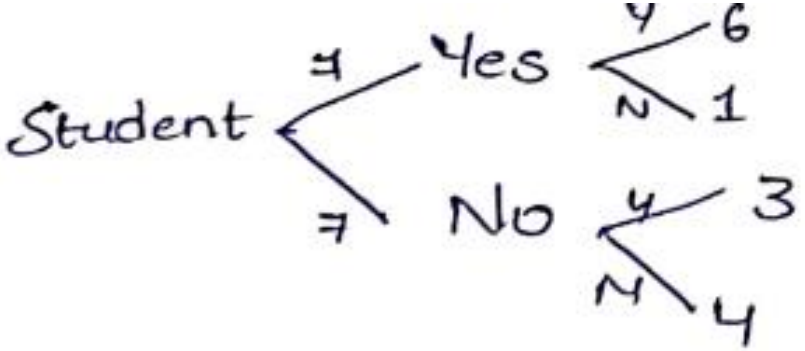
$$\frac{4}{14}\times\left(-\frac{3}{4}\log_2\frac{3}{4}-\frac{1}{4}\log_2\frac{1}{4}\right)$$

$$= 0.9110$$

$$\text{Gain}(A) = 0.940 - 0.9110$$

$$= 0.029$$

Student



$$\text{info}_{stu}(D) = \frac{7}{14} \times \left(-\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7}\right) +$$

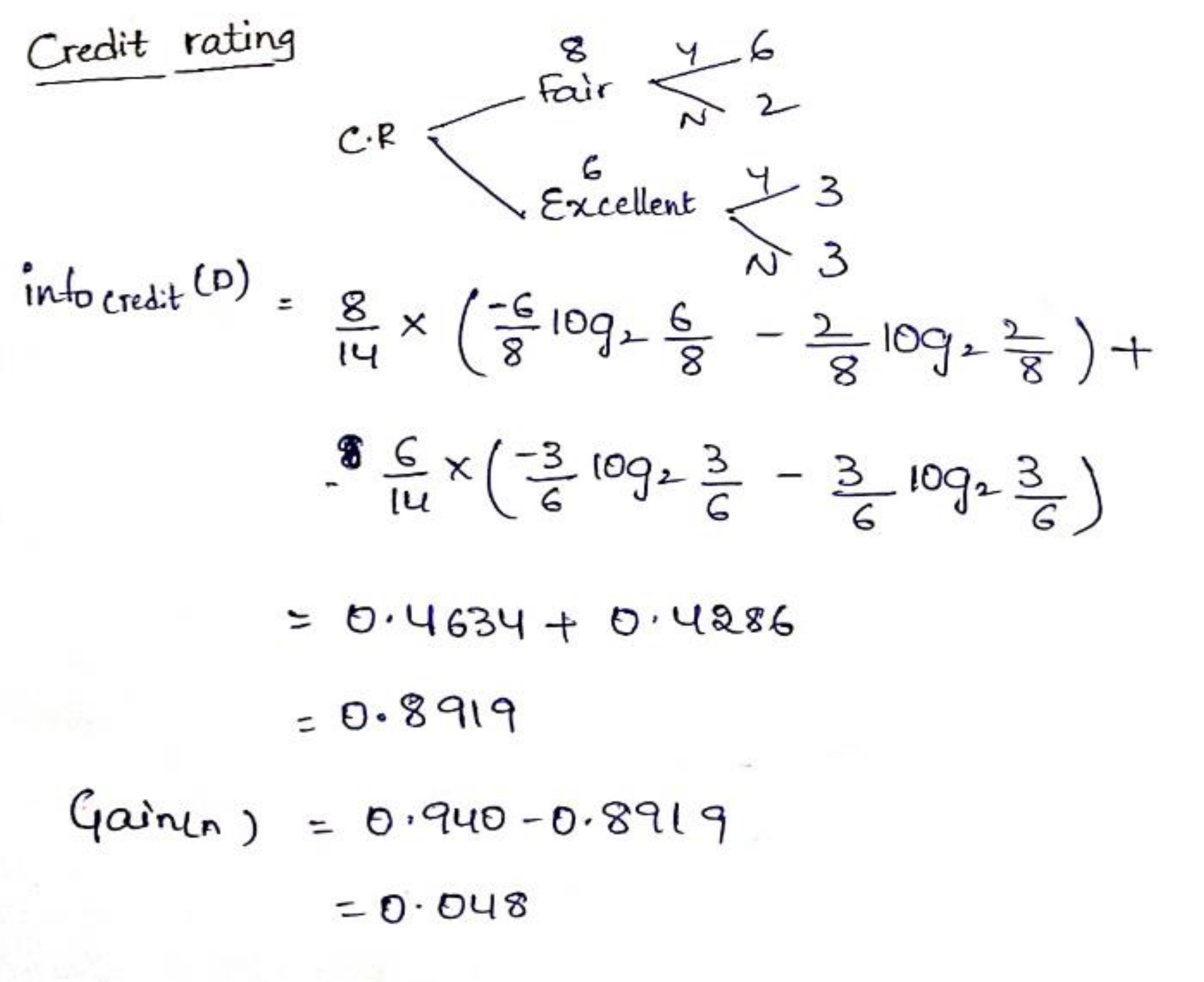
$$\frac{7}{14} \times \left(\frac{-3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7}\right)$$

$$= 0.2958 + 0.4926$$

$$= 0.7884$$

$$\text{Gain}(A) = 0.940 - 0.7884$$

$$= 0.151$$



Credit rating

C.R

8 Fair: Y 6, N 2

6 Excellent: Y 3, N 3

$$\text{info}_{credit}(D) = \frac{8}{14} \times \left(\frac{-6}{8} \log_2 \frac{6}{8} - \frac{2}{8} \log_2 \frac{2}{8} \right) + \frac{6}{14} \times \left(\frac{-3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6} \right)$$

$$= 0.4634 + 0.4286$$

$$= 0.8919$$

$$\text{Gain}_{(n)} = 0.940 - 0.8919$$

$$= 0.048$$

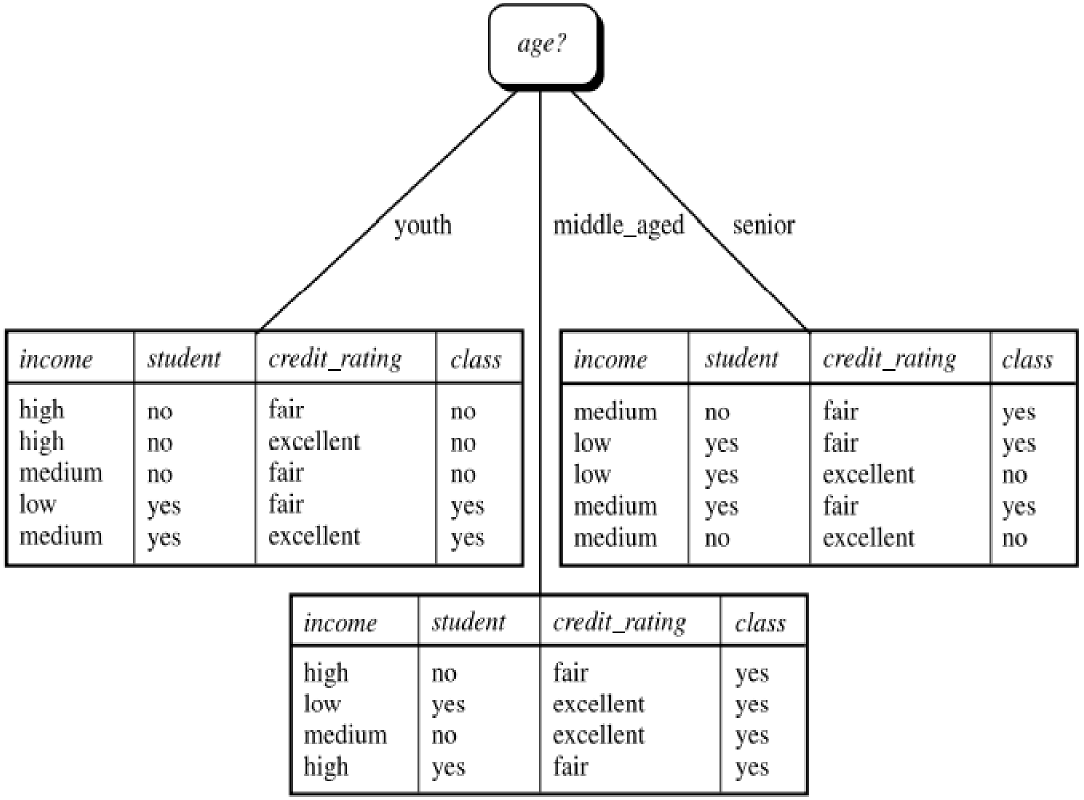
Similarly, we can compute

Gain(income) = 0.029 bits,

Gain(student) = 0.151 bits, and

Gain(credit rating) = 0.048 bits.

Because age has the highest information gain among the attributes, it is selected as the splitting attribute. Node N is labeled with age, and branches are grown for each of the attribute’s values. The tuples are then partitioned accordingly, as shown in Figure



income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

5 The attribute *age* has the highest information gain and therefore becomes the splitting attribute at the root node of the decision tree. Branches are grown for each outcome of *age*. The tuples are shown partitioned accordingly.

CHARACTERISTICS OF ID3 ALGORITHM ARE AS FOLLOWS:

- ID3 uses a greedy approach that's why it does not guarantee an optimal solution; it can get stuck in local optimums.
- ID3 can overfit to the training data (to avoid overfitting, smaller decision trees should be preferred over larger ones).
- This algorithm usually produces small trees, but it does not always produce the smallest possible tree.
- ID3 is harder to use on continuous data (if the values of any given attribute is continuous, then there are many more places to split the data on this attribute, and searching for the best value to split by can be time consuming).
- Attributes must be nominal values, dataset must not include missing data.

IMPLEMENTATION

Decision trees in scikit-learn are implemented in the DecisionTreeRegressor and DecisionTreeClassifier classes. scikit-learn only implements pre-pruning, not post-pruning.

```
from sklearn.tree import DecisionTreeClassifier

cancer = load_breast_cancer()

X_train, X_test, y_train, y_test = train_test_split( cancer.data, cancer.target,
stratify=cancer.target, random_state=42)

tree = DecisionTreeClassifier(random_state=0)

tree.fit(X_train, y_train)

print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))

print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
```

```
Accuracy on training set: 1.000 Accuracy on test set: 0.937
```

IMPLEMENTATION

```
tree = DecisionTreeClassifier(max_depth=4, random_state=0)

tree.fit(X_train, y_train)

print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
```

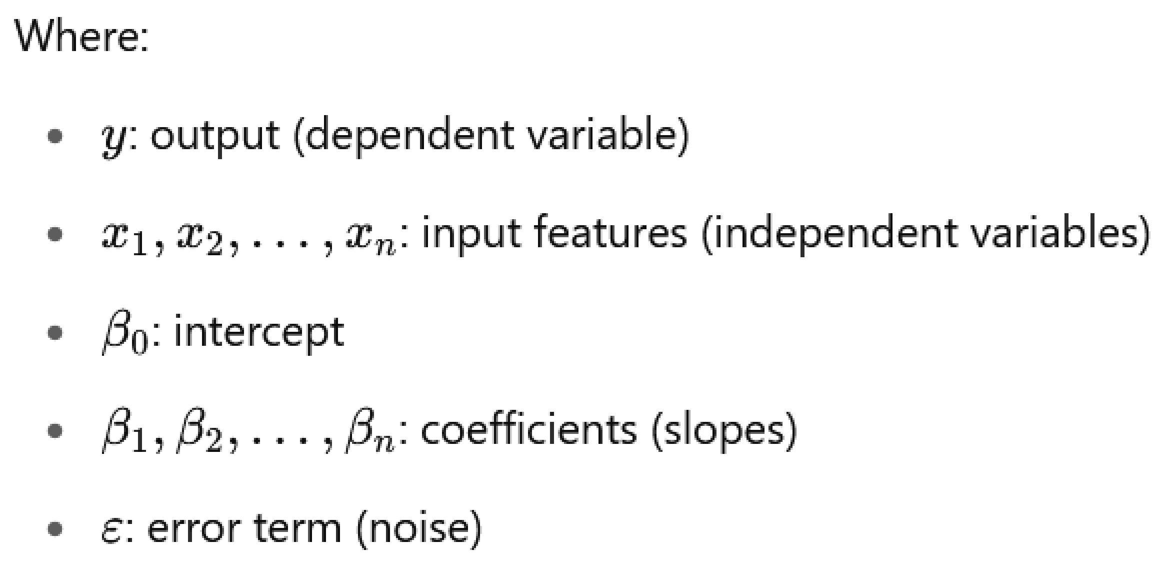
```
Accuracy on training set: 0.988 Accuracy on test set: 0.951
```

WHAT ARE LINEAR MODELS?

A **linear model** is a mathematical model that assumes a **linear relationship** between the **input variables (independent variables)** and the **output (dependent variable)**.

General Form of a Linear Model:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$



Where:

- y: output (dependent variable)
- $x_1, x_2, \ldots, x_n$: input features (independent variables)
- β_0: intercept
- $\beta_1, \beta_2, \ldots, \beta_n$: coefficients (slopes)
- ε: error term (noise)

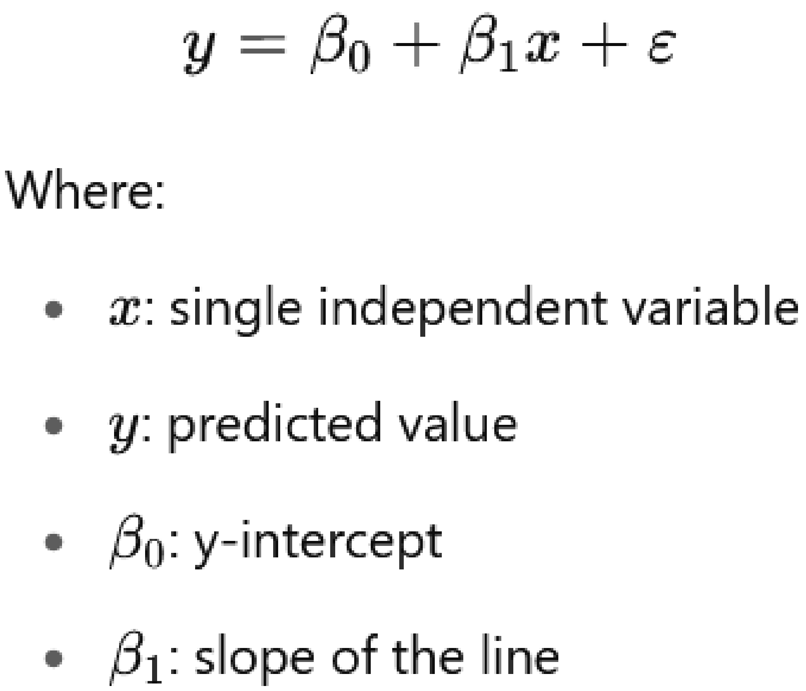
WHAT IS LINEAR REGRESSION?

Linear Regression is a type of linear model used to predict a continuous output variable based on one or more input features.

Types of Linear Regression:

Type	Description
Simple Linear Regression	Only 1 input feature (e.g., predict salary based on years of experience)
Multiple Linear Regression	More than 1 input feature (e.g., predict house price based on area, location, and number of rooms)

Simple Linear Regression Formula



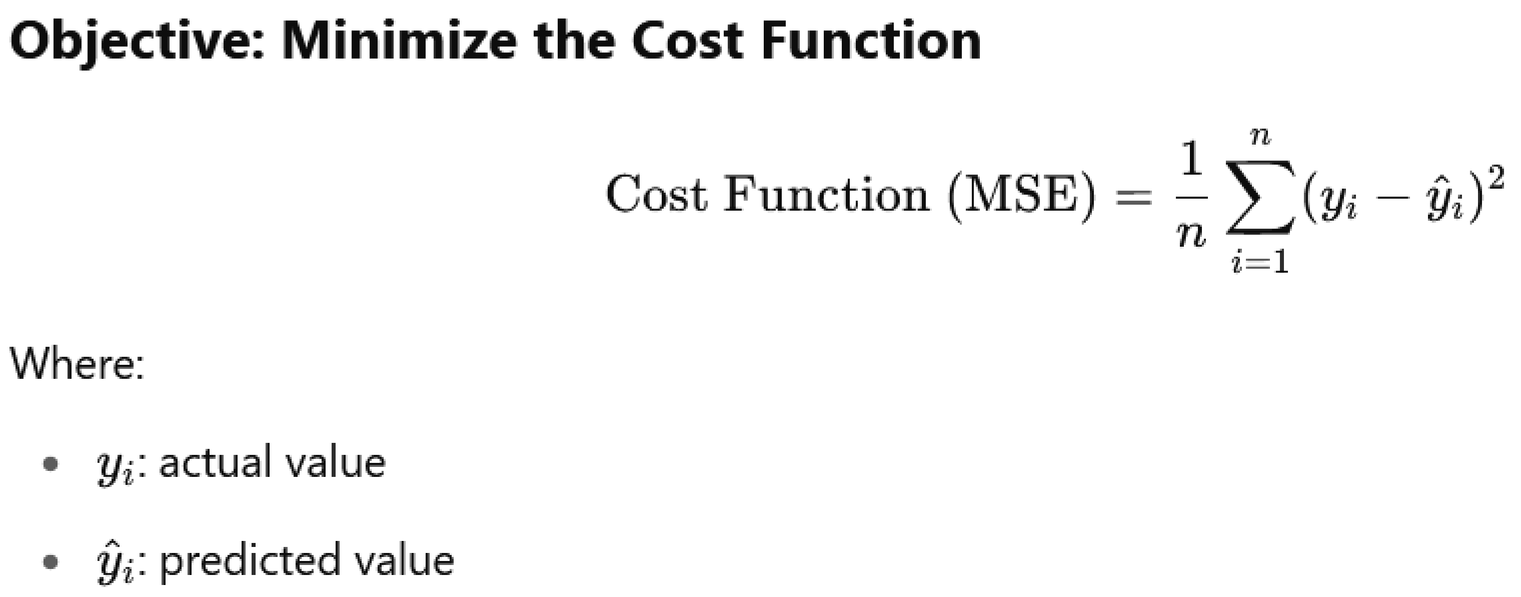
$$y = \beta_0 + \beta_1 x + \varepsilon$$

Where:

- x: single independent variable
- y: predicted value
- β_0: y-intercept
- β_1: slope of the line

How It Works (Mechanism)

Linear regression **finds the best-fit line** by minimizing the error between the predicted values and the actual values using a method called **Least Squares**.



Objective: Minimize the Cost Function

$$\text{Cost Function (MSE)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

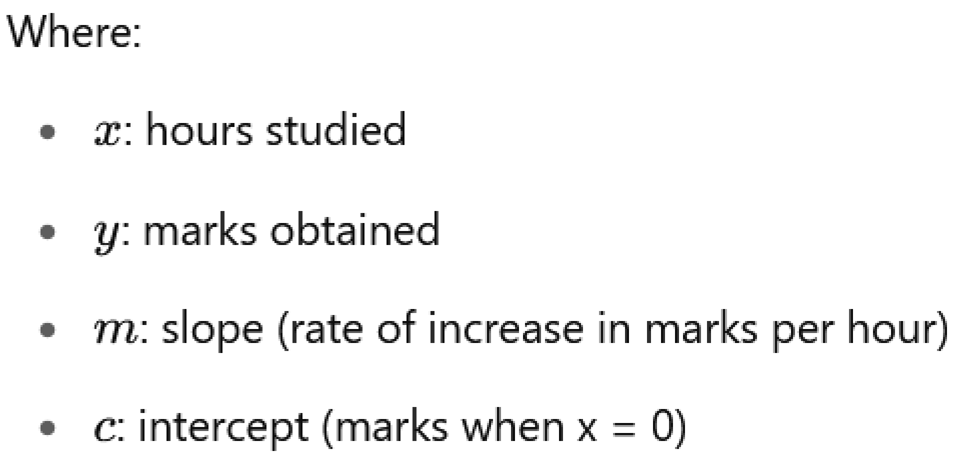
Where:

- y_i: actual value
- $\hat{y}_i$: predicted value

EXAMPLE PROBLEM: PREDICT MARKS FROM HOURS STUDIED

Hours Studied (x)	Marks Obtained (y)
1	40
2	50
3	60
4	70

We want to find a **linear equation** of the form: $y = mx + c$



Where:

- x: hours studied
- y: marks obtained
- m: slope (rate of increase in marks per hour)
- c: intercept (marks when x = 0)

Step-by-Step Solution

Step 1: Find the Mean of x and y

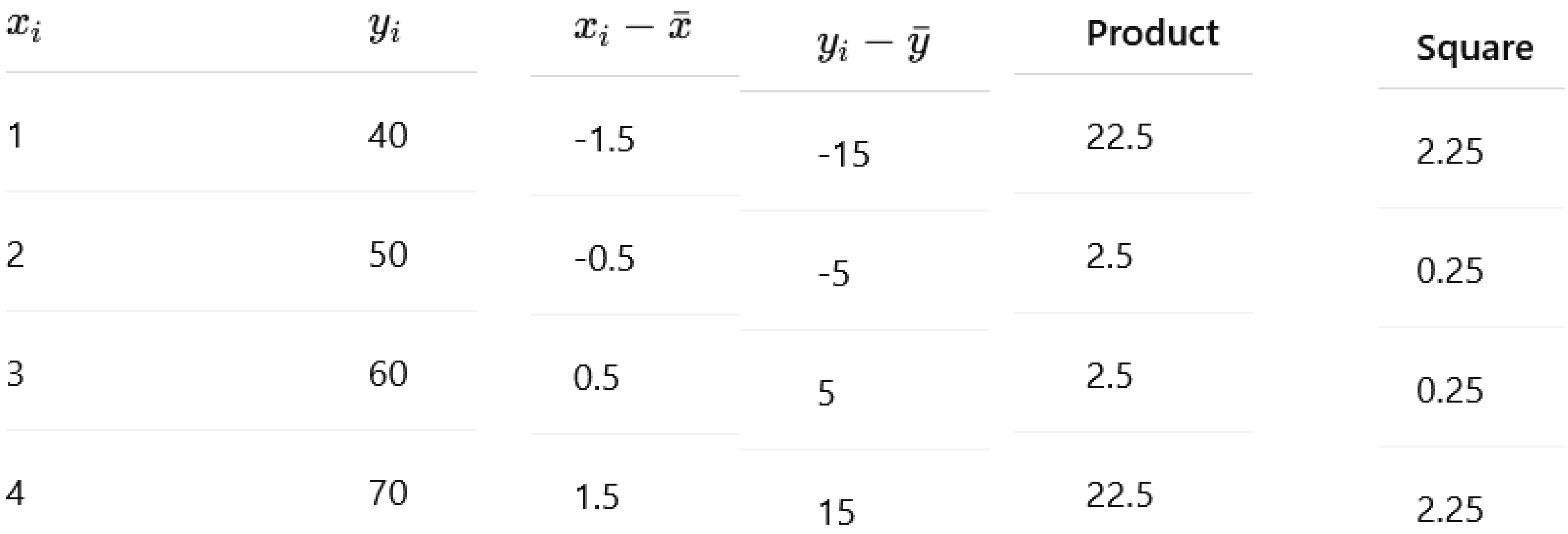
$$\bar{x} = \frac{1+2+3+4}{4} = \frac{10}{4} = 2.5$$

$$\bar{y} = \frac{40+50+60+70}{4} = \frac{220}{4} = 55$$

Step 2: Calculate the Slope (m)

The formula for slope m is:

$$m = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$



x_i	y_i	$x_i - \bar{x}$	$y_i - \bar{y}$	Product	Square
1	40	-1.5	-15	22.5	2.25
2	50	-0.5	-5	2.5	0.25
3	60	0.5	5	2.5	0.25
4	70	1.5	15	22.5	2.25

Now sum the last two columns:

$$\sum (x_i - \bar{x})(y_i - \bar{y}) = 22.5 + 2.5 + 2.5 + 22.5 = 50$$

$$\sum (x_i - \bar{x})^2 = 2.25 + 0.25 + 0.25 + 2.25 = 5$$

$$m = \frac{50}{5} = 10$$

Step 3: Find the Intercept (c)

$$c = \bar{y} - m\bar{x}$$

$$55 - 10(2.5) = 55 - 25 = 30$$

Final Equation: $y = 10x + 30$

Step 4: Predict Marks for 5 Hours

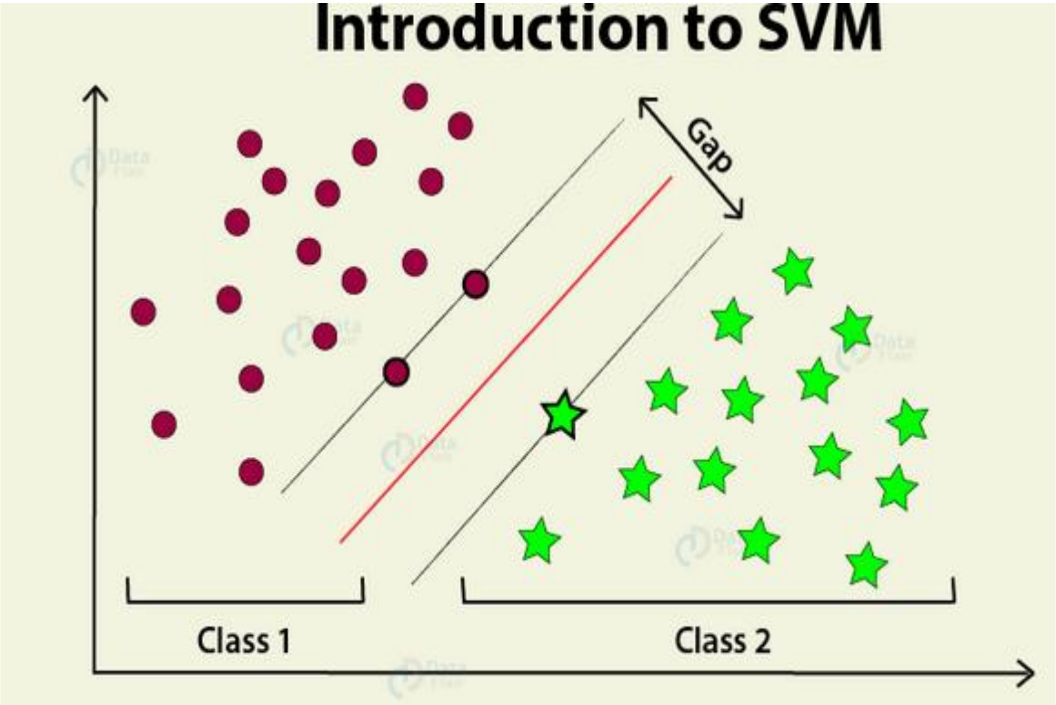
$$y = 10(5) + 30 = 50 + 30 = \boxed{80 \text{ marks}}$$

PYTHON IMPLEMENTATION- SIMPLE LINEAR REGRESSION

SUPPORT VECTOR MACHINES

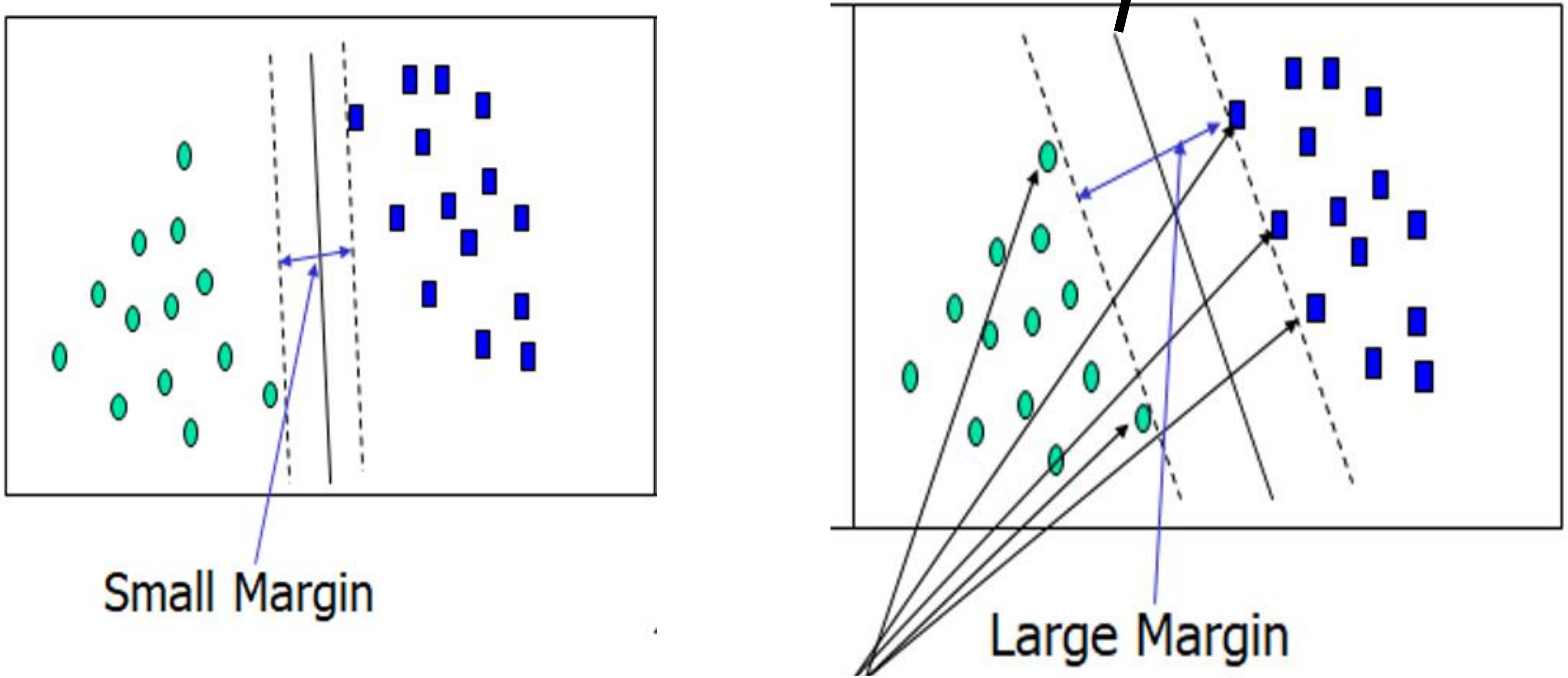
SUPPORT VECTOR MACHINES

- **Support Vector Machine (SVM)** is a **supervised machine learning algorithm** used for **classification** and sometimes **regression**.
- It aims to **find the optimal boundary (hyperplane)** that best separates data into classes.
- In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.



SVM—GENERAL PHILOSOPHY

Hyperplane



Small Margin

Large Margin

Support Vectors

KEY CONCEPTS-SVM

✅ 1. Hyperplane

A decision boundary that separates classes.

- In 2D: a line
- In 3D: a plane
- In higher dimensions: a hyperplane

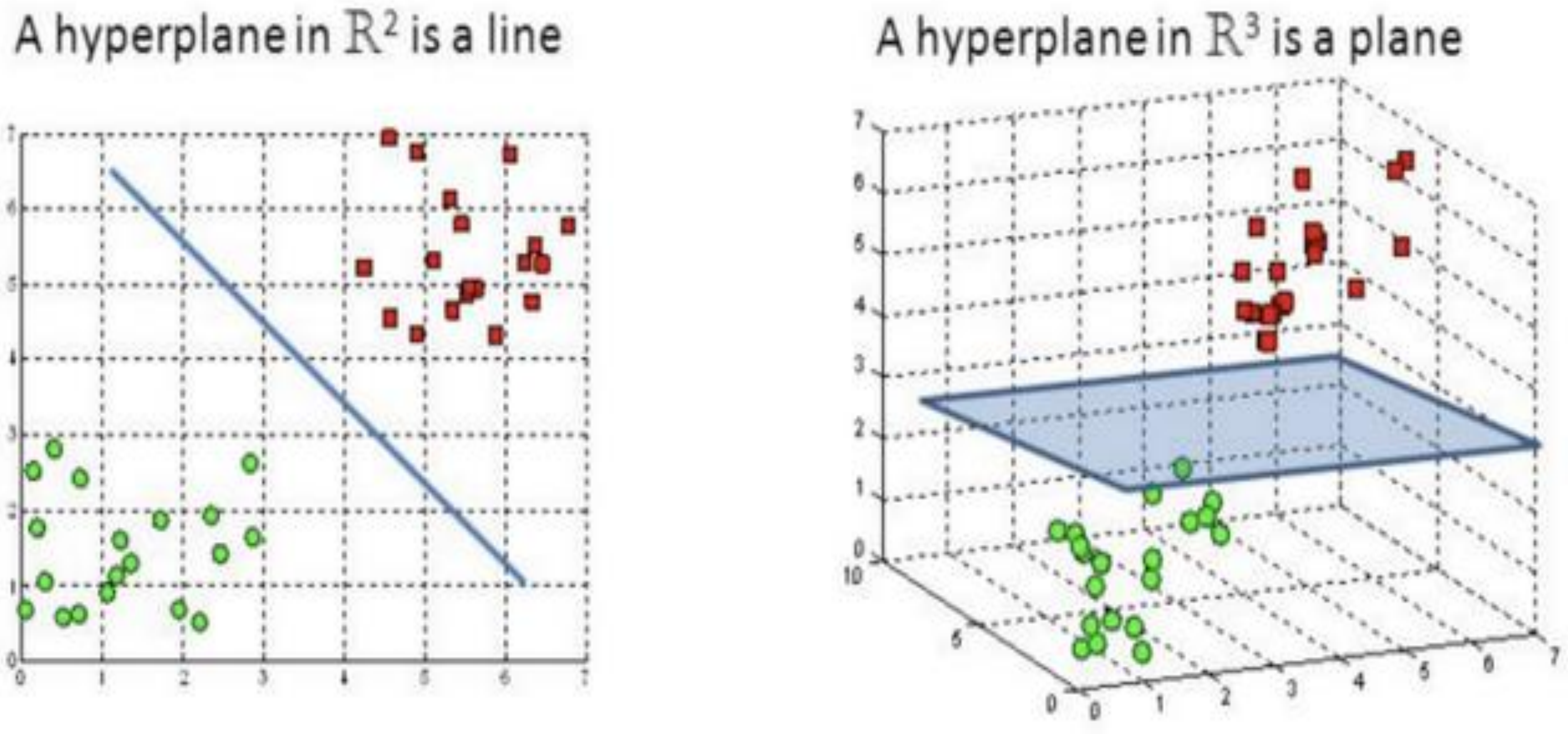
✅ 2. Support Vectors

- The **data points** that are **closest to the hyperplane**.
- These points are **critical** for defining the decision boundary.

✅ 3. Margin

- The **distance** between the hyperplane and the nearest support vectors.
- SVM aims to **maximize this margin** for better generalization.

- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.
- **This best decision boundary is called a hyperplane.**
- To separate the two classes of data points, there are many possible hyperplanes that could be chosen.
- Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes.
- Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.
- Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes.
- Also, the dimension of the hyperplane depends upon the number of
- features. If the number of input features is 2, then the hyperplane is just a line.
- If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

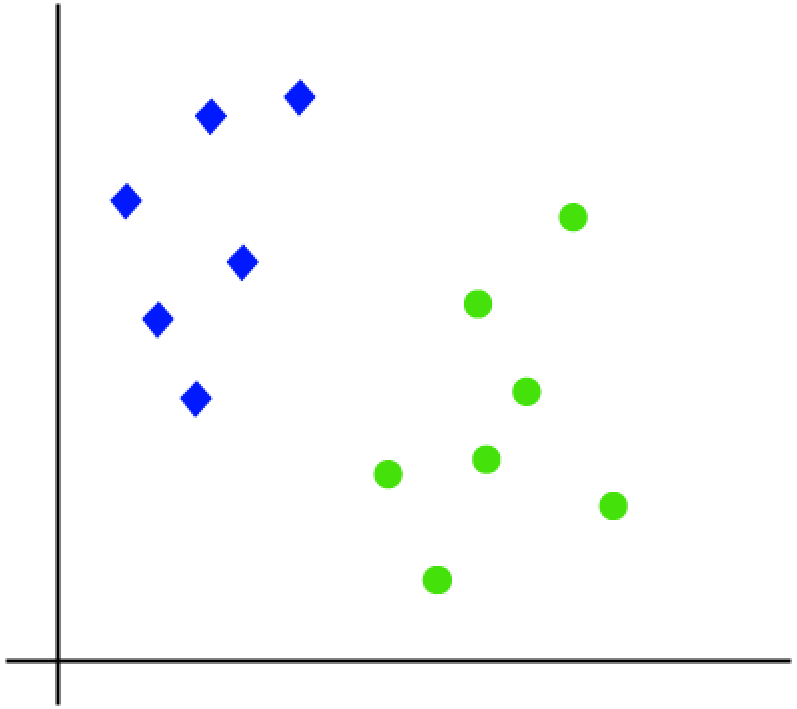


Hyperplanes in 2D and 3D feature space

HOW DOES SVM WORKS?

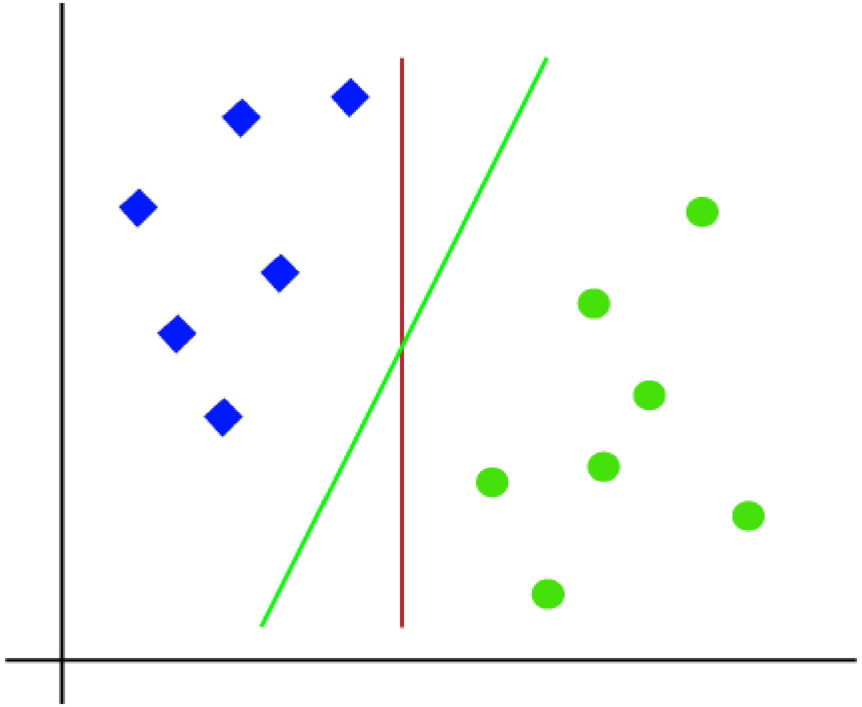
Linear SVM:

Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue.

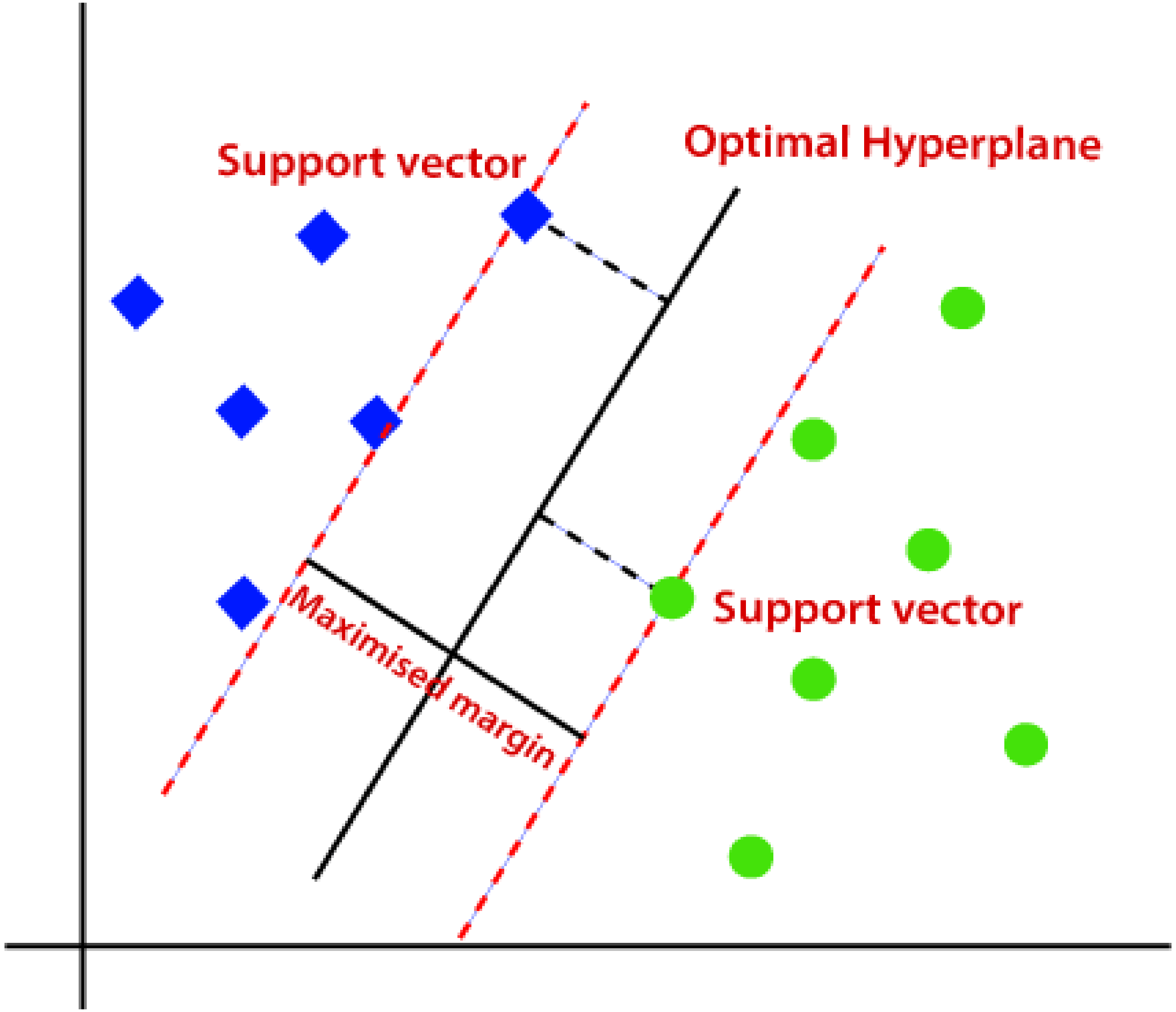


So as it is 2-d space so by just using a straight line, we can easily separate these two classes.

But there can be multiple lines that can separate these classes



- SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**.
- SVM algorithm finds the closest point of the lines from both the classes. These points are called **support vectors**.
- The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin.
- The **hyperplane** with maximum margin is called the **optimal hyperplane**.

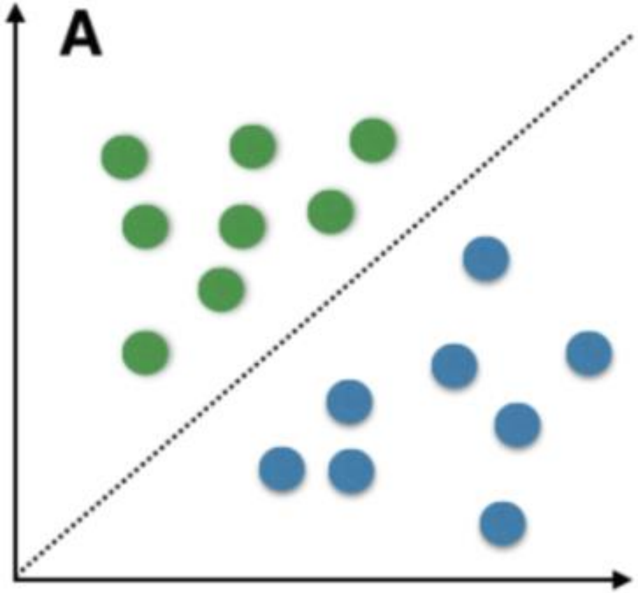

Support vector
Optimal Hyperplane
Support vector
Maximised margin

TYPES OF SVM
2)

SVM can be of two types:

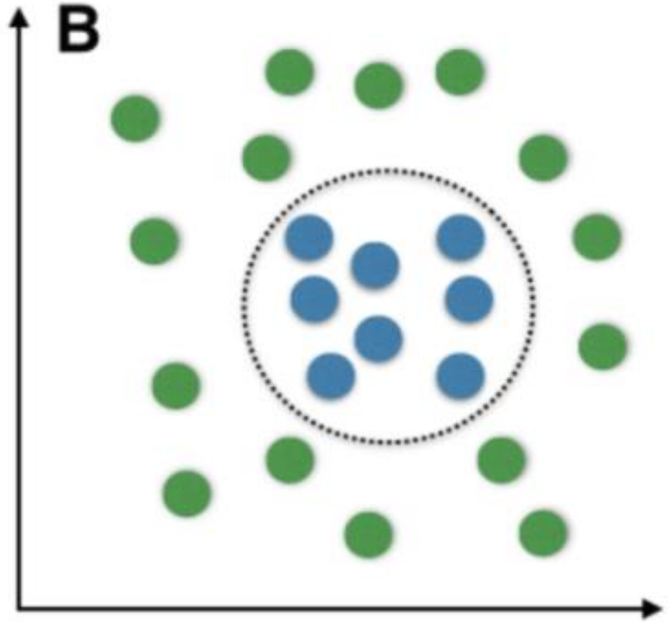
1) Linear SVM:

- In cases where data is linearly separable, a linear SVM can find a straight line (or hyperplane in higher dimensions) that best separates the classes.



2) Non-linear SVM:

- **In real life, most datasets are not linearly separable.**
- **That means, you can't draw a straight line (or plane) to perfectly separate the classes.**
- Use **Non-Linear SVM** that:
- **Maps data to higher-dimensional space** (e.g., from 2D to 3D)
- In this new space, the data may become **linearly separable**



SVM-TYPES

Type	Description
Hard Margin SVM	Assumes data is **linearly separable**. No misclassification allowed.
Soft Margin SVM	Allows some misclassification (good for noisy data). Introduces **slack variables**.
Non-Linear SVM	Uses **kernel trick** to transform data into higher dimensions where it becomes linearly separable.

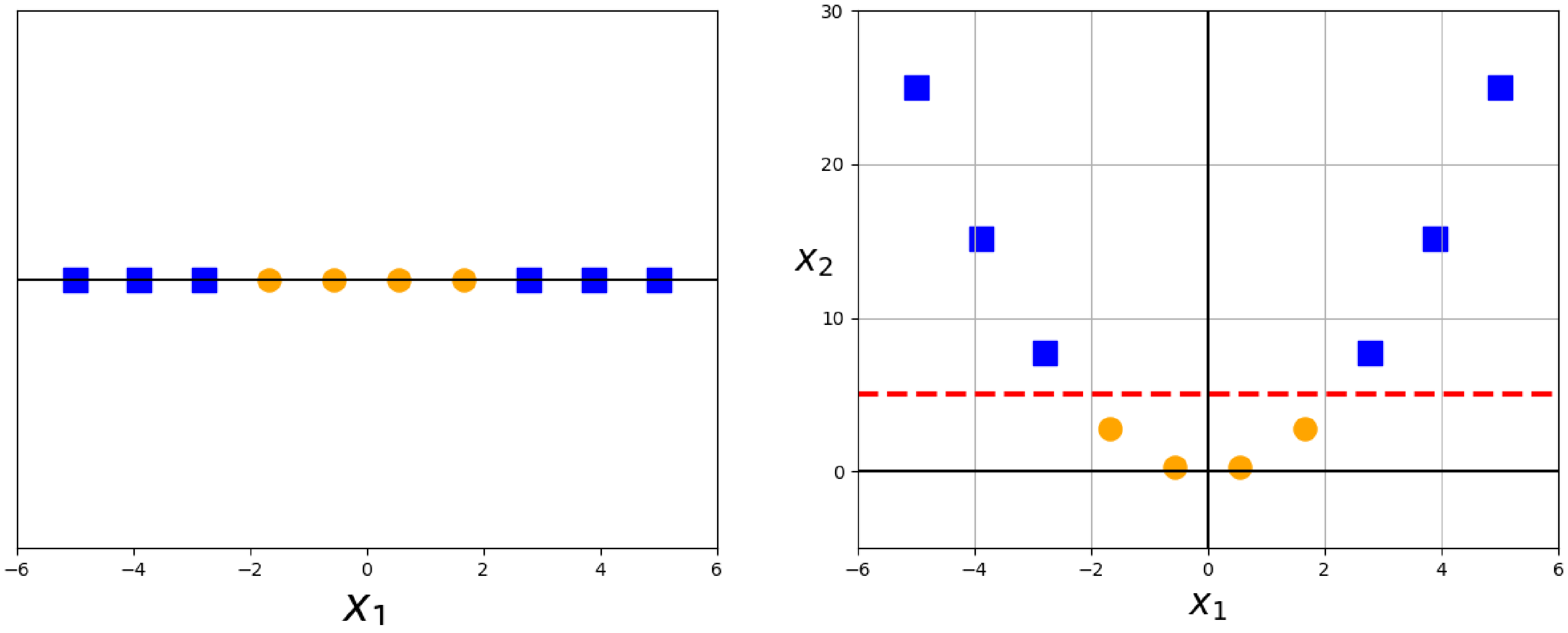
KERNALIZED SVM

- In linear Support Vector Machines, the learning of a hyperplane is done by transforming the existing problem using concepts from linear algebra.
- The kernel trick of the SVM is nothing but functions that take low dimensional input space and transforms it into a higher dimension space.
- it coverts any non-separable problem into a separable one.

SVM kernal visualization

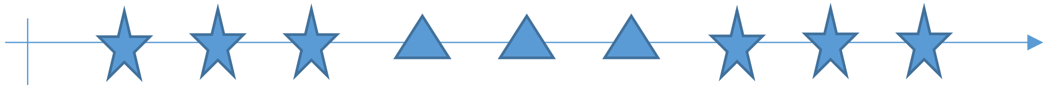
- https://www.youtube.com/watch?v=OdlNM96sHio&list=PPSV

KNOWING THE PROBLEM AND THE SOLUTION



MAPPING TO A HIGHER DIMENSION

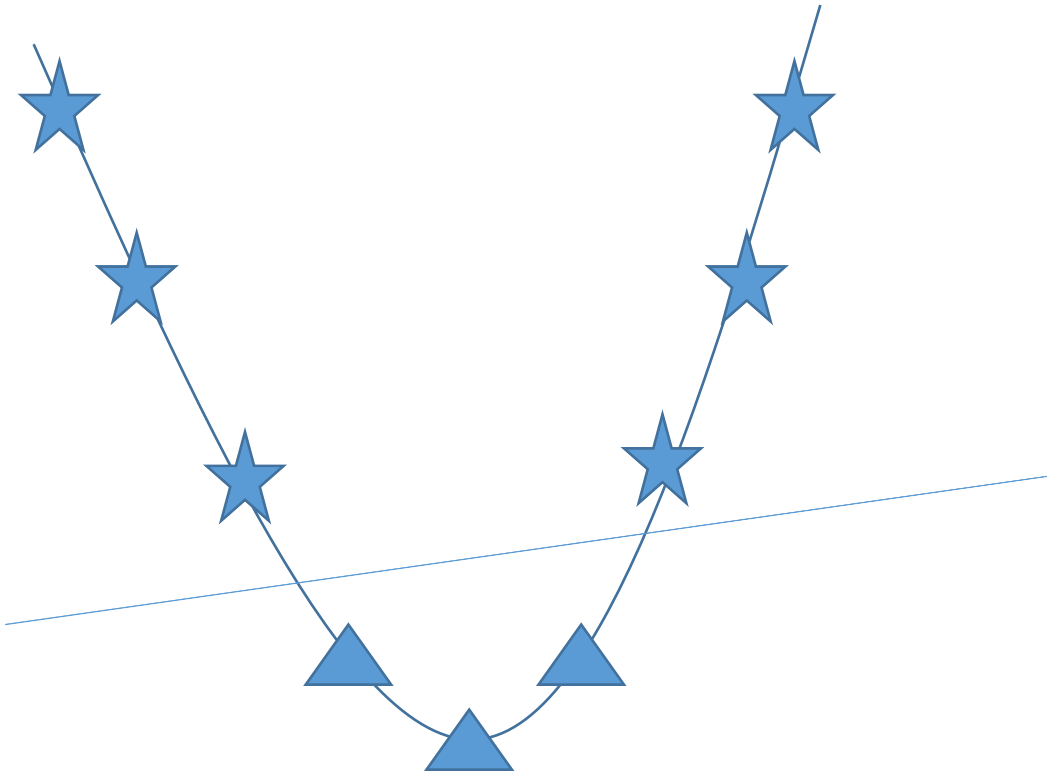
- In this given data set is it possible to separate the data points linearly??(with a straight line like we did before??)
- - No…
- There is a TRICK to do that…
- The basic idea is that when a data set is inseparable in the current dimensions, **add another dimension**, maybe that way the data will be separable.
- Consider a function, f=x-5



FIND THE VALUES

x	x-5	$(x-5)^2$
1	-4	16
2	-3	9
3	-2	4
4	-1	1
5	0	0
6	1	1
7	2	4
8	3	9
9	4	16

NOW THE DATA POINTS ARE LINEARLY SEPARABLE

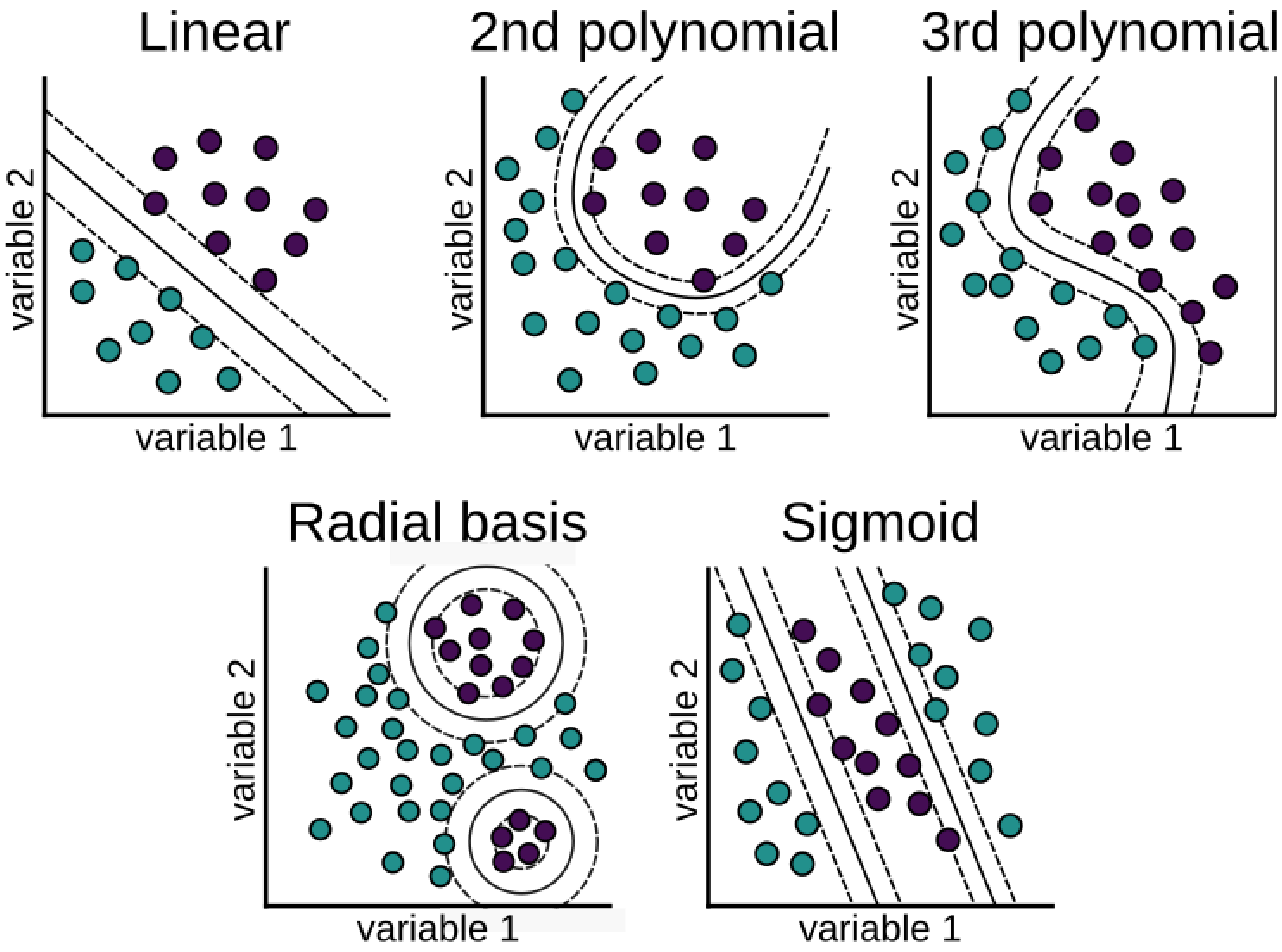


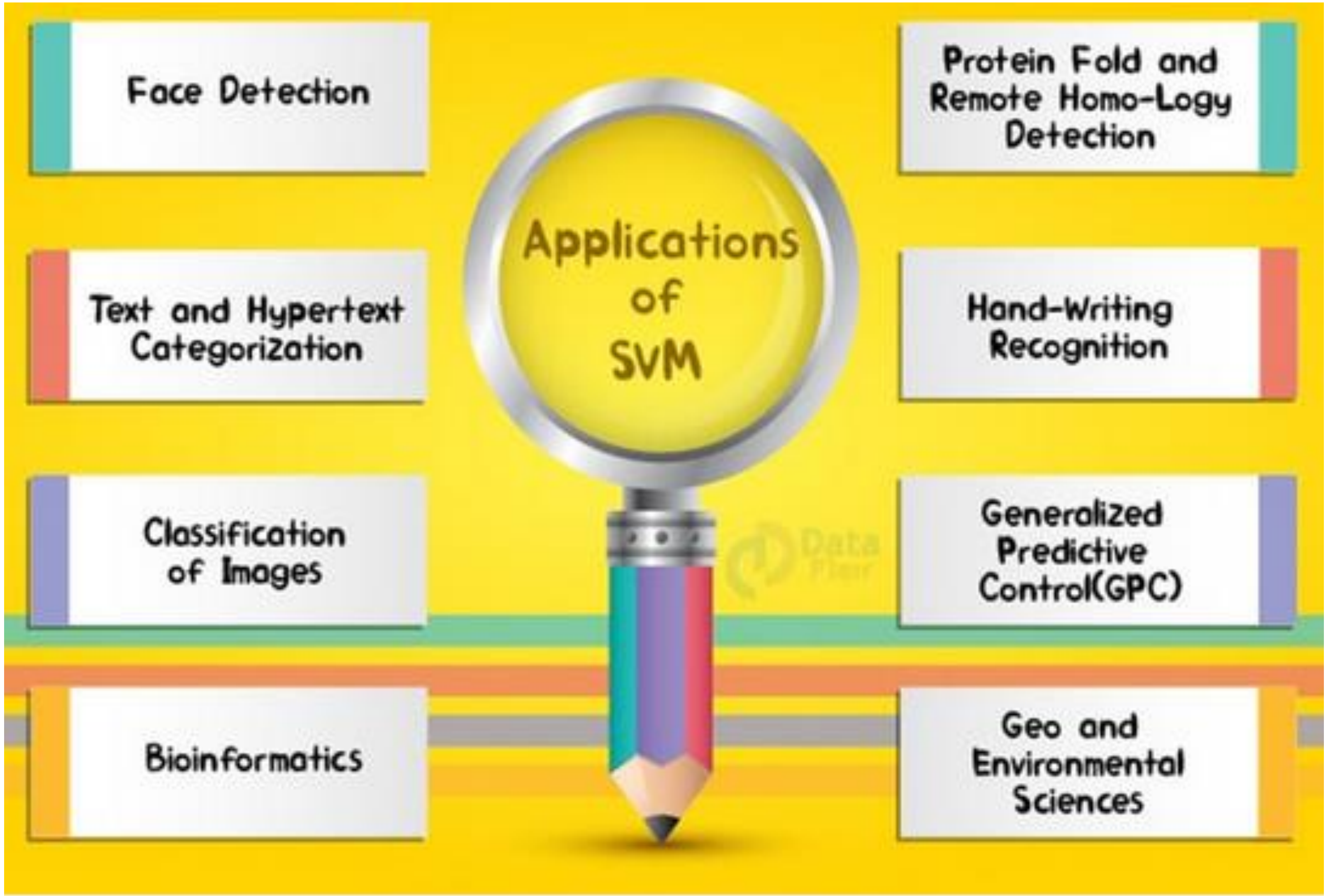
SO.. HOW IS KERNELIZED FUNCTION DEFINED

- The transformations are called **kernels.**
- The **function** of **kernel** is to take data as input and transform it into the required form(For Ex., f=x-5, that we took, to map from lower dimension to higher dimension in order to separate the data points linearly.

VARIOUS KERNEL FUNCTIONS

- Linear kernel(where data points are linearly separable)
- Others are non-linear kernel functions. The examples are as shown beside.




Face Detection
Text and Hypertext Categorization
Classification of Images
Bioinformatics
Applications of SVM
Protein Fold and Remote Homo-Logy Detection
Hand-Writing Recognition
Generalized Predictive Control(GPC)
Geo and Environmental Sciences
Data Flair

Real-Life Applications of SVM (Support Vector Machines)

Pros and Cons associated with SVM

- **Pros:**
  - works really well with a clear margin of separation
  - effective in high dimensional spaces.
  - effective in cases where the number of dimensions is greater than the number of samples.
  - uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

- **Cons:**
  - doesn’t perform well when we have large data set because the required training time is higher
  - doesn’t perform very well, when the data set has more noise i.e. target classes are overlapping
  - SVM doesn’t directly provide probability estimates.

SVM IMPLEMENTATION