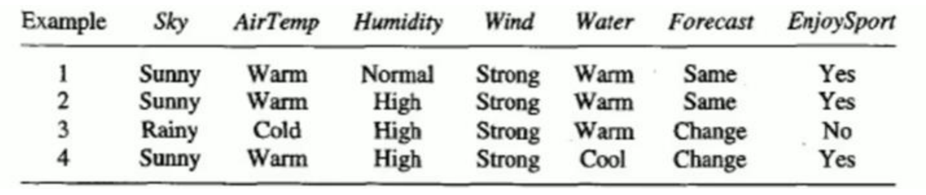
UNIT-III:- Machine Learning: What is Machine Learning, Examples of Various Learning Paradigms, Perspectives and Issues, Version Spaces, Finite and Infinite Hypothesis Spaces.

Supervised Learning: Learning a Class from Examples, Linear, Non-linear, Multi-class and Multi-label classification, Generalization error bounds: VC Dimension

QUESTION BANK

1. Define machine learning and what are the important objectives of machine learning?
2. Briefly explain about Components of learning Process.
3. What is supervised and unsupervised learning? Explain with the examples.
4. Define Ordering of Hypothesis and illustrate it with an example.
5. Illustrate finite and infinite hypothesis spaces with an example.
6. Define Version space and illustrate it with an example.
7. Explain steps of candidate elimination algorithm. Apply the algorithm to obtain the final version space for the given training set.



Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

8. Explain about linear and non-linear classification.
9. Define Multiclass and Multi-label Classification with a neat diagram?
10. Differentiate Traditional programming approach vs Machine learning approach and discuss different perspectives of Machine Learning.

1. Define machine learning and what are the important objectives of machine learning?

Machine Learning Objectives

Machine Learning is a technique used by machines to make predictions based on their information. It is mostly used in data analysis, but it is also used in decision-making using data collected from different sources. The origin of Machine Learning is in the development of computers to automate tasks that humans do. The development of Artificial Intelligence gave rise to the concept of Machine Learning, and Machine Learning develops an algorithm that supports machines in better comprehensive data and making data-based decisions. Machine Learning can be found in different industries such as banks, restaurants, industrial plants, etc.

1. **Understading patterns in data**
   The primary goal of machine learning is to understatnd and identify patterns in data. Machine learninig algorithms analyze large datasets to uncover relationships, trends ana structures that are not immediately obvious.

2. **Making predictions**
   Once patterns are identified, Machine learning aims to make predictions and informed decisions based on the learned data. This involves applying the insights gained from training data to new, unseen data. Common in applications like weather forecasting, stock market analysis, and sales forecasting.

3. **Automating tasks and improve efficiency**
   Another key goal is to automate tasks and improve efficiency. machine learning enables systems to perform tasks automatically by learning from data, reducing the need for human intervent. Examples include image classification, fraud detection, and natural language processing.

4. **Persolizing user Experiences**
   Machine learning also focuses on personalizing user experiences. By understanding individual preferences and behaviors, machine learning alogorithms can tailor recommendations and interactions to meet specific needs. Commonly applied in recommendation systems, targeted advertising, and customer relationship management

5. **Enchancing Decision Processes**
   Machine learning enhances decision-making processes by providing data-driven insights. Decision makers can leverage machine learning models to make more informed and strategic choices. Used in fields like finance, healthcare, and logistics to make data-driven choices.

6. **Optimization**
   Refine processes and systems to maximize desired outcomes, such as minimizing costs or maximizing performance. Examples include resource allocation, production planning, and scheduling.

7. **Classification and Clustering**
   Categorize data into defined classes (classification) or group similar data points without predefined categories (clustering). Applications range from spam detection to market segmentation.

8. **Anomaly Detection**
   Identify unusual patterns or outliers in data, which can signal issues or novel insights. Useful for fraud detection, network security, and quality control.

9. **Continuous Learning and Adaptation**
   Enable systems to adapt over time with new data, improving performance and accuracy.Essential in applications where the data or environment changes, such as autonomous driving and dynamic pricing.

10. **Cognitive Augmentation**
    Assist or augment human abilities, such as analyzing large-scale datasets or interpreting unstructured information.Employed in fields like healthcare, where ML aids doctors in diagnosing diseases, or in customer support through AI chatbots.

These objectives ultimately aim to create intelligent systems that can perform complex tasks independently, provide valuable insights, and support or enhance human decision-making in diverse fields.

2. Briefly explain about Components of learning Process.

Definition of learning:
Definition

- A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience E.

Examples

i) Handwriting recognition learning problem

- Task T: Recognising and classifying handwritten words within images
- Performance P: Percent of words correctly classified
- Training experience E: A dataset of handwritten words with given classifications

ii) A robot driving learning problem

- Task T: Driving on highways using vision sensors
- Performance measure P: Average distance traveled before an error
- Training experience: A sequence of images and steering commands recorded while observing a human driver

iii) A chess learning problem

- Task T: Playing chess
- Performance measure P: Percent of games won against opponents
- Training experience E: Playing practice games against itself

A computer program which learns from experience is called a machine learning program or simply a learning program. Such a program is sometimes also referred to as a learner.

How machines learn

- Basic components of learning process
- The learning process, whether by a human or a machine, can be divided into four components, namely, data storage, abstraction, generalization and evaluation. Figure 1.1 illustrates the various components and the steps involved in the learning process.

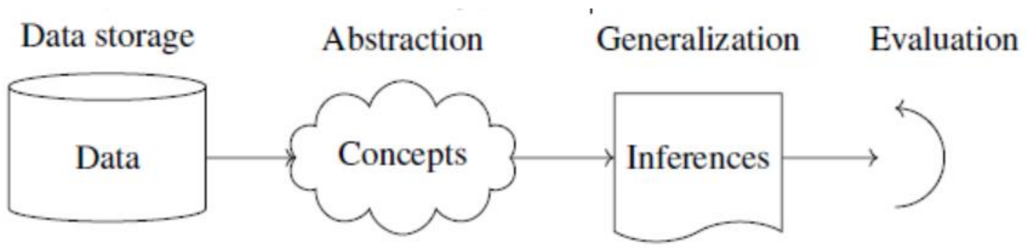


Figure 1.1: Components of learning process

Data storage

- Facilities for storing and retrieving huge amounts of data are an important component of the learning process. Humans and computers alike utilize data storage as a foundation for advanced reasoning.
- In a human being, the data is stored in the brain and data is retrieved using electrochemical signals.
- Computers use hard disk drives, flash memory, random access memory and similar devices to store data and use cables and other technology to retrieve data.

Abstraction

- The second component of the learning process is known as abstraction.
- Abstraction is the process of extracting knowledge about stored data. This involves creating general concepts about the data as a whole. The creation of knowledge involves application of known models and creation of new models.
- The process of fitting a model to a dataset is known as training. When the model has been trained, the data is transformed into an abstract form that summarizes the original information.

Generalization

- The third component of the learning process is known as generalization.
- The term generalization describes the process of turning the knowledge about stored data into a form that can be utilized for future action.
- These actions are to be carried out on tasks that are similar, but not identical, to those what have been seen before.

- In generalization, the goal is to discover those properties of the data that will be most relevant to future tasks.

Evaluation

- Evaluation is the last component of the learning process.
- It is the process of giving feedback to the user to measure the utility of the learned knowledge.
- This feedback is then utilised to effect improvements in the whole learning process.

3. What is supervised and unsupervised learning? Explain with the examples.

Supervised learning

- Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.
- In supervised learning, each example in the training set is a pair consisting of an input object (typically a vector) and an output value.
- A supervised learning algorithm analyzes the training data and produces a function, which can be used for mapping new examples.
- In the optimal case, the function will correctly determine the class labels for unseen instances.
- Both classification and regression problems are supervised learning problems.

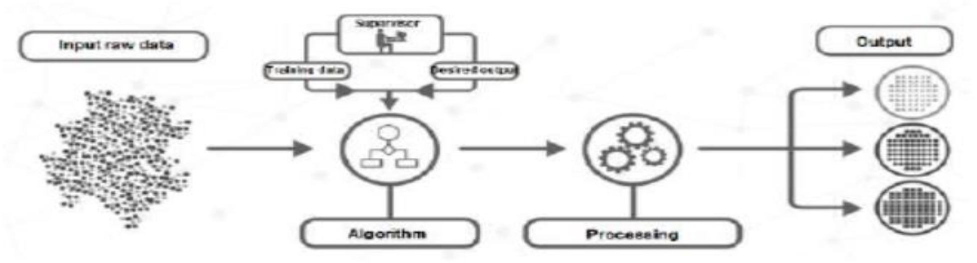
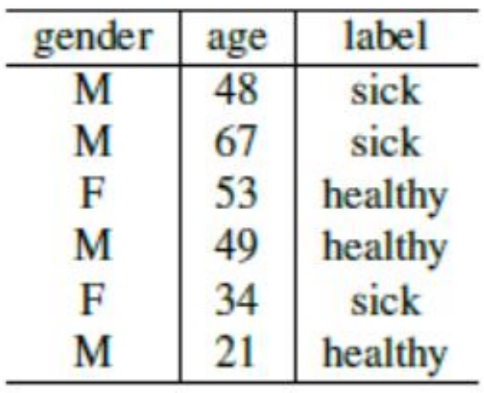


Figure 1.4: Supervised learning

- A “supervised learning” is so called because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process.

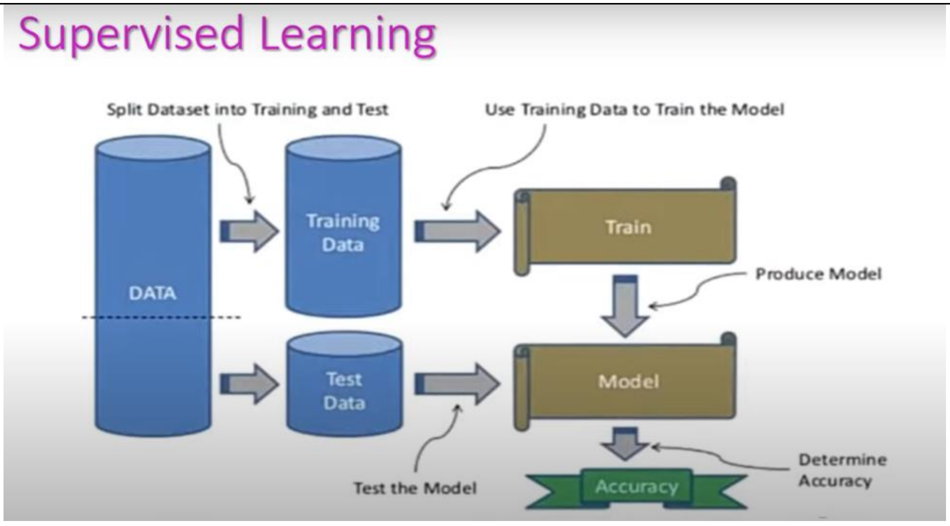
Example

- Consider the following data regarding patients entering a clinic. The data consists of the gender and age of the patients and each patient is labeled as “healthy” or “sick”.



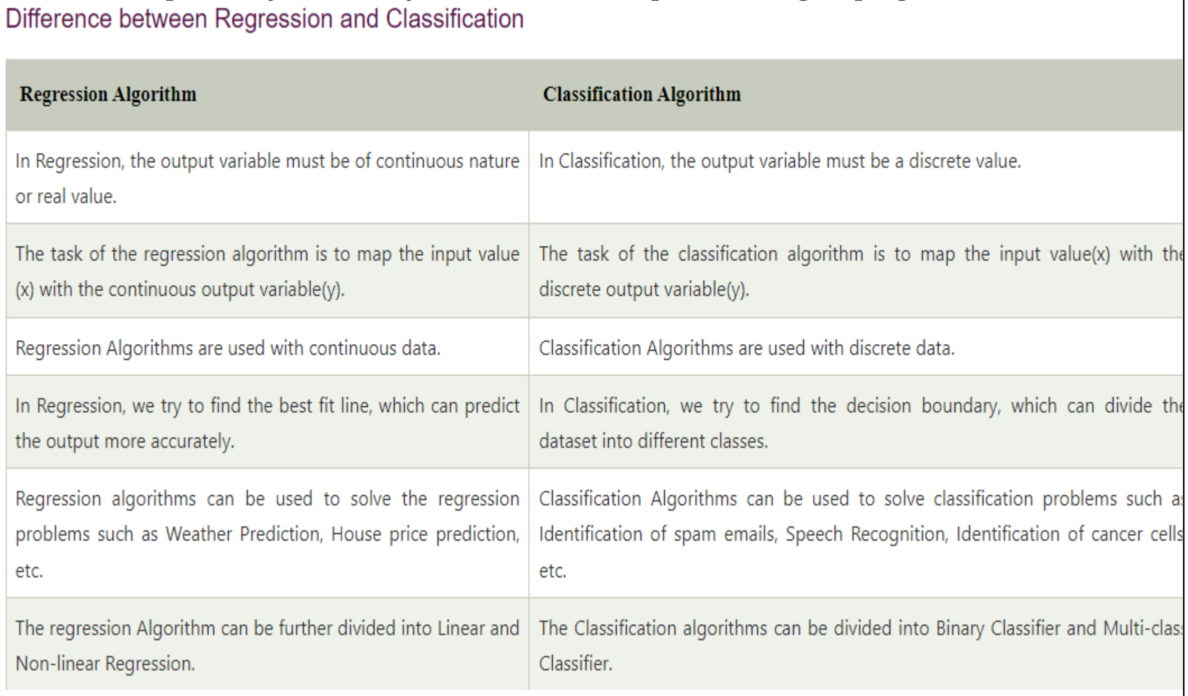
gender	age	label
M	48	sick
M	67	sick
F	53	healthy
M	49	healthy
F	34	sick
M	21	healthy

- Based on this data, when a new patient enters the clinic, how can one predict whether he/she is healthy or sick?



Unsupervised learning

- Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses.
- In unsupervised learning algorithms, a classification or categorization is not included in the observations.
- There are no output values and so there is no estimation of functions. Since the examples given to the learner are unlabeled, the accuracy of the structure that is output by the algorithm cannot be evaluated.
- The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data.



Difference between Regression and Classification

Regression Algorithm	Classification Algorithm
In Regression, the output variable must be of continuous nature or real value.	In Classification, the output variable must be a discrete value.
The task of the regression algorithm is to map the input value (x) with the continuous output variable(y).	The task of the classification algorithm is to map the input value(x) with th discrete output variable(y).
Regression Algorithms are used with continuous data.	Classification Algorithms are used with discrete data.
In Regression, we try to find the best fit line, which can predict the output more accurately.	In Classification, we try to find the decision boundary, which can divide th dataset into different classes.
Regression algorithms can be used to solve the regression problems such as Weather Prediction, House price prediction, etc.	Classification Algorithms can be used to solve classification problems such a Identification of spam emails, Speech Recognition, Identification of cancer cells etc.
The regression Algorithm can be further divided into Linear and Non-linear Regression.	The Classification algorithms can be divided into Binary Classifier and Multi-clas Classifier.

Example

- Consider the following data regarding patients entering a clinic. The data consists of the gender and age of the patients.

gender	age
M	48
M	67
F	53
M	49
F	34
M	21

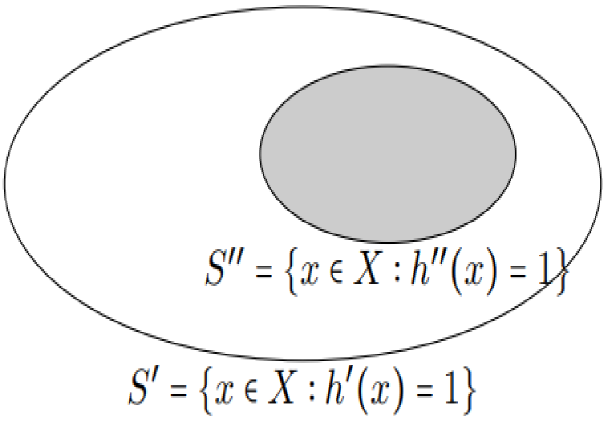
- Based on this data, can we infer anything regarding the patients entering the clinic?

4. Define Ordering of Hypothesis and illustrate it with an example.

Ordering of hypotheses

Definition

- Let X be the set of all possible examples for a binary classification problem and let h ′ and h ″ be two hypotheses for the problem.


$S'' = \{x \in X : h''(x) = 1\}$
$S' = \{x \in X : h'(x) = 1\}$

Hypothesis h' is more general than hypothesis h'' if and only if $S'' \subseteq S'$

1. We say that h ′ is more general than h ″ if and only if for every x ∈ X, if x satisfies h ″ then x satisfies h ′ also; that is, if h ″(x) = 1 then h ′ (x) = 1 also. The relation "is more general than" defines a partial ordering relation in hypothesis space.
2. We say that h′ is more specific than h″, if h″ is more general than h′.
3. We say that h′ is strictly more general than h″ if h′ is more general than h ″ and h″ is not more general than h ′ .
4. We say that h ′ is strictly more specific than h″ if h′ is more specific than h″ and h″ is not more specific than h ′

Example

Consider the hypotheses h′ and h″ . defined in Eqs. Then it is easy to check that if h′(x) = 1 then h″ (x) = 1 also. So, h″ is more general than h′ . • But, h′ is not more general than h″ and so h″ is strictly more general than h′.

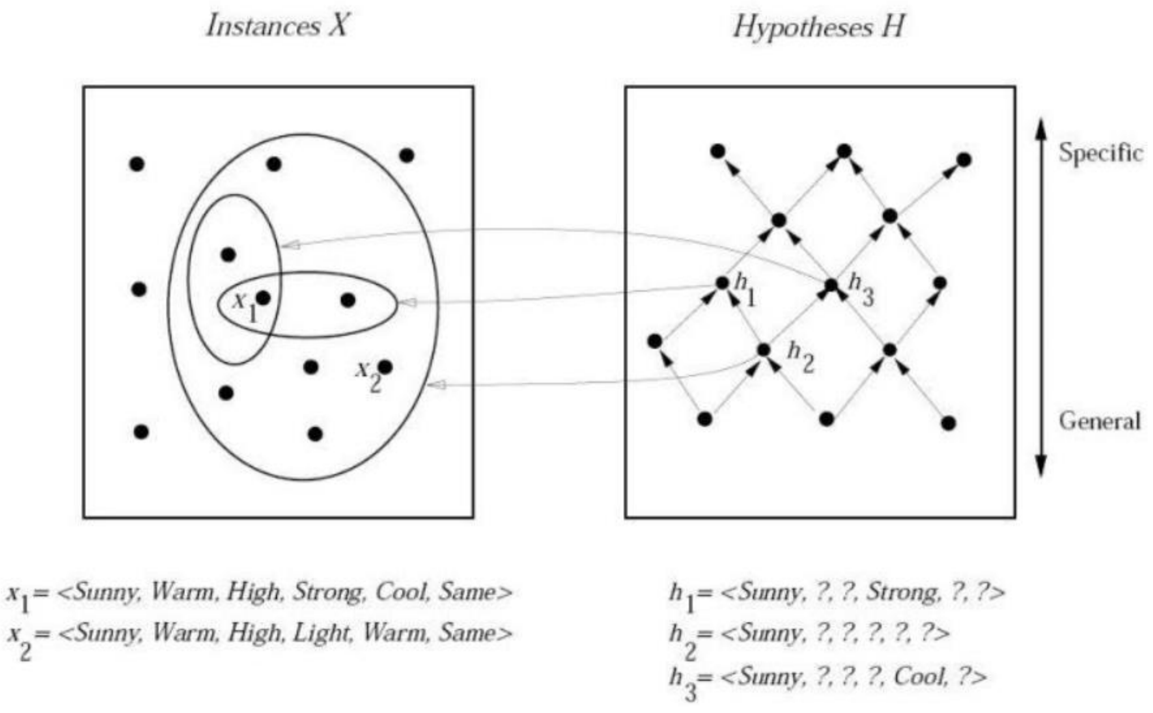
Example

Consider two hypotheses, h1 = (Sunny, ?, ?, Strong, ?, ?) and - h2 = (Sunny, ?, ?, ?, ?, ?)
By looking at the 2 hypothesis representations, the hypothesis h2 has fewer constraints on attributes than hypothesis h1. The sets of instances that are classified positive by hl will be less than sets of instances that are classified positive by h2 as h2 imposes fewer constraints on the instance.

In other words, we can also say that, any instance classified positive by hl will also be classified positive by h2.

Therefore, we say that h2 is more general than hl. In reverse, we can also say that, h1 is more specific than h2.
Lets formalize this concept. Hypothesis h1 and h2 classifies the instance x as positive can written as h1(x) = 1 and h2(x) = 1. Now h2 is more general than h1, so it can be written as if h1(x) = 1 implies h2(x) = 1. In symbols ..

Now lets describe a definition for more-general-than-or-equal-to —
In similar way, more-specific-than, also be defined as below.
By now, we understood the kind of relationship could exist between 2 hypothesis like more-general-than-equal-to or more-general-than.


Instances X
Hypotheses H
x_1
x_2
h_1
h_2
h_3
Specific
General
x_1 = <Sunny, Warm, High, Strong, Cool, Same>
x_2 = <Sunny, Warm, High, Light, Warm, Same>
h_1 = <Sunny, ?, ?, Strong, ?, ?>
h_2 = <Sunny, ?, ?, ?, ?, ?>
h_3 = <Sunny, ?, ?, ?, Cool, ?>

In the figure, the box on the left represents the set X of all instances, the box on the right the set H of all hypotheses
Each hypothesis corresponds to some subset of X, that is, the subset of instances that it classifies positive. In the figure, there are 2 instances x1 and x2, and 3 hypothesis h1, h2 and h2.

5. Illustrate finite and infinite hypothesis spaces with an example.

Hypothesis

- In a binary classification problem, a hypothesis is a statement or a proposition purporting to explain a given set of facts or observations.

Hypothesis space

- The hypothesis space for a binary classification problem is a set of hypotheses for the problem that might possibly be returned by it.

Consistency and satisfying

- Let x be an example in a binary classification problem and let c(x) denote the class label assigned to x (c(x) is 1 or 0).
- Let D be a set of training examples for the problem. Let h be a hypothesis for the problem and h(x) be the class label assigned to x by the hypothesis h.
- (a) We say that the hypothesis h is consistent with the set of training examples D if h(x) = c(x) for all x > D.
- (b) We say that an example x satisfies the hypothesis h if h(x) = 1.

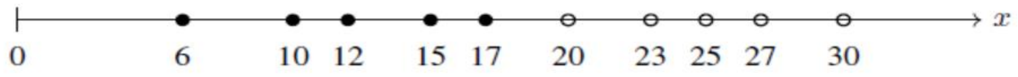
Examples

- Consider the set of observations of a variable x with the associated class labels given in Table

x	27	15	23	20	25	17	12	30	6	10
Class	1	0	1	1	1	0	0	1	0	0

Table 2.1: Sample data to illustrate the concept of hypotheses

Figure 2.1 shows the data plotted on the x-axis.



- Data in Table with hollow dots representing positive examples and solid dots representing negative examples
- Looking at Figure 2.1, it appears that the class labeling has been done based on the following rule.

  h' : IF x ≥ 20 THEN "1" ELSE "0".
- It is not enough that the hypothesis explains the given data; it must also predict correctly the class label of future observations.
- So we consider a set of such hypotheses and choose the "best" one.
- The set of hypotheses can be defined using a parameter, say m, as given below:

  h_m : IF x ≥ m THEN "1" ELSE "0".
- The set of all hypotheses obtained by assigning different values to m constitutes the hypothesis space H; that is,

  H = {h_m : m is a real number}:
- For the same data, we can have different hypothesis spaces. For example, for the data in Table, we may also consider the hypothesis space defined by the following proposition:

  h'_m : IF x ≤ m THEN "0" ELSE "1".

- Consider the problem of assigning the label “family car” or “not family car” to cars.
- For convenience, we shall replace the label “family car” by “1” and “not family car” by “0”.
- Suppose we choose the features “price (’000 $)” and “power (hp)” as the input representation for the problem.
- Further, suppose that there is some reason to believe that for a car to be a family car, its price and power should be in certain ranges.
- This supposition can be formulated in the form of the following proposition:

$$\text{IF } (p_1 < \text{price} < p_2) \text{ AND } (e_1 < \text{power} < e_2) \text{ THEN “1” ELSE ”0”}$$

for suitable values of p1, p2, e1 and e2

- Since a solution to the problem is a proposition of the form Eq.(2.5) with specific values for p1, p2, e1 and e2.
- The hypothesis space for the problem is the set of all such propositions obtained by assigning all possible values for p1, p2, e1 and e2.

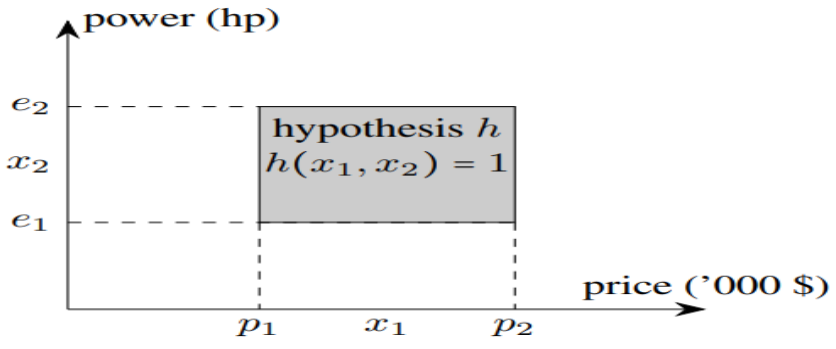


Figure 2.2: An example hypothesis defined by Eq.

- It is interesting to observe that the set of points in the power–price plane which satisfies the condition

$$(p_1 < \text{price} < p_2) \text{ AND } (e_1 < \text{power} < e_2)$$

defines a rectangular region (minus the boundary) in the price–power space as shown in Figure.

- The sides of this rectangular region are parallel to the coordinate axes. Such a rectangle is called an axis-aligned rectangle.
- If h is the hypothesis and (x1, x2) is any point in the price–power plane, then h(x1, x2) = 1 if and only if (x1, x2) is within the rectangular region.

The two types of hypothesis classes ("finite hypothesis class" and "infinite hypothesis class") are used in machine learning or data science:

1. Finite Hypothesis Class: In this case, the hypothesis class (often denoted as H) is assumed to be finite. A hypothesis class represents the set of possible models or functions that a machine learning algorithm can choose from when trying to learn a

relationship between input data and output. When the hypothesis class is finite, it means that there are a limited number of potential models or functions to choose from. Conceptually, a finite hypothesis class H can be represented as a set of distinct hypotheses or models. Suppose H has "m" hypotheses. You can represent it as: H = $\{h_1, h_2, ..., h_m\}$.

2. Infinite Hypothesis Class: Conversely, an "infinite hypothesis class" implies that the set of potential models or functions is not limited and can be infinite. This can be the case when, for example, the algorithm can consider an unbounded number of different models or functions to fit the data. An infinite hypothesis class H typically doesn't have a finite, enumerable list of hypotheses. It can include an infinite number of possible hypotheses. There is no specific equation to represent this class; it's described by the fact that it's unbounded.

Finite hypothesis class	**Infinite hypothesis class**
Number of classes are $\|H\| = R$	Number of classes are infinite
Limited Options: Finite Set: The hypothesis class consists of a finite, countable number of hypotheses. Limited Expressiveness: The model has a restricted capacity to represent complex relationships in the data.	Unlimited Options: nfinite Set: The hypothesis class contains an infinite number of possible hypotheses.
Easier to Manage: Handling a finite set of hypotheses is computationally more feasible and often simpler. Less Computational Cost: Training and evaluating models with a finite hypothesis class may be less computationally intensive.	Increased Complexity: Dealing with an infinite hypothesis class can introduce computational challenges in terms of training and evaluation. Higher Computational Cost: Training and evaluating models with an infinite hypothesis class may require more resources.
More Conclusive Results: Theoretical analysis, such as deriving generalization bounds, may be more straightforward due to the finite nature of the hypothesis class. Well-Defined Bounds: It may be easier to establish bounds on the algorithm's performance.	Complex Analysis: Theoretical analysis becomes more intricate due to the infinite nature of the hypothesis space. Generalization Bounds Challenges: Establishing precise generalization bounds may be more challenging in the presence of an infinite hypothesis class.
Restricted Representational Power: A finite hypothesis class is limited in the types of functions or relationships it can represent. There are only a finite number of hypotheses to choose from, which may not be sufficient to capture complex patterns in the data. Potential Underfitting: Due to its limited expressiveness, a finite hypothesis class may struggle to fit the training data well, leading to underfitting. Underfitting occurs when the model is too simplistic	Adaptability: An infinite hypothesis class allows for greater flexibility in adapting to complex and nuanced patterns in the data. Risk of Overfitting: There's a risk of overfitting as the model may have the capacity to fit the training data too closely.

6. Define Version space and illustrate it with an example.

Version space

Definition

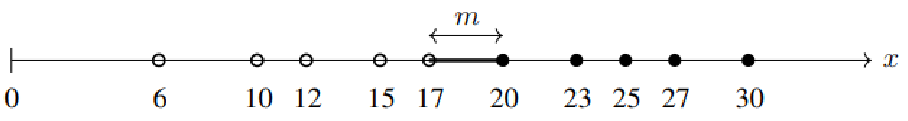
- Consider a binary classification problem. Let D be a set of training examples and H a hypothesis space for the problem.
- The version space for the problem with respect to the set D and the space H is the set of hypotheses from H consistent with D; that is, it is the set

$$\mathrm{VS}_{D,H} = \{h \in H \mid : h(x) = c(x) \text{ for all } x \in D\}.$$

- The version space $VS_{H,D}$ is the subset of the hypothesis from H consistent with the training example in D.
- In general terms, say that $VS_{H,D}$ contains all hypothesis which are consistent with the training examples.

Example

- Consider the data D given in Table and the hypothesis space defined by Equations



$$\mathrm{VS}_{D,H} = \{h_m : 17 < m \le 20\}.$$

Example

- Consider the problem of assigning the label “family car” (indicated by “1”) or “not family car” (indicated by “0”) to cars.
- Given the following examples for the problem and assuming that the hypothesis space is as defined by Eq.

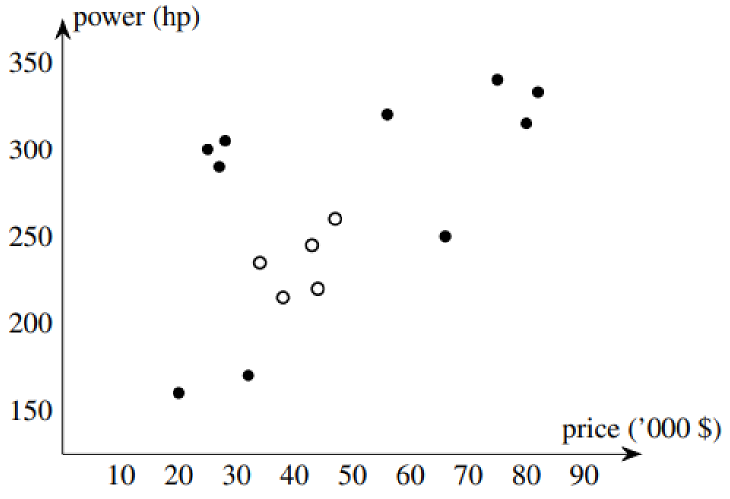
IF $(p_1 < \text{price} < p_2)$ AND $(e_1 < \text{power} < e_2)$ THEN “1” ELSE ”0”

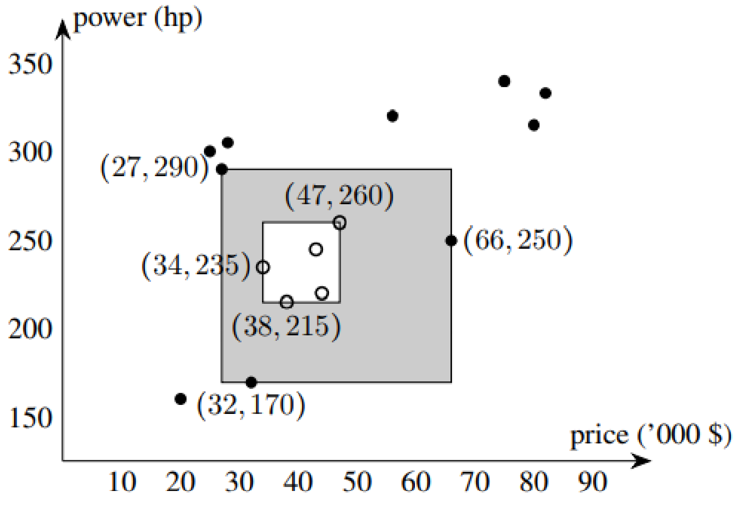
the version space for the problem.

x_1: Price in ’000 (\$)	32	82	44	34	43	80	38
x_2: Power (hp)	170	333	220	235	245	315	215
Class	0	0	1	1	1	0	1

x_1	47	27	56	28	20	25	66	75
x_2	260	290	320	305	160	300	250	340
Class	1	0	0	0	0	0	0	0

- A hypothesis as given by above equation with specific values for the parameters p1, p2, e1 and e2 specifies an axis-aligned rectangle as shown in Figure.
- So the hypothesis space for the problem can be thought as the set of axis-aligned rectangles in the price-power plane





- The version space consists of hypotheses corresponding to axis-aligned rectangles contained in the shaded region.
- The version space consists of all hypotheses specified by axis-aligned rectangles contained in the shaded region.
- The inner rectangle is defined by

$$(34 < \text{price} < 47) \text{ AND } (215 < \text{power} < 260)$$

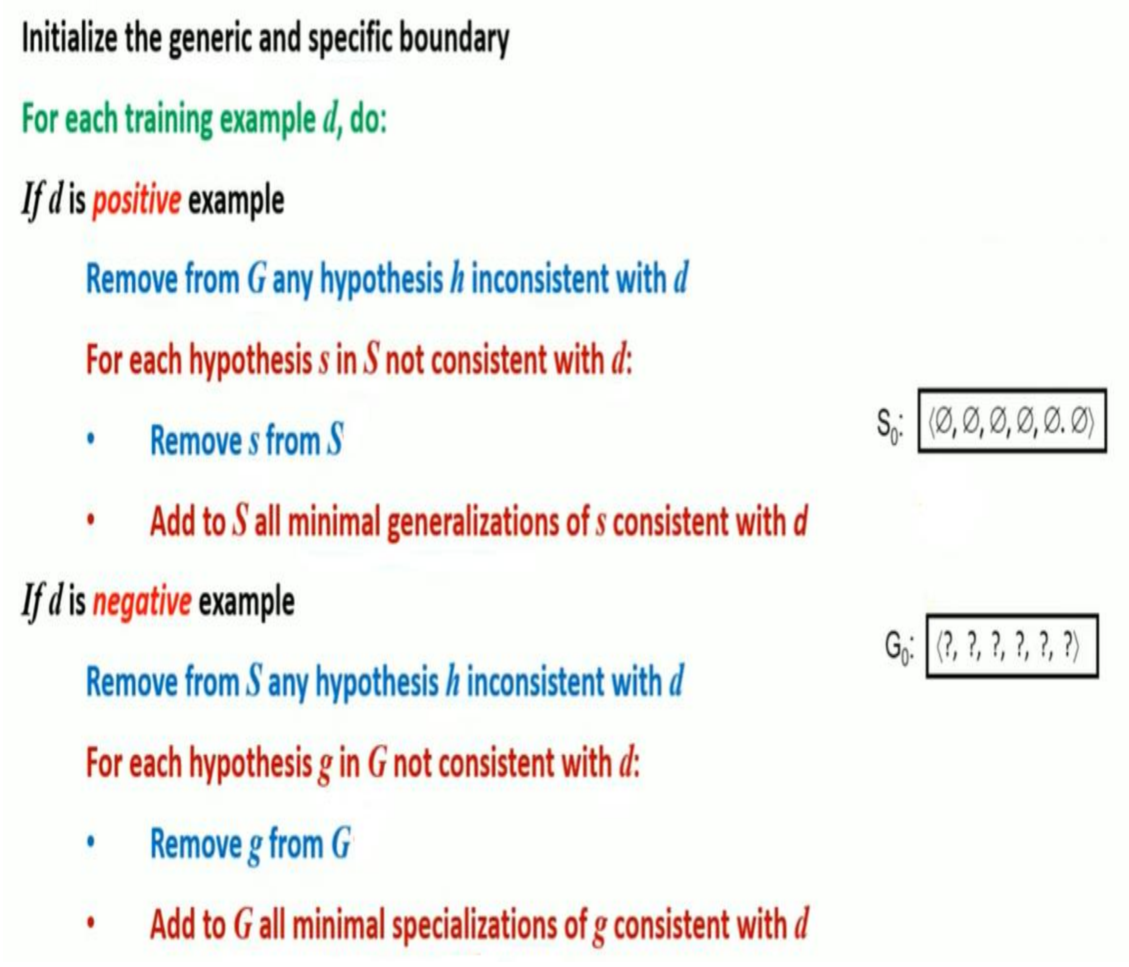
- The outer rectangle is defined by

$$(27 < \text{price} < 66) \text{ AND } (170 < \text{power} < 290).$$

7. Explain steps of candidate elimination algorithm. Apply the algorithm to obtain the final version space for the given training set.

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

The candidate-elimination algorithm manipulates the boundary-set representation of a version space to create boundary sets that represent a new version space consistent with all the previous instances plus the new one.

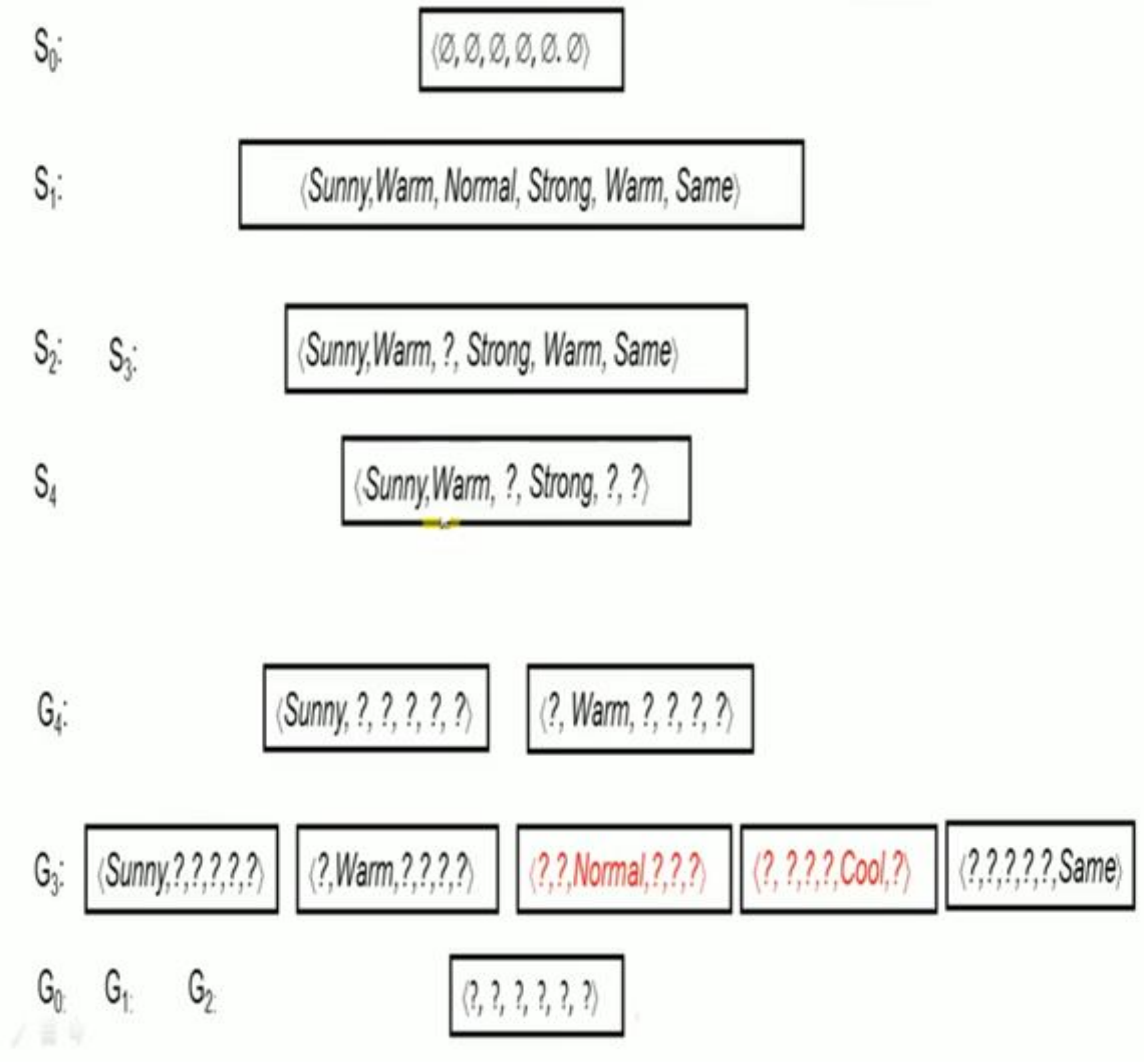


```
Initialize the generic and specific boundary
For each training example d, do:
If d is positive example
    Remove from G any hypothesis h inconsistent with d
    For each hypothesis s in S not consistent with d:
        •   Remove s from S
        •   Add to S all minimal generalizations of s consistent with d
If d is negative example
    Remove from S any hypothesis h inconsistent with d
    For each hypothesis g in G not consistent with d:
        •   Remove g from G
        •   Add to G all minimal specializations of g consistent with d
```

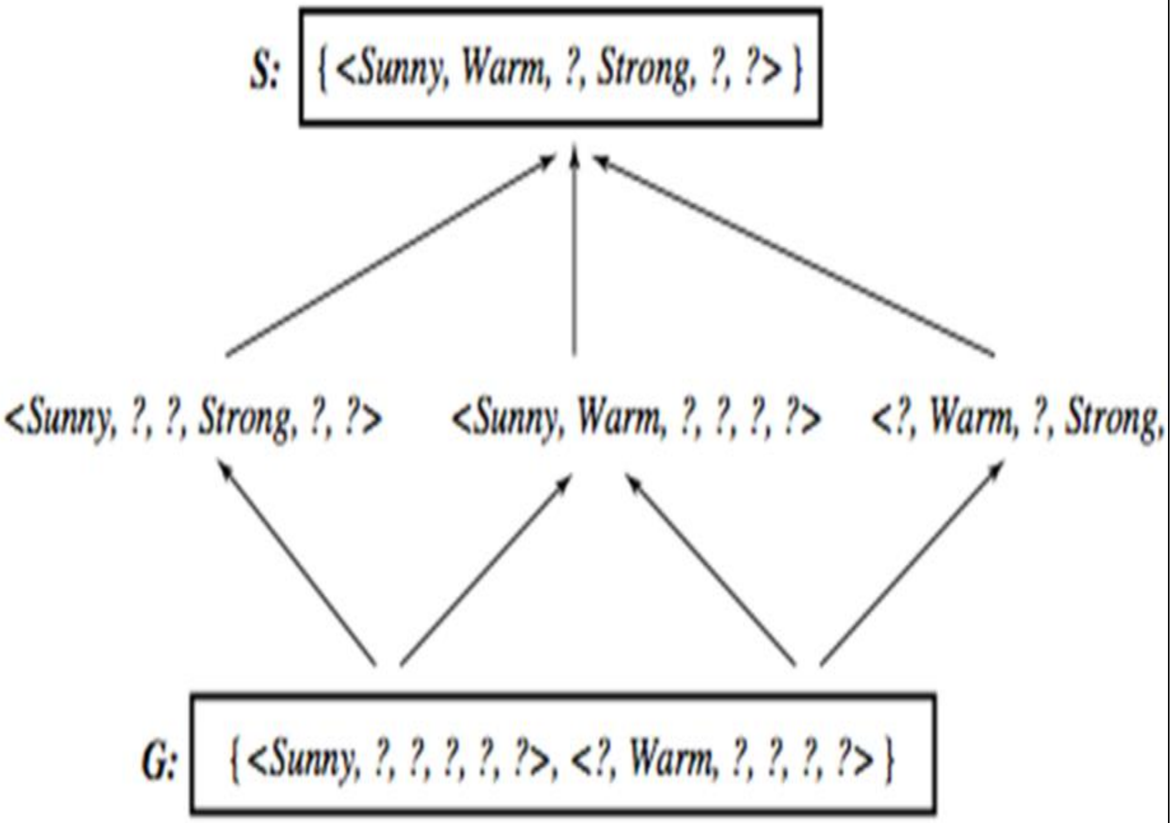
S0: **(ø, ø, ø, ø, ø, ø)** Most Specific Boundary

G0: (?, ?, ?, ?, ?, ?) Most Generic Boundary
The first example is positive, the hypothesis at the specific boundary is inconsistent, hence we extend the specific boundary, and the hypothesis at the generic boundary is consistent hence we retain it.
S1: (*Sunny*,*Warm*, *Normal*, *Strong*, *Warm*, *Same*)
G1: (?, ?, ?, ?, ?, ?)
The second example in positive, again the hypothesis at the specific boundary is inconsistent, hence we extend the specific boundary, and the hypothesis at the generic boundary is consistent hence we retain it.
S2: (*Sunny*,*Warm*, *?*, *Strong*, *Warm*, *Same*)
G2: (?, ?, ?, ?, ?, ?)
The third example is negative, the hypothesis at the specific boundary is consistent, hence we retain it, and hypothesis at the generic boundary is inconsistent hence we write all consistent hypotheses by removing one "?" (question mark) at time.
S3: (*Sunny*,*Warm*, *?*, *Strong*, *Warm*, *Same*)
G3: (*Sunny*,*?*,*?*,*?*,*?*,*?*) (*?*,*Warm*,*?*,*?*,*?*,*?*) (*?*,*?*,*?*,*?*,*?*,*Same*)
The fourth example is positive, the hypothesis at the specific boundary is inconsistent, hence we extend the specific boundary, and the consistent hypothesis at the generic boundary are retained.

S4: (*Sunny*, *Warm*, *?*, *Strong*, *?*, *?*)
G4: (*Sunny*,*?*,*?*,*?*,*?*,*?*) (*?*,*Warm*,*?*,*?*,*?*,*?*)


S0: ⟨Ø, Ø, Ø, Ø, Ø. Ø⟩
S1: ⟨Sunny,Warm, Normal, Strong, Warm, Same⟩
S2: S3: ⟨Sunny,Warm, ?, Strong, Warm, Same⟩
S4 ⟨Sunny,Warm, ?, Strong, ?, ?⟩
G4: ⟨Sunny, ?, ?, ?, ?, ?⟩ ⟨?, Warm, ?, ?, ?, ?⟩
G3: ⟨Sunny,?,?,?,?,?⟩ ⟨?,Warm,?,?,?,?⟩ ⟨?,?,Normal,?,?,?⟩ ⟨?, ?,?,?,Cool,?⟩ ⟨?,?,?,?,?,Same⟩
G0: G1: G2: ⟨?, ?, ?, ?, ?, ?⟩

Learned Version Space by Candidate Elimination Algorithm for given data set is:


S: { <Sunny, Warm, ?, Strong, ?, ?> }
<Sunny, ?, ?, Strong, ?, ?>
<Sunny, Warm, ?, ?, ?, ?>
<?, Warm, ?, Strong,
G: { <Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?> }

8. Explain about linear and non-linear classification.

LINEAR CLASSIFICATION

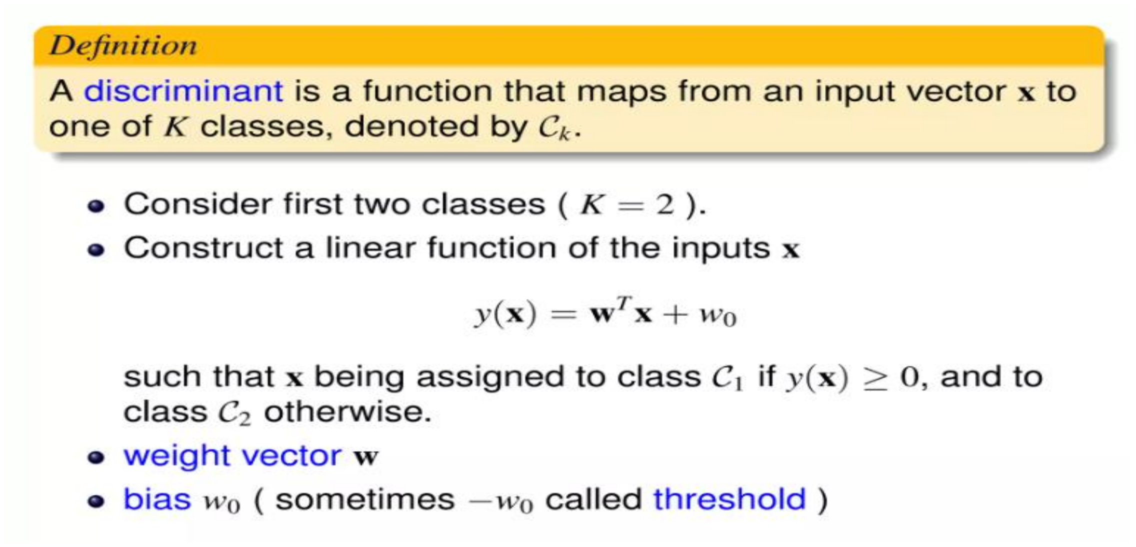
A **classification algorithm** (Classifier) that makes its classification based on a linear predictor function combining a set of weights with the feature vector.

In linear classification the main assumption is the **Boundary hyperplane / decision hyperplane** is always linear.

Discrimating function is used classifies unknown individuals and the probability of their classification into a certain group.

(i) **Discriminant Function**

- Linear Discriminant Analysis **(LDA)** is the most commonly used dimensionality reduction technique in supervised learning. Basically, it is a preprocessing step for pattern classification and machine learning applications. LDA is a powerful algorithm that can be used to determine the best separation between two or more classes.
- LDA is a **supervised learning algorithm**, which means that it requires a labelled training set of data points in order to learn the linear discriminant function.
- The main purpose of LDA is to find the line or plane that best separates data points belonging to different classes. The key idea behind LDA is that the decision boundary should be chosen such that it maximizes the distance between the means of the two classes while simultaneously minimizing the variance within each class's data or within-class scatter. This criterion is known as the Fisher criterion.
- LDA is one of the most widely used machine learning algorithms due to its accuracy and flexibility. LDA can be used for a variety of tasks such as classification, dimensionality reduction, and feature selection.



Definition

A discriminant is a function that maps from an input vector $\mathbf{x}$ to one of K classes, denoted by $\mathcal{C}_k$.

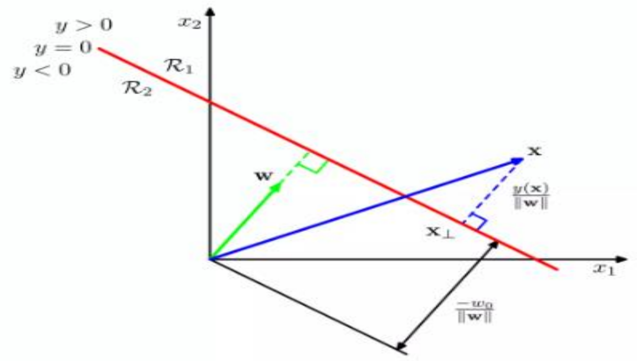
- Consider first two classes ($K = 2$).
- Construct a linear function of the inputs $\mathbf{x}$

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

such that $\mathbf{x}$ being assigned to class $\mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$, and to class $\mathcal{C}_2$ otherwise.

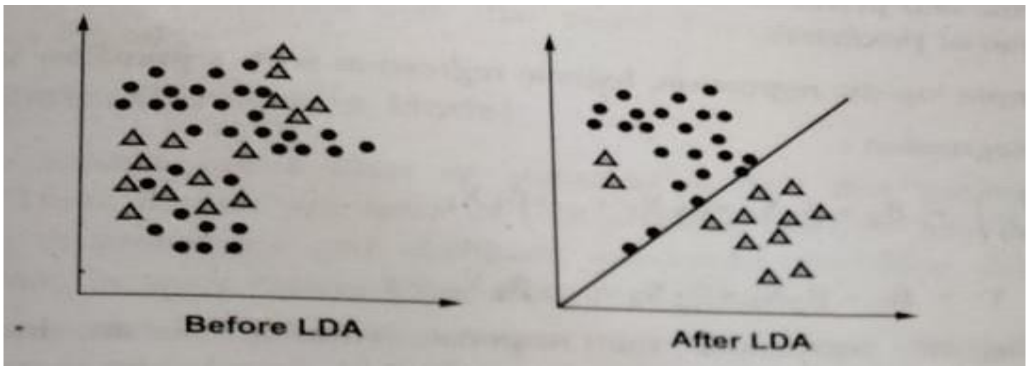
- weight vector $\mathbf{w}$
- bias w_0 (sometimes $-w_0$ called threshold)

If Y(x) >0 , means it belongs to classA
If Y(x) <0 , means it belongs to classB
If Y(x) == 0 , means it represents the boundary hyperplane.



- ➢ Linear classifiers can represent a lot of things, but they can't represent everything. The classic example of what they can't represent is the XOR function.
- ➢ Suppose we have two classes and we need to classify them efficiently, then using LDA, classes are divided as follows

LDA algorithm works based on the following steps:



a) The first step is to calculate the means and standard deviation of each feature.
b) Within class scatter matrix and between class scatter matrix is calculated
c) These matrices are then used to calculate the eigenvectors and eigenvalues.
d) LDA chooses the k eigenvectors with the largest eigenvalues to form a transformation matrix.
e) LDA uses this transformation matrix to transform the data into a new space with k dimensions.

f) Once the transformation matrix transforms the data into new space with k dimensions, LDA can then be used for classification or dimensionality reduction

- Benefits of using LDA:

  a) LDA is used for classification problems.
  b) LDA is a powerful tool for dimensionality reduction.
  c) LDA is not susceptible to the "curse of dimensionality" like many other machine learning algorithms.

NOTE:- If the dependent variable is continous then we have to use Linear regression If the dependent variable is categorical/binary then we have to use Linear regression

(ii) Logistic Regression

- Logistic regression is a form of regression analysis in which the outcome variable is binary or dichotomous (categorical). A statistical method used to model dichotomous or binary outcomes using predictor variables.
- **Logistic component**: Instead of modelling the outcome, Y, directly, the method models the log odds (Y) using the logistic function.

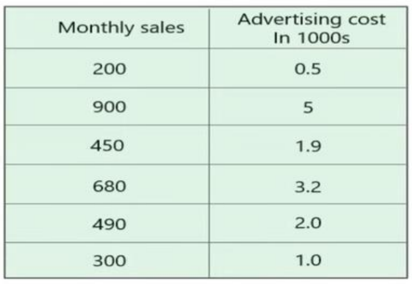
With logistic regression, the response variable is an indicator of some characteristic, that is, a 0/1 variable. Logistic regression is used to determine whether other measurements are related to the presence of some characteristic, for example, whether certain blood measures are predictive of having a disease.

The linear model assumes that the probability p is a linear function of the regressors, while the logistic model assumes that the natural log of the odds p/(1-p) is a linear function of the regressors.

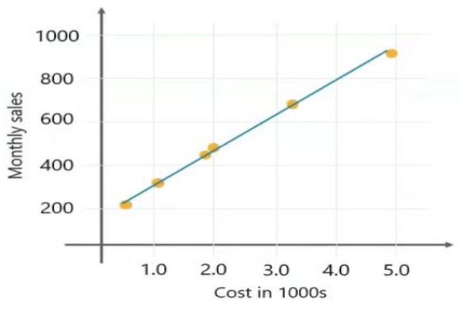
$$\ln[p/(1-p)] = b_0 + b_1X_1 + b_2X_2 + \ldots + b_kX_k$$

Example:-

To forecast monthly sales by studying the relationship between the monthly e-commerce sales and the online advertising costs.



Monthly sales	Advertising cost In 1000s
200	0.5
900	5
450	1.9
680	3.2
490	2.0
300	1.0



NONLINEAR CLASSIFIERS:

Linear classifiers work by finding a straight line, plane, or hyperplane that separates the different classes in the feature space.

They are relatively simple, easy to interpret, and fast to train. Some common linear classifiers include **logistic regression** and linear support vector machines.

Non-linear classifiers, on the other hand, can find more complex decision boundaries to separate the classes. They can capture intricate patterns and relationships within the data that linear classifiers might miss.

Non-linear classifiers include decision trees, neural networks, kernel support vector machines, and many others.

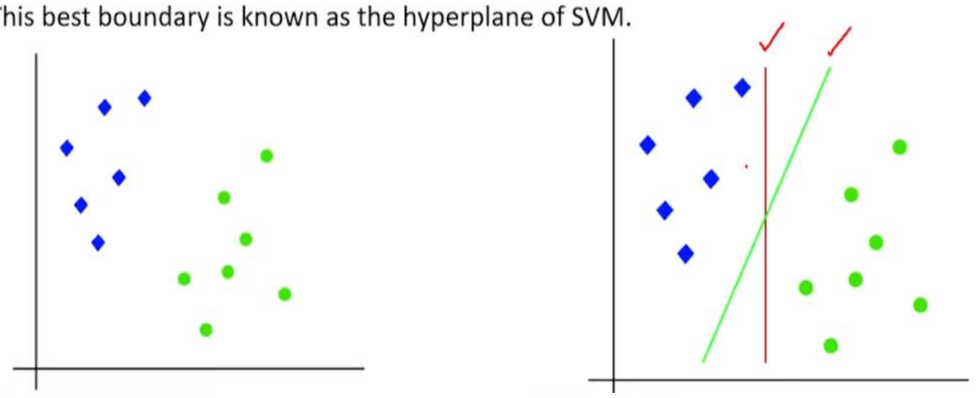
Support Vector Machines (SVM)

Svm classifier mostly used in addressing multi-classification problems. Multi-classification

problem means having more that 2 target classes to predict.
In the first example of predicting the fruit type. The target class will have many fruits like apple, mango, orange, banana, etc. This is same with the other two examples in predicting. The problem of the new article, the target class having different topics like sport, movie, tech news ..etc

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.
- However, primarily, it is used for Classification problems in Machine Learning.
- 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.

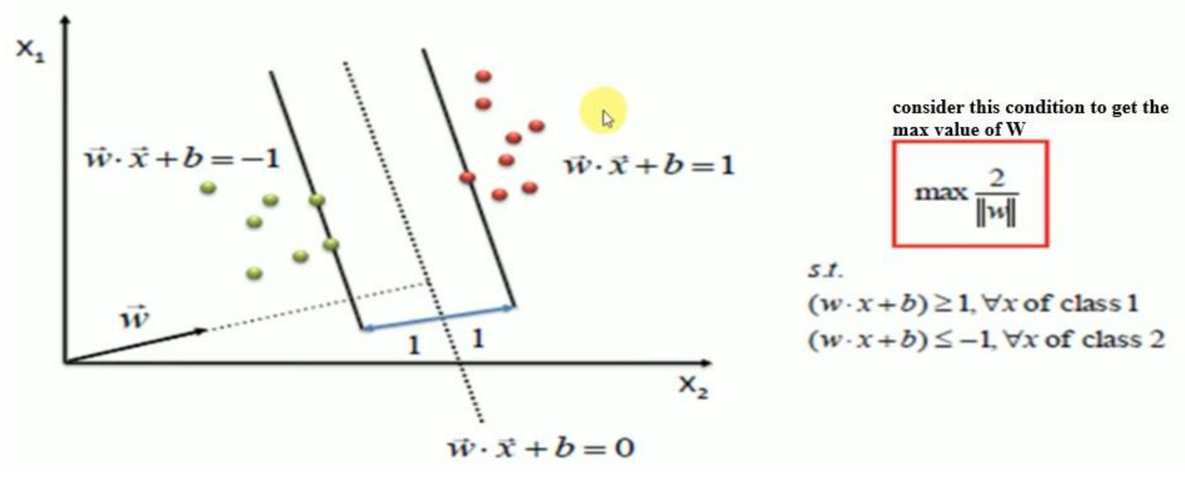
- There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points.
- This best boundary is known as the hyperplane of SVM.



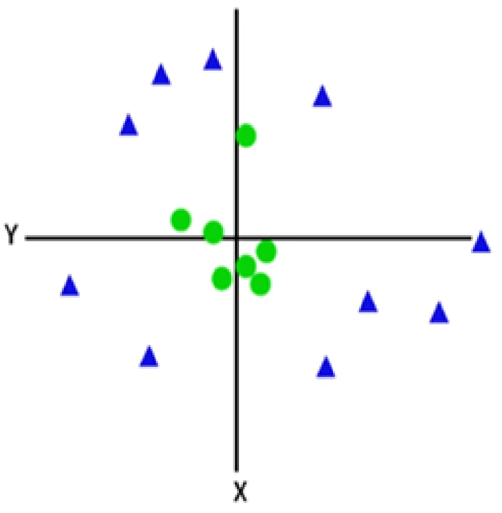
SVM can be of two types:

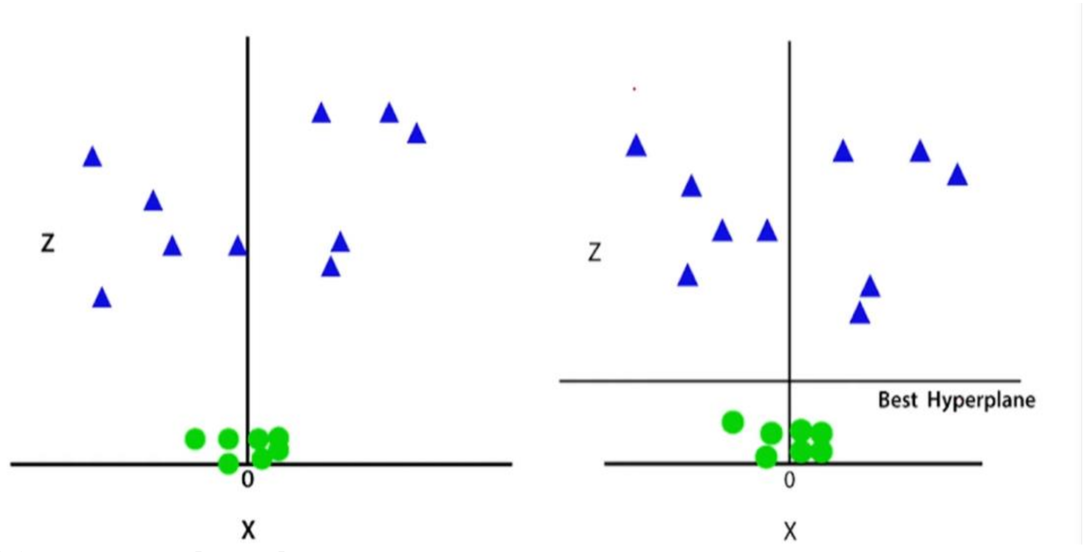
1. **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
2. **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

The equation of a hyperplane is w.x+b=0 where w is a vector normal to hyperplane and b is an offset.


x_1
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}$
1
1
x_2
$\vec{w}\cdot\vec{x}+b=0$
consider this condition to get the max value of W
$\max \frac{2}{\|w\|}$
s.t.
$(w\cdot x+b)\geq 1, \forall x$ of class 1
$(w\cdot x+b)\leq -1, \forall x$ of class 2

- **Non-Linear SVM:**
- If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line.
- So to separate these data points, we need to add one more dimension.
- For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z.
- It can be calculated as: $z=x^2+y^2$





Decision Trees and Random Forests

Decision Trees are non-linear classifiers that recursively split the feature space into regions, each associated with a class label. They are easy to understand and **visualize decision tree**, making them a popular choice for many applications.

However, decision trees can be prone to overfitting. **Random Forests** address this issue by creating an ensemble of decision trees, each trained on a random subset of the data and features.

The final prediction is obtained by combining the predictions of all trees in the ensemble, typically through majority voting.

Neural Networks

The structure and functioning of the human brain inspire **Neural Networks**. They consist of interconnected layers of nodes (or neurons) that can learn complex, non-linear relationships between input features and output classes.

Neural networks are particularly useful for high-dimensional data and have achieved state-of-the-art performance in many applications, including **image recognition**, natural language processing, and speech recognition.

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple yet effective non-linear classification algorithm. It works by finding the k closest training examples to a new observation and assigning the most common class label among these neighbours.

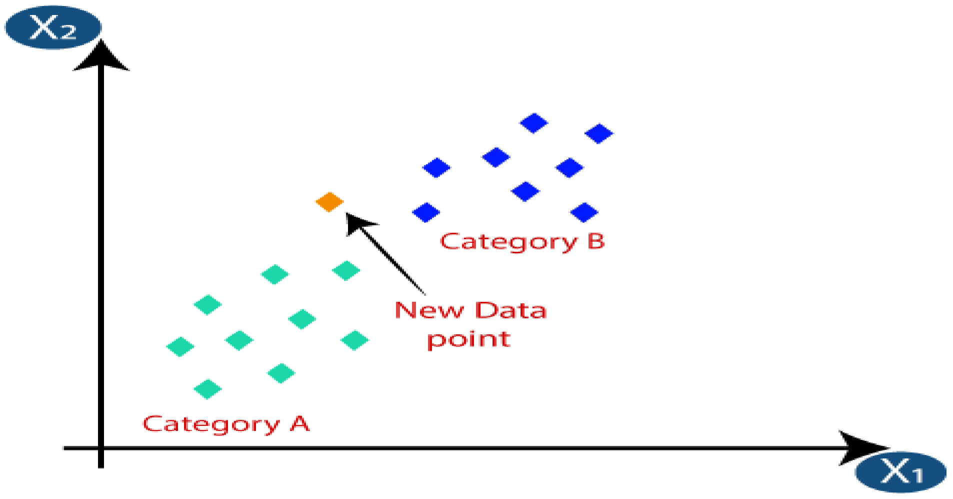
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

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. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

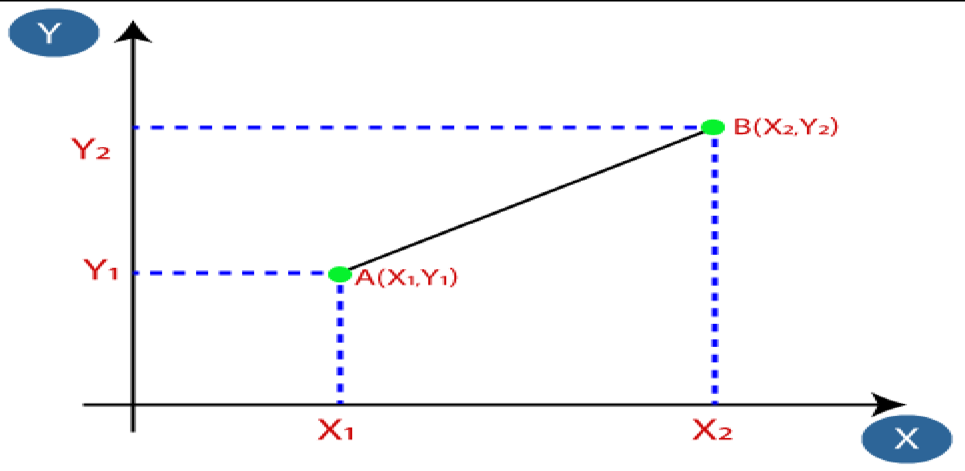
The K-NN working can be explained on the basis of the below algorithm:

- **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.

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

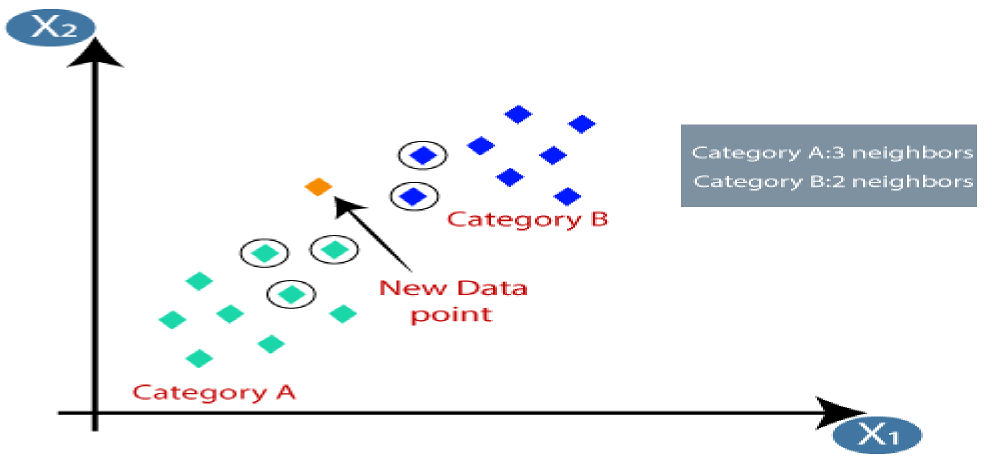


- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



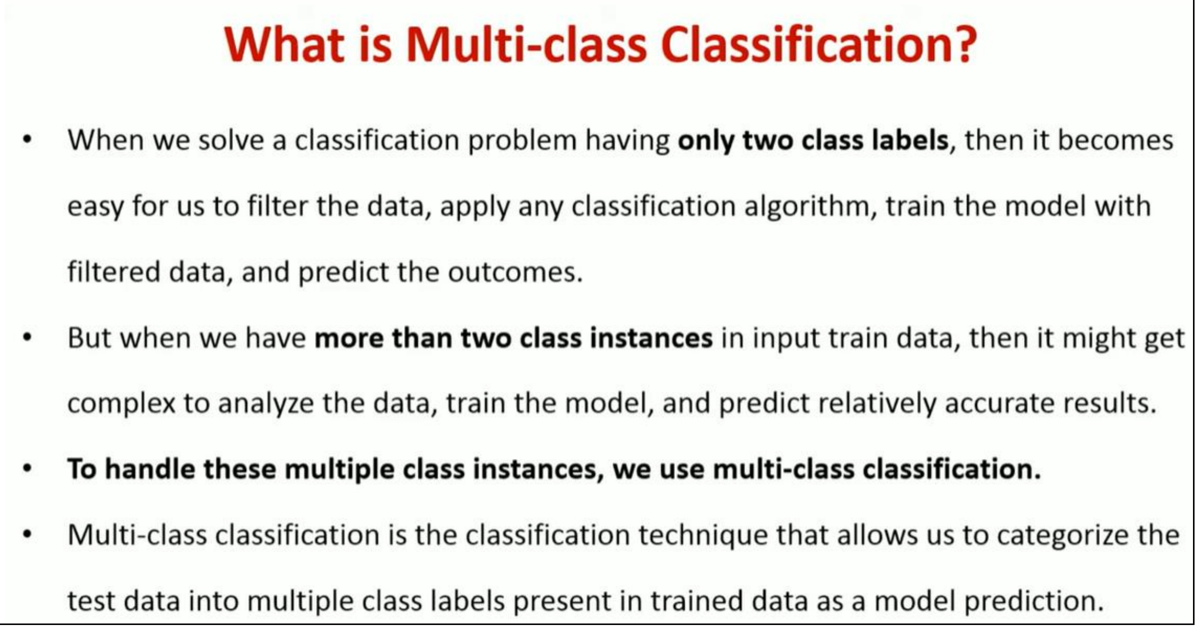
Euclidean Distance between A_1 and $B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- 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.

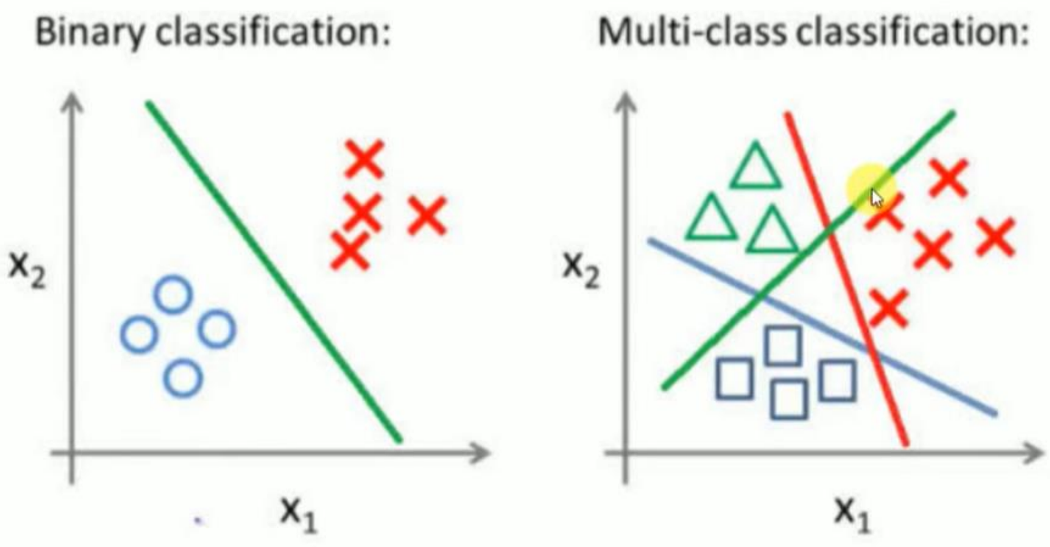
9. Define Multiclass and Multi-label Classification with a neat diagram?

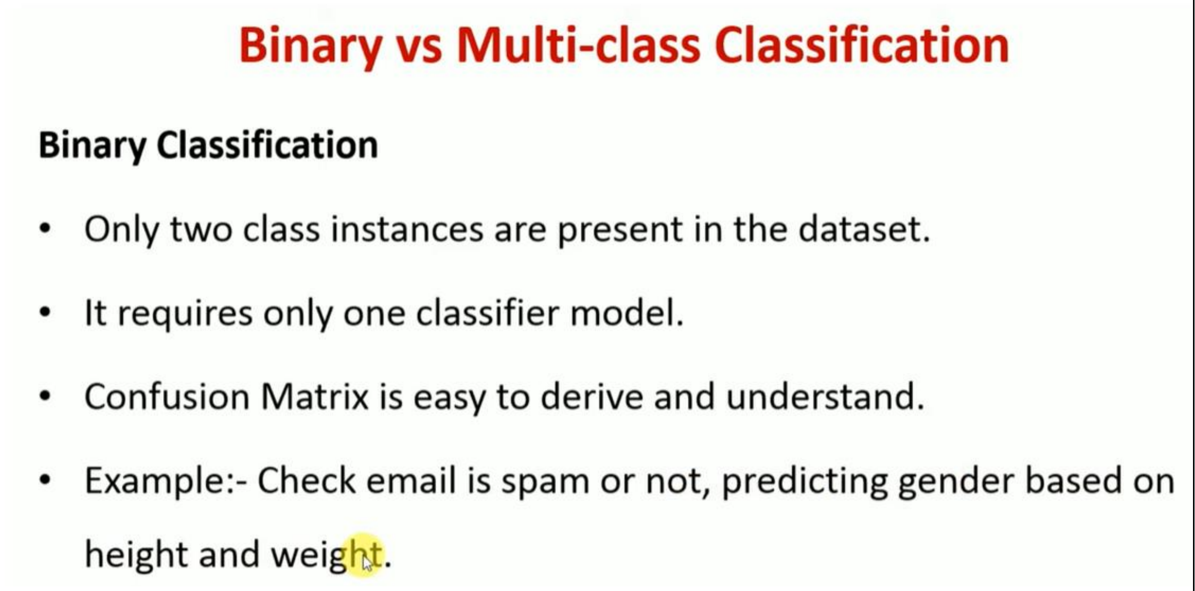


What is Multi-class Classification?

- When we solve a classification problem having **only two class labels**, then it becomes easy for us to filter the data, apply any classification algorithm, train the model with filtered data, and predict the outcomes.
- But when we have **more than two class instances** in input train data, then it might get complex to analyze the data, train the model, and predict relatively accurate results.
- **To handle these multiple class instances, we use multi-class classification.**
- Multi-class classification is the classification technique that allows us to categorize the test data into multiple class labels present in trained data as a model prediction.

Binary vs Multi-class Classification



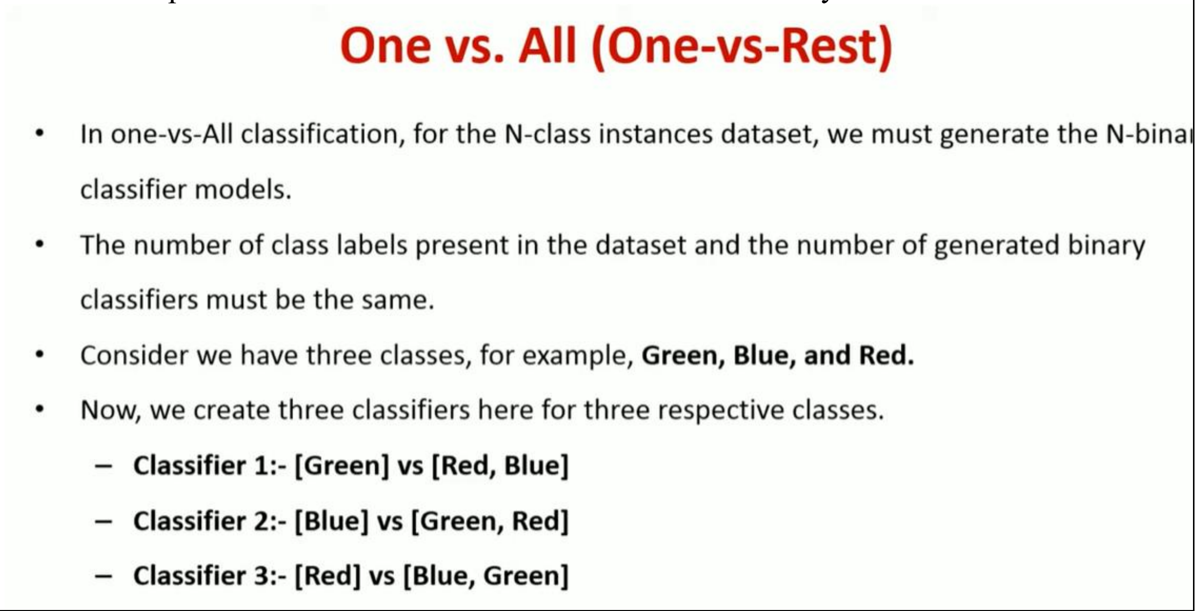


Binary vs Multi-class Classification

Binary Classification

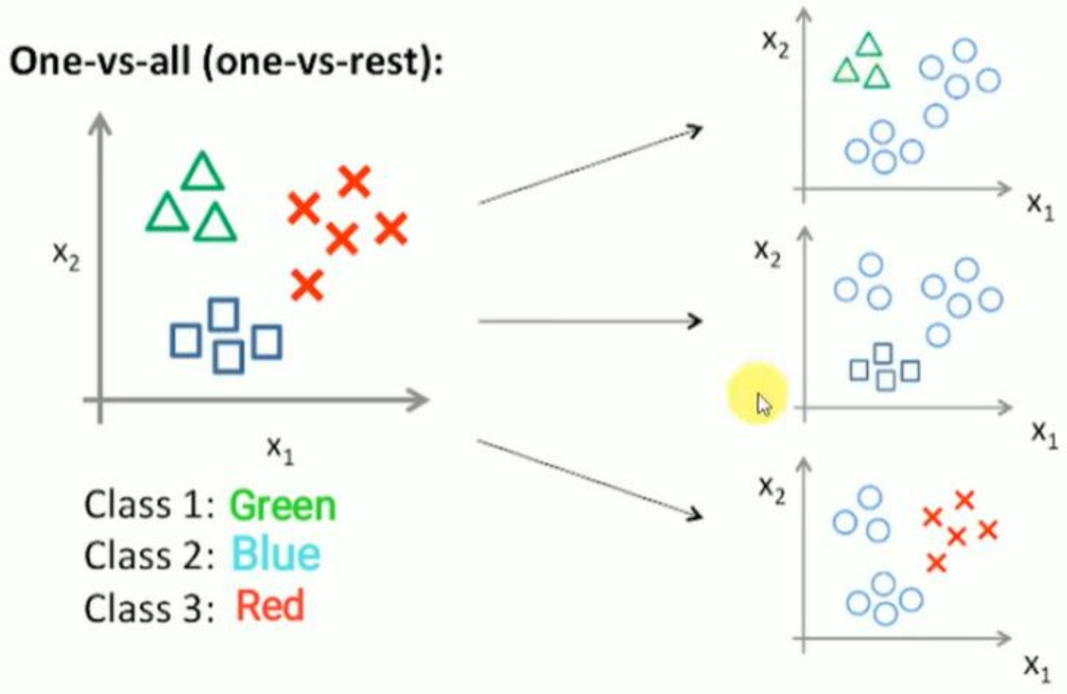
- Only two class instances are present in the dataset.
- It requires only one classifier model.
- Confusion Matrix is easy to derive and understand.
- Example:- Check email is spam or not, predicting gender based on height and weight.

NOTE:- It's possible to create multiclass classifiers out of binary classifiers.



One vs. All (One-vs-Rest)

- In one-vs-All classification, for the N-class instances dataset, we must generate the N-binar classifier models.
- The number of class labels present in the dataset and the number of generated binary classifiers must be the same.
- Consider we have three classes, for example, **Green, Blue, and Red.**
- Now, we create three classifiers here for three respective classes.
  - **Classifier 1:- [Green] vs [Red, Blue]**
  - **Classifier 2:- [Blue] vs [Green, Red]**
  - **Classifier 3:- [Red] vs [Blue, Green]**



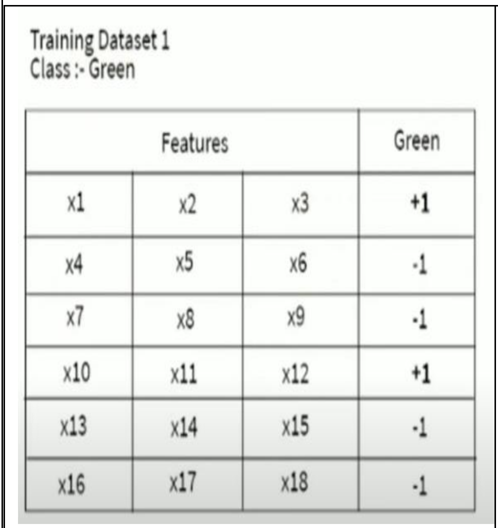
- You can see that there are three class labels **Green**, **Blue,** and **Red** present in the dataset. Now we must create a training dataset for each class.

Features			Classes
x1	x2	x3	G
x4	x5	x6	B
x7	x8	x9	R
x10	x11	x12	G
x13	x14	x15	B
x16	x17	x18	R

Class 1 :- Green

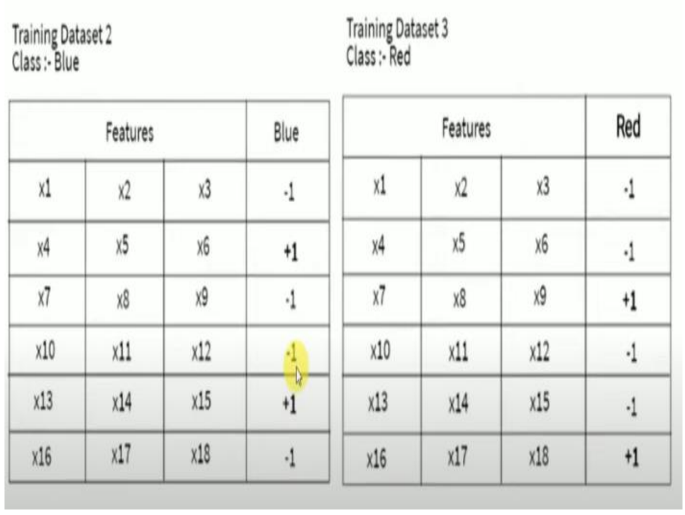
Class 2 :- Blue

Class 3 :- Red



Training Dataset 1
Class :- Green

Features			Green
x1	x2	x3	+1
x4	x5	x6	-1
x7	x8	x9	-1
x10	x11	x12	+1
x13	x14	x15	-1
x16	x17	x18	-1

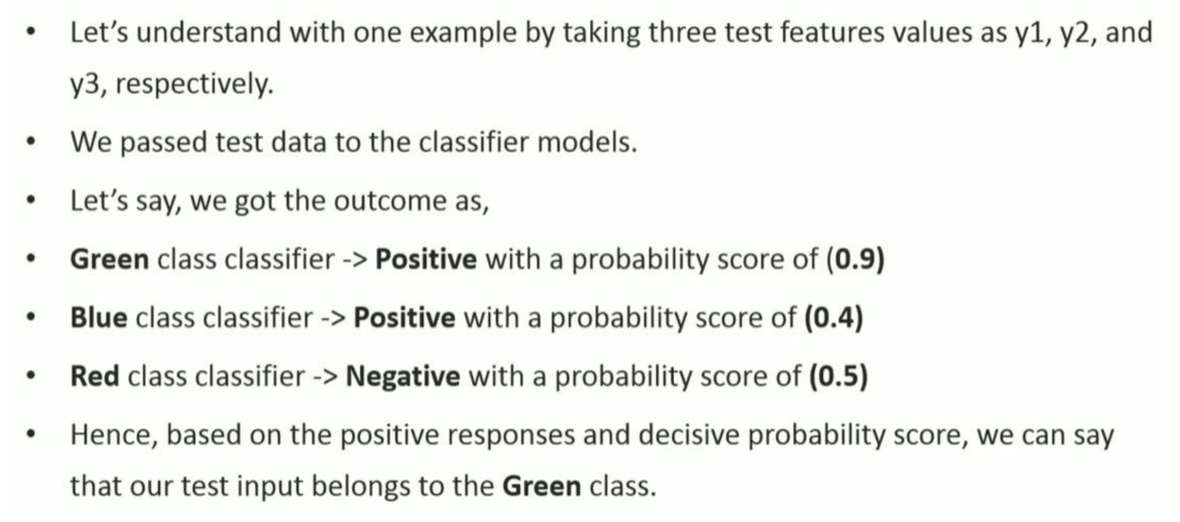


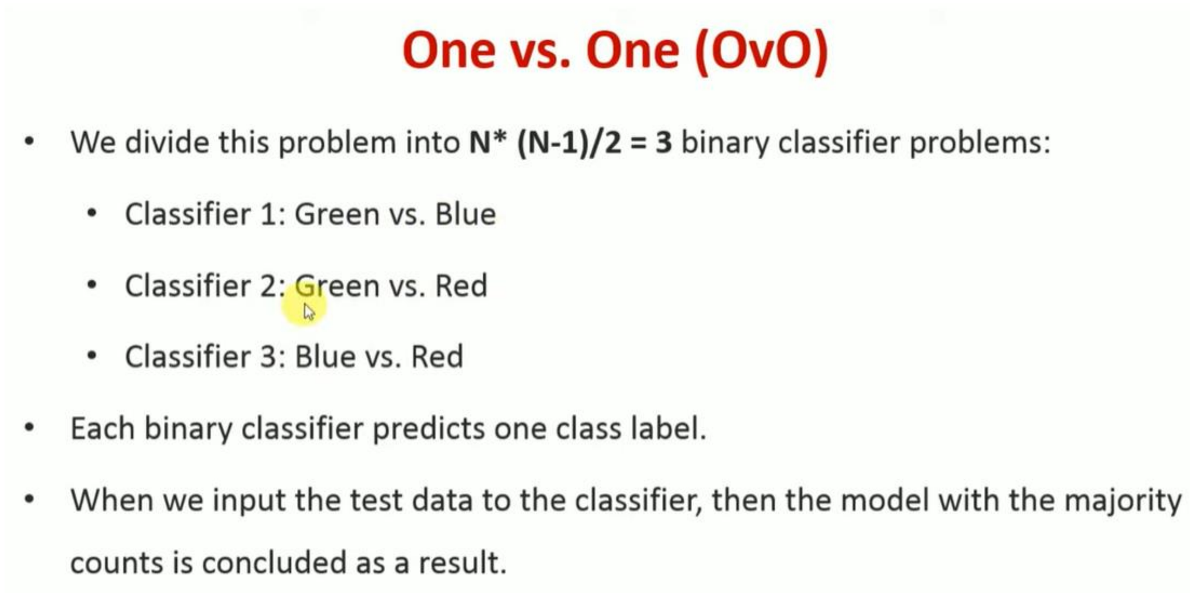
Training Dataset 2
Class :- Blue

Features			Blue
x1	x2	x3	-1
x4	x5	x6	+1
x7	x8	x9	-1
x10	x11	x12	-1
x13	x14	x15	+1
x16	x17	x18	-1

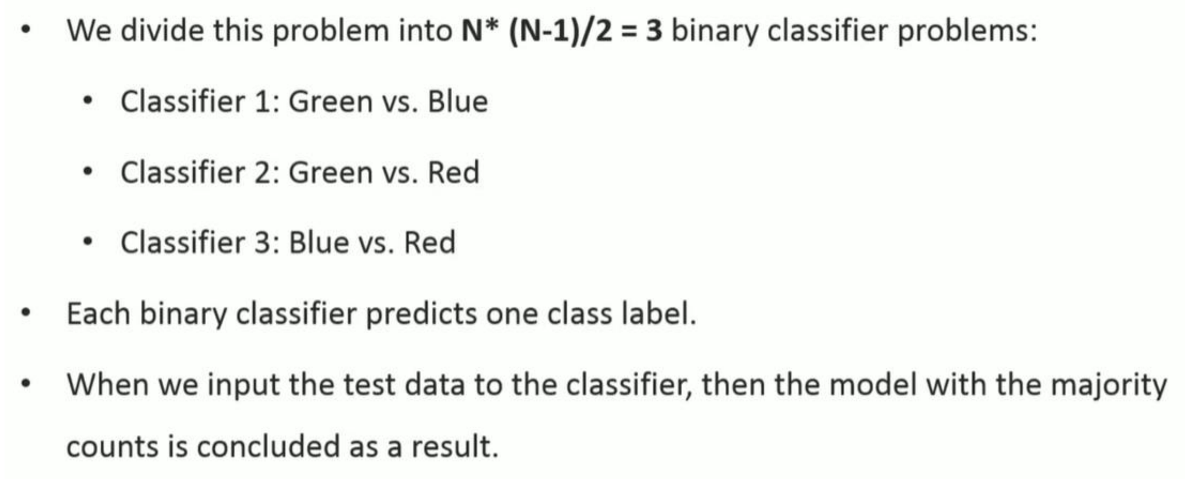
Training Dataset 3
Class :- Red

Features			Red
x1	x2	x3	-1
x4	x5	x6	-1
x7	x8	x9	+1
x10	x11	x12	-1
x13	x14	x15	-1
x16	x17	x18	+1


- Let’s understand with one example by taking three test features values as y1, y2, and y3, respectively.
- We passed test data to the classifier models.
- Let’s say, we got the outcome as,
- **Green** class classifier -> **Positive** with a probability score of (**0.9**)
- **Blue** class classifier -> **Positive** with a probability score of **(0.4)**
- **Red** class classifier -> **Negative** with a probability score of **(0.5)**
- Hence, based on the positive responses and decisive probability score, we can say that our test input belongs to the **Green** class.


One vs. One (OvO)

- We divide this problem into **N* (N-1)/2 = 3** binary classifier problems:
  - Classifier 1: Green vs. Blue
  - Classifier 2: Green vs. Red
  - Classifier 3: Blue vs. Red
- Each binary classifier predicts one class label.
- When we input the test data to the classifier, then the model with the majority counts is concluded as a result.

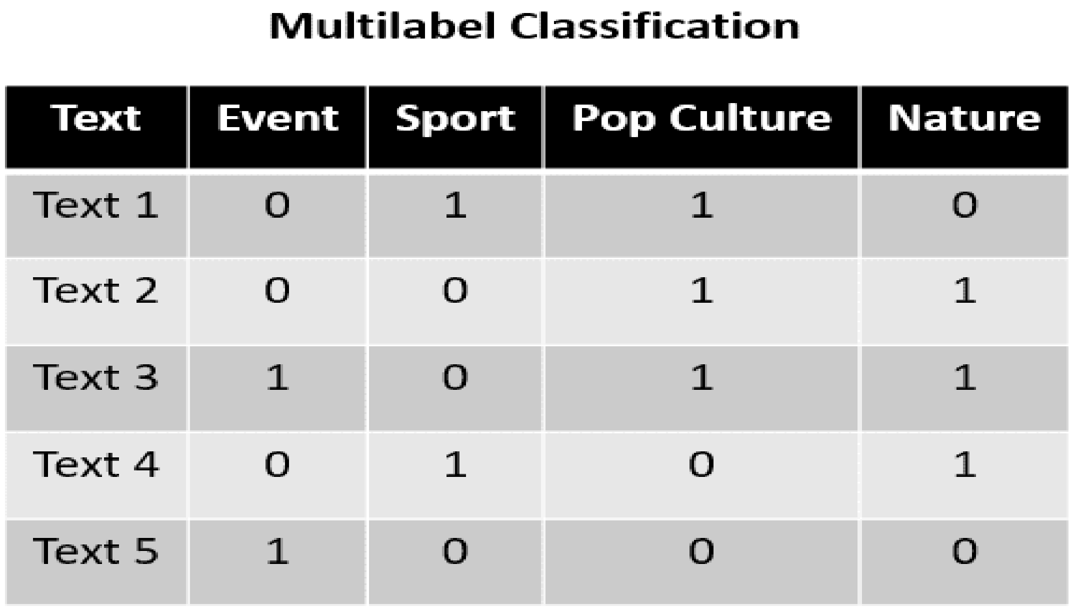


- We divide this problem into **N* (N-1)/2 = 3** binary classifier problems:
  - Classifier 1: Green vs. Blue
  - Classifier 2: Green vs. Red
  - Classifier 3: Blue vs. Red
- Each binary classifier predicts one class label.
- When we input the test data to the classifier, then the model with the majority counts is concluded as a result.

Multi-Label Classification
Multi-label classification is a classification problem where each instance can be assigned to one or more classes. For example, in text classification, an article can be about 'Technology,' 'Health,' and 'Travel' simultaneously.
Challenges in Multi-Label Classification
- **Label Correlation**: Some labels might be correlated, complicating the classification process.
- **Imbalanced Data**: Some labels might have significantly more occurrences than others.
- **Algorithm Selection**: Choosing the right algorithm depends on the nature of the data

and the problem.

Multilabel Classification is often used in the text data classification task. For example, here is an example dataset for Multilabel Classification.



Multilabel Classification

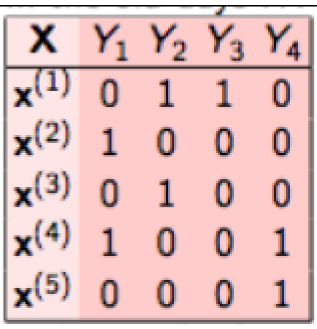
Text	Event	Sport	Pop Culture	Nature
Text 1	0	1	1	0
Text 2	0	0	1	1
Text 3	1	0	1	1
Text 4	0	1	0	1
Text 5	1	0	0	0

In the example above, imagine Text 1 to Text 5 is a sentence that can be categorized into four categories: Event, Sport, Pop Culture, and Nature. With the training data above, the Multilabel Classification task predicts which label applies to the given sentence. Each category is not against the other as they are not mutually exclusive; each label can be considered independent.
For more detail, we can see that Text 1 labels Sport and Pop Culture, while Text 2 labels Pop Culture and Nature. This shows that each label was mutually exclusive, and Multilabel Classification can have prediction output as none of the labels or all the labels simultaneously.
With that introduction, let’s try to build Multiclass Classifier with Scikit-Learn.

Techniques for Solving a Multi-Label classification problem
Basically, there are three methods to solve a multi-label classification problem, namely:

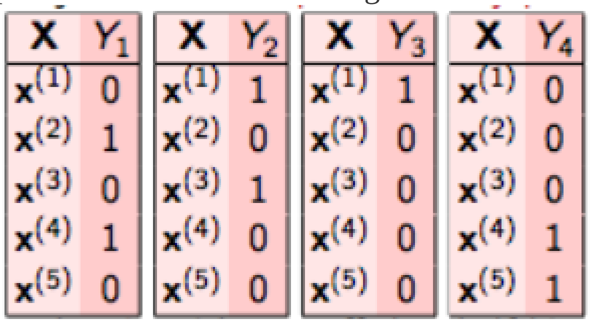
1. Problem Transformation
2. Adapted Algorithm
3. Ensemble approaches

Binary Relevance
This is the simplest technique, which basically treats each label as a separate single class classification problem.



X	Y_1	Y_2	Y_3	Y_4
$x^{(1)}$	0	1	1	0
$x^{(2)}$	1	0	0	0
$x^{(3)}$	0	1	0	0
$x^{(4)}$	1	0	0	1
$x^{(5)}$	0	0	0	1

For example, let us consider a case as shown below. We have the data set like this, where X is the independent feature and Y's are the target variable.
In binary relevance, this problem is broken into 4 different single class classification problems as shown in the figure below.



X	Y_1
$x^{(1)}$	0
$x^{(2)}$	1
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	0

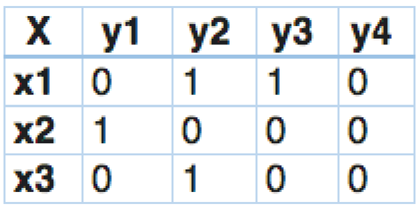
X	Y_2
$x^{(1)}$	1
$x^{(2)}$	0
$x^{(3)}$	1
$x^{(4)}$	0
$x^{(5)}$	0

X	Y_3
$x^{(1)}$	1
$x^{(2)}$	0
$x^{(3)}$	0
$x^{(4)}$	0
$x^{(5)}$	0

X	Y_4
$x^{(1)}$	0
$x^{(2)}$	0
$x^{(3)}$	0
$x^{(4)}$	1
$x^{(5)}$	1

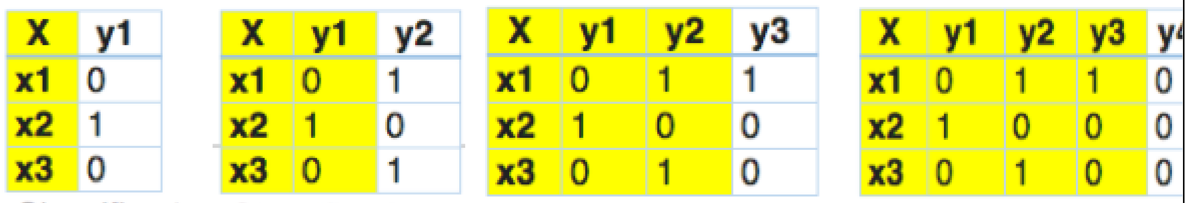
We don't have to do this manually, the multi-learn library provides its implementation in python
NOTE: Here, we have used Naive Bayes algorithm but you can use any other classification algorithm.
Now, in a multi-label classification problem, we can't simply use our normal metrics to calculate the accuracy of our predictions. For that purpose, we will use **accuracy score** metric. This function calculates subset accuracy meaning the predicted set of labels should exactly match with the true set of labels.
It is most simple and efficient method but the only drawback of this method is that it doesn't consider labels correlation because it treats every target variable independently.

Classifier Chains
In this, the first classifier is trained just on the input data and then each next classifier is trained on the input space and all the previous classifiers in the chain.
Let's try to this understand this by an example. In the dataset given below, we have X as the input space and Y's as the labels.



X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

In classifier chains, this problem would be transformed into 4 different single label problems, just like shown below. Here yellow colored is the input space and the white part represent the target variable.



X	y1
x1	0
x2	1
x3	0

Classifier 1

X	y1	y2
x1	0	1
x2	1	0
x3	0	1

Classifier 2

X	y1	y2	y3
x1	0	1	1
x2	1	0	0
x3	0	1	0

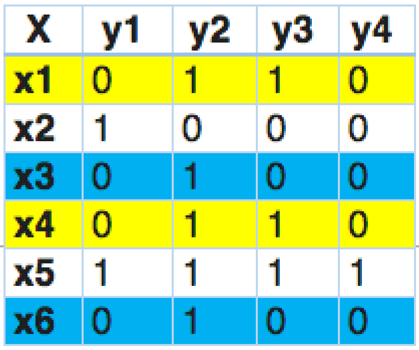
Classifier 3

X	y1	y2	y3	y
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0

Classifier 4

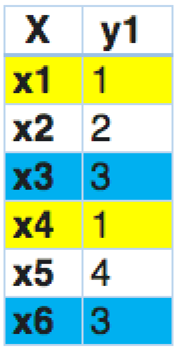
This is quite similar to binary relevance, the only difference being it forms chains in order to preserve label correlation. Gives less accuracy than binary relevance.

Label Powerset
In this, we transform the problem into a multi-class problem with one multi-class classifier is trained on all unique label combinations found in the training data.
Let’s understand it by an example.



X	y1	y2	y3	y4
x1	0	1	1	0
x2	1	0	0	0
x3	0	1	0	0
x4	0	1	1	0
x5	1	1	1	1
x6	0	1	0	0

In this, we find that x1 and x4 have the same labels, similarly, x3 and x6 have the same set of labels. So, label powerset transforms this problem into a single multi-class problem as shown below.

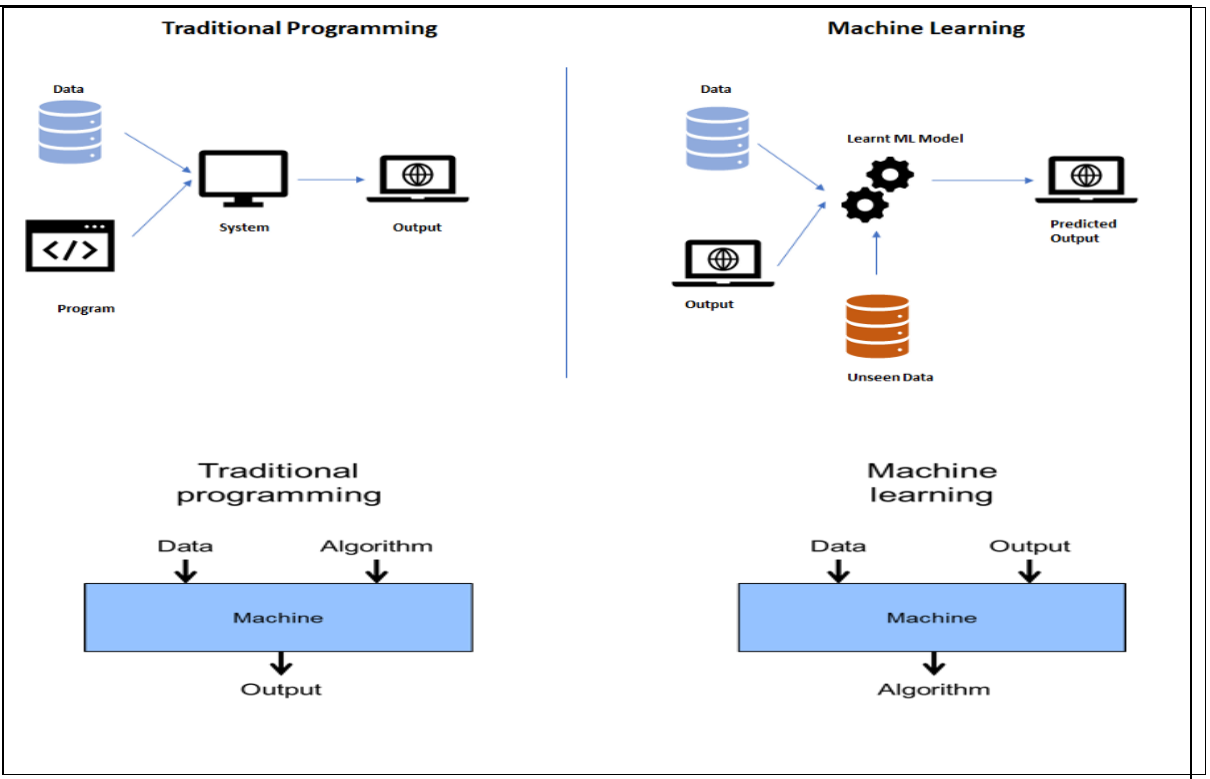


X	y1
x1	1
x2	2
x3	3
x4	1
x5	4
x6	3

So, label powerset has given a unique class to every possible label combination that is present in the training set.
This gives us the highest accuracy among all the three we have discussed till now. The only disadvantage of this is that as the training data increases, number of classes become more. Thus, increasing the model complexity, and would result in a lower accuracy.

10. Differentiate Traditional Programming approach vs Machine learning approach and discuss different perspectives of Machine Learning.

Traditional programming	**Machine Learning**
computer follows a set of predefined instructions to perform a task	The computer is given a set of examples (data) and a task to perform, but it's up to the computer to figure out how to accomplish the task based on the examples it's given the computer is given a set of examples (data) and a task to perform, but it's up to the computer to figure out how to accomplish the task based on the examples it's given



Perspectives and Issues in Machine Learning

Following are the list of issues in machine learning:

1. What algorithms exist for learning general target functions from specific training examples? In what settings will particular algorithms converge to the desired function, given sufficient training data? Which algorithms perform best for which types of problems and representations?

2. How much training data is sufficient? What general bounds can be found to relate the confidence in learned hypotheses to the amount of training experience and the character of the learner's hypothesis space?

3. When and how can prior knowledge held by the learner guide the process of generalizing from examples? Can prior knowledge be helpful even when it is only approximately correct?

4. What is the best strategy for choosing a useful next training experience, and how does the choice of this strategy alter the complexity of the learning problem?

5. What is the best way to reduce the learning task to one or more function approximation problems? Put another way, what specific functions should the system attempt to learn? Can this process itself be automated?

6. How can the learner automatically alter its representation to improve its ability to represent and learn the target function?