Regularization in Machine Learning

Regularization is one of the most important concepts of machine learning. **It is a technique to prevent the model from overfitting by adding extra information to it.**

Sometimes the machine learning model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called **overfitted. This problem can be deal with the help of a regularization technique.**

This technique can be used in such a way that it will allow to maintain all variables or features in the model by reducing the magnitude of the variables. Hence, it maintains accuracy as well as a generalization of the model.

It mainly regularizes or reduces the coefficient of features toward zero.

Explanation for types of REGULARIZATION

Ridge Regression

Consider the equation: Y = 0.9 + 1.2X1 + 20X2 + 39X3 - This is imp. To determine Y – since Regression Co-efficient with high value.

Scaled down to: Y=0.9 + 0.7X1 + 2X2 + 5X3

Where Y=Dependent and X=Independent Variables

Ridge R = Loss + $\alpha \, \| W \|^2$ (Penalty)

$\| W \|^2 = W_1^2 + W_2^2 \ \ldots\ldots\ldots + W_n^2$

α – Constant and $\|W\|$ - Vector of co-efficient

- Loss (diff. between the actual and predicted) –after training phase.
- Penalty – will bring down the loss
- Need of Regularization – Down the computational expenses- down the complexity of the model – to shrink the magnitude of the regression coefficient (α)
- When α – increases (0.01, 1,..) , magnitude of the regression coefficient will decrease – (will be almost equal to 0)

Lasso Regression

Consider the equation: Y= 0.9 + 1.2X1 + 20X2 + 39X3 - This is imp. To determine Y – since Regression Co-efficient with high value.

Scaled down to: Y=0.9 + 0X1 + 0X2 + 5X3

Where Y=Dependent and X=Independent Variables

Lasso R = Loss + α || W|| (Penalty)

$\| W \| = W_1 + W_2 \ldots\ldots\ + W_n$

α – Constant and ||W|| - Vector of co-efficient

- Loss (diff. between the actual and predicted) –after training phase.
- Penalty – will bring down the loss
- Need of Regularization – Down the computational expenses- down the complexity of the model – to shrink the magnitude of the regression coefficient (α)
- When α – increases (0.01, 1,..) , magnitude of the regression coefficient will decrease – (will be equal to 0)- eliminate those variables - Y =0.9+5X3

-Feature Selection too

How does Regularization Work

Regularization works by adding a penalty or complexity term to the complex model. Let's consider the simple linear regression equation:

$$y= \beta 0+\beta 1x1+\beta 2x2+\beta 3x3+\cdots+\beta nxn +b$$

- In the above equation, Y represents the value to be predicted
- **X1, X2, …Xn are the features for Y.**
- **β0,β1,…..βn are the weights or magnitude attached to the features**, respectively, **b represents the intercept**.
- Linear regression models try to **optimize the β0 and b to minimize the cost function.**

One would also add a loss function and optimize parameter to make the model that can predict the accurate value of Y. The loss function for the linear regression is called as RSS or Residual sum of squares.

Techniques of Regularization

Ridge Regression

- Ridge regression is one of the types of linear regression in which **a small amount of bias is introduced so that we can get better long-term predictions.**
- Ridge regression is a regularization technique, which is used to reduce the complexity of the model. It is also called as **L2 regularization**.
- In this technique, **the cost function is altered by adding the penalty term to it. The amount of bias added to the model is called Ridge Regression penalty**. We can calculate it by multiplying with the **lambda to the squared weight of each individual feature.**
- The equation for the cost function in ridge regression will be:

$$\sum_{i=1}^{M}(y_i - y'_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{n}\beta_j * x_{ij}\right)^2 + \lambda\sum_{j=0}^{n}\beta_j^2$$

- In the above equation, **the penalty term regularizes the coefficients of the model, and hence ridge regression reduces the amplitudes of the coefficients that decreases the complexity of the model and less computational expenses.**

- As we can see from the above equation, if the values of **λ tend to zero, the equation becomes the cost function of the linear regression model.** Hence, for **the minimum value of λ, the model will resemble the linear regression model.**
- **A general linear or polynomial regression will fail, if there is high collinearity between the independent variables, so to solve such problems, Ridge regression can be used.**

[Collinearity means a set of points or objects are aligned **on a single line**. In statistics, collinearity can also refer to a **correlation between independent variables** in a regression analysis. **]**

- It helps to solve the problems if we have more parameters than samples.

Lasso Regression:

- Lasso regression is another regularization technique to reduce the complexity of the model. It stands for **Least Absolute and Selection Operator.**
- **It is similar to the Ridge Regression except that the penalty term contains only the absolute weights instead of a square of weights.**
- **Since it takes absolute values, hence, it can shrink the slope to 0, whereas Ridge Regression can only shrink it near to 0.**
- It is also called as **L1 regularization.** The equation for the cost function of Lasso regression will be:

$$\sum_{i=1}^{M}\left(y_i - y'_i\right)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{n}\beta_j * x_{ij}\right)^2 + \lambda\sum_{j=0}^{n}|\beta_j|$$

- **Some of the features in this technique are completely neglected for model evaluation.**

- Hence, **the Lasso regression can help us to reduce the overfitting in the model as well as the feature selection.**

Key Difference between Ridge Regression and Lasso Regression

- **Ridge regression** is mostly used to reduce the overfitting in the model, and it includes all the features present in the model.
- **Lasso regression** helps to reduce the overfitting in the model as well as feature selection.

Elastic Net Regression

Elastic Net Regression is a combination of both **L1 as well as L2 regularization.** That implies that we **add the absolute norm of the weights as well as the squared measure of the weights**. With the help of an extra hyperparameter that controls the ratio of the L1 and L2 regularization.

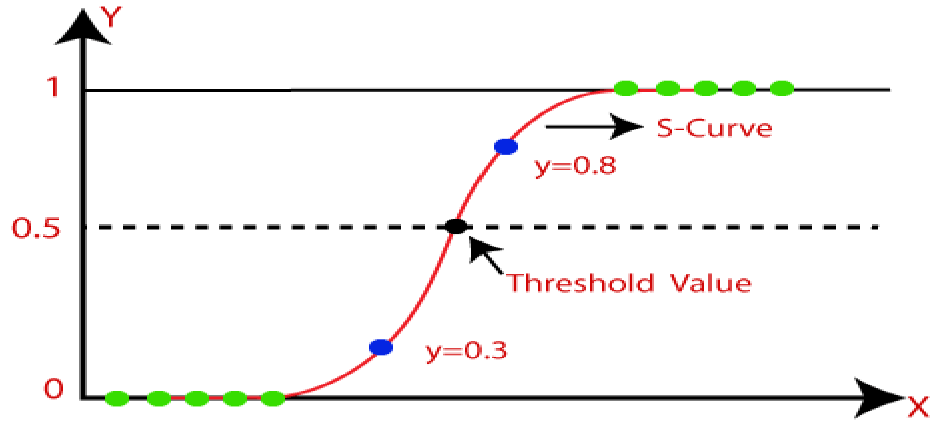
Benefits of Regularization

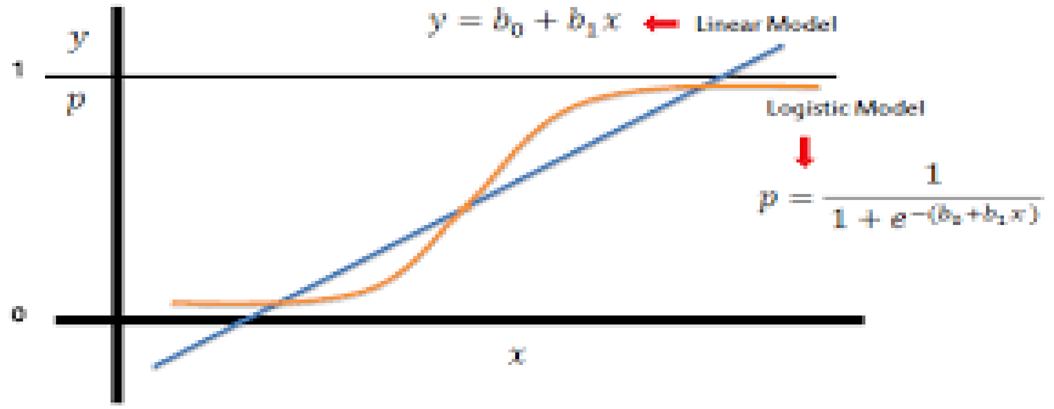
- **Prevents Overfitting:** Regularization helps models focus on underlying patterns instead of memorizing noise in the training data.
- **Improves Interpretability:** L1 (Lasso) regularization simplifies models by **reducing less important feature coefficients to zero.**
- **Enhances Performance:** Prevents **excessive weighting of outliers or irrelevant features**, improving overall model accuracy.
- **Stabilizes Models: Reduces sensitivity to minor data changes**, ensuring consistency across different data subsets.
- **Prevents Complexity:** Keeps models from becoming too complex, which is **crucial for limited or noisy data**.
- **Handles Multicollinearity: Reduces the magnitudes of correlated coefficients**, improving model stability.
- **Allows Fine-Tuning: Hyperparameters like alpha and lambda control regularization strength, balancing bias and variance.**
- **Promotes Consistency:** Ensures **reliable performance across different datasets**, reducing the risk of large performance shifts.

Logistic Regression

- Logistic regression is one of the most popular Machine learning algorithm that comes under **Supervised Learning techniques.**
- It can be used for **Classification as well as for Regression** problems, but mainly used for Classification problems.
- Logistic regression is used to **predict the categorical dependent variable with the help of independent variables**.
- The output of Logistic Regression problem can be only between the **0 and 1.**
- Logistic regression **can be used where the probabilities between two classes is required. Such as whether it will rain today or not, either 0 or 1, true or false etc.**

- In logistic regression**, we pass the weighted sum of inputs through an activation function that can map values in between 0 and 1. Such activation function is known as sigmoid function** and the curve obtained is called as **sigmoid curve or S-curve**. Consider the below image:





Sigmoid function: It is simply - trying to **convert the independent variable into an expression of probability (range between 0 to 1) with perspective to the dependent variable.**

X- Independent variable, e – Euler's constant (2.72) - which is the **base of the natural logarithm and is used within the logistic function that calculates the probability of an event occurring based on the model's predictions; essentially, it's the number that the regression equation is exponentiated by - to produce the final probability value.**

Datapoints between,

0 – No probability of certain occurrence

1 – Probability of certain occurrence is there

0.5 – Unclassifiable (Rare, because of the sigmoid function)

Types of Logistic Regression

On the basis of the categories, Logistic Regression can be classified into three types:

1. **Binomial:** In binomial Logistic regression, there can be only **two possible types of the dependent variables,** such as 0 or 1, Pass or Fail, etc.

2. **Multinomial:** In multinomial Logistic regression, **there can be 3 or more possible unordered types of the dependent variable**, such as “cat”, “dogs”, or “sheep”

3. **Ordinal:** In ordinal Logistic regression, there can be **3 or more possible ordered types of dependent variables**, such as “low”, “Medium”, or “High”.

Assumptions of Logistic Regression

The assumptions of logistic regression - as understanding these assumptions is important to ensure that one can use the appropriate application - of the model. The assumption include:

1. **Independent observations:** Each observation is independent of the other. meaning there is **no correlation** between any input variables.
2. **Binary dependent variables:** It takes the assumption that the dependent variable must be binary or **dichotomous,** meaning it can take **only two values**. **For more than two categories SoftMax functions are used**. *(SoftMax Fn.- is a mathematical function that **converts a vector of raw prediction scores (often called logits) from the neural network into probabilities**.)*
3. Linearity relationship between independent variables and log odds: **The relationship between the independent variables and the log odds of the dependent variable should be linear.**
4. **No outliers:** There should be no outliers in the dataset.
5. **Large sample size:** The sample size is sufficiently large

How does Logistic Regression work?

- The logistic regression model **transforms the linear regression function continuous value output into categorical value output using a sigmoid function,** which maps any real-valued set of independent variables input into a value between 0 and 1. This function is known as the **logistic function.**

- Let the independent input features be:

$$X = \begin{bmatrix} x_{11} & \dots & x_{1m} \\ x_{21} & \dots & x_{2m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nm} \end{bmatrix}$$

and the dependent variable is Y having only binary value i.e. 0 or 1.

$$Y = \begin{cases} 0 & \text{if } Class\ 1 \\ 1 & \text{if } Class\ 2 \end{cases}$$

then, apply the multi-linear function to the input variables X.

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

Here x_i is the ith observation of X, $w_i = [w_1, w_2, w_3, \cdots, w_m]$ is the weights or Coefficient, and b is the bias term also known as intercept. simply this can be represented as the dot product of weight and bias.

$$z = w \cdot X + b$$

Linear Regression	Logistic Regression
Linear regression is used to predict the continuous dependent variable using a given set of independent variables.	Logistic Regression is used to predict the categorical dependent variable using a given set of independent variables.
Linear Regression is used for solving Regression problem.	Logistic regression is used for solving Classification problems.
In Linear regression, we predict the value of continuous variables.	In logistic Regression, we predict the values of categorical variables.
In linear regression, we find the best fit line, by which we can easily predict the output.	In Logistic Regression, we find the S-curve by which we can classify the samples.
Least square estimation method is used for estimation of accuracy.	Maximum likelihood estimation method is used for estimation of accuracy.
The output for Linear Regression must be a continuous value, such as price, age, etc.	The output of Logistic Regression must be a Categorical value such as 0 or 1, Yes or No, etc.
In Linear regression, it is required that relationship between dependent variable and independent variable must be linear.	In Logistic regression, it is not required to have the linear relationship between the dependent and independent variable.
In linear regression, there may be collinearity between the independent variables.	In logistic regression, there should not be collinearity between the independent variable.

Hypothesis representation

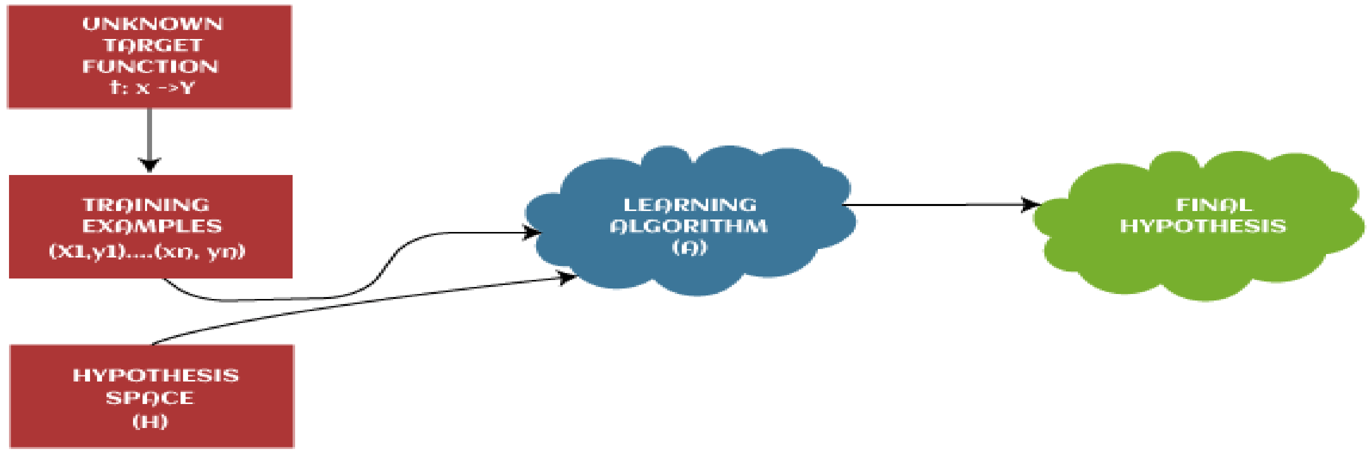
The hypothesis is defined as the supposition or proposed explanation based on insufficient evidence or assumptions. It is **just a guess based on some known facts but has not yet been proven**. A good hypothesis is testable, which results in either true or false.

Example: Let's understand the hypothesis with a common example. Some scientist claims that ultraviolet (UV) light can damage the eyes then it may also cause blindness.

In this example, a scientist just claims that UV rays are harmful to the eyes, but we assume they may cause blindness. However, it may or may not be possible. Hence, these types of assumptions are called a hypothesis.

Hypothesis in Machine Learning (ML)

The hypothesis is one of the commonly used concepts of statistics in Machine Learning. It is specifically used in Supervised Machine learning, where an ML model learns a function that best maps the input to corresponding outputs with the help of an available dataset.


UNKNOWN TARGET FUNCTION f: x ->Y
TRAINING EXAMPLES (X1,y1)....(xn, yn)
HYPOTHESIS SPACE (H)
LEARNING ALGORITHM (A)
FINAL HYPOTHESIS

In supervised learning techniques, **the main aim is to determine the possible hypothesis out of hypothesis space that best maps input to the corresponding or correct outputs.**

- There are some common methods given to find out the possible hypothesis from the Hypothesis space, where **hypothesis space is represented by uppercase-h (H) and hypothesis by lowercase-h (h).** Th ese are defined as follows:

Hypothesis space (H):

- ***Hypothesis space is defined as a set of all possible legal hypotheses; hence it is also known as a hypothesis set.*** It is used by supervised machine learning algorithms to determine the best possible hypothesis to describe the target function or best maps input to output.

- It is **often constrained** by choice of the framing of the problem, the choice of model, and the choice of model configuration.

Hypothesis (h):

It is defined as the approximate function that best describes the target in supervised machine learning algorithms. It is primarily based on data as well as bias and restrictions applied to data.

Hence hypothesis (h) can be concluded as a single hypothesis that maps input to proper output and can be evaluated as well as used to make predictions.

The hypothesis (h) can be formulated in machine learning as follows:

$y= mx + b$

Where,

Y: Range

m: Slope of the line which divided test data or changes in y divided by change in x.

x: domain

c: intercept (constant)

Defining Hypothesis in Machine Learning

In machine learning, a hypothesis is a mathematical function or model that converts input data into output predictions. The model's first belief or explanation is based on the facts supplied. The hypothesis is typically expressed as a collection of parameters characterizing the behavior of the model.

If we're building a model to predict the price of a property based on its size and location. The hypothesis function may look something like this −

$$\mathrm{h(x)} = \theta 0 + \theta 1 * \mathrm{x1} + \theta 2 * \mathrm{x2}$$

The hypothesis function is h(x), its input data is x, the model's parameters are 0, 1, and 2, and the features are x1 and x2.

The machine learning model's purpose is to discover the optimal values for parameters 0 through 2 that minimize the difference between projected and actual output labels.

To put it another way, we're looking for the hypothesis function that best represents the underlying link between the input and output data.

Types of Hypotheses in Machine Learning

- **A hypothesis is an explanation or solution to a problem based on insufficient data.** It acts as a springboard for further investigation and experimentation. **A hypothesis is a machine learning function that converts inputs to outputs based on some assumptions. A good hypothesis contributes to the creation of an accurate and efficient machine-learning model.** Several machine learning theories are as follows −

1. Null Hypothesis

- **A null hypothesis is a basic hypothesis that states that no link exists between the independent and dependent variables**. In other words, it assumes the independent variable has no influence on the dependent variable. **It is symbolized by the symbol H0**. **If the p-value falls outside the significance level, the null hypothesis is typically rejected ()**. If the null hypothesis is correct, the coefficient of determination is the probability of rejecting it. A null hypothesis is involved in test findings such as **t-tests and ANOVA** (**(Analysis of Variance)**

2. Alternative Hypothesis (research hypothesis)

- An alternative hypothesis is a hypothesis that **contradicts the null hypothesis. It assumes that there is a relationship between the independent and dependent variables**. In other words, it assumes that there is an effect of the independent variable on the dependent variable. **It is denoted by Ha**. **An alternative hypothesis is generally accepted if the p-value is less than the significance level (α).**

Note: The level of statistical significance is usually represented as a P-value between 0 and 1

3. One-tailed Hypothesis

- **A one-tailed test is a type of significance test in which the region of rejection is located at one end of the sample distribution. It denotes that the estimated test parameter is more or less than the crucial value, implying that the alternative hypothesis rather than the null hypothesis should be accepted**. It is most commonly used in the chi-square distribution, where all of the crucial areas, related to, are put in either of the two tails. Left-tailed or right-tailed one-tailed tests are both possible.

4. Two-tailed Hypothesis

- **The two-tailed test is a hypothesis test in which the region of rejection or critical area is on both ends of the normal distribution.** It determines whether the sample tested falls within or outside a certain range of values, and **an alternative hypothesis is accepted if the calculated value falls in either of the two tails of the probability distribution**. α is bifurcated into two equal parts, and the **estimated parameter is either above or below the assumed parameter,** so extreme values work as evidence against the null hypothesis.
- Overall, **the hypothesis plays a critical role in the machine learning model. It provides a starting point for the model to make predictions and helps to guide the learning process**. The accuracy of the hypothesis is **evaluated using various metrics like mean squared error or accuracy**.

- The hypothesis is a mathematical function or model that converts input data into output predictions, typically expressed as a collection of parameters characterizing the behavior of the model. It is an explanation or solution to a problem based on insufficient data. **A good hypothesis contributes to the creation of an accurate and efficient machine-learning model. A two-tailed hypothesis is used when there is no prior knowledge or theoretical basis to infer a certain direction of the link.**

Cost Function

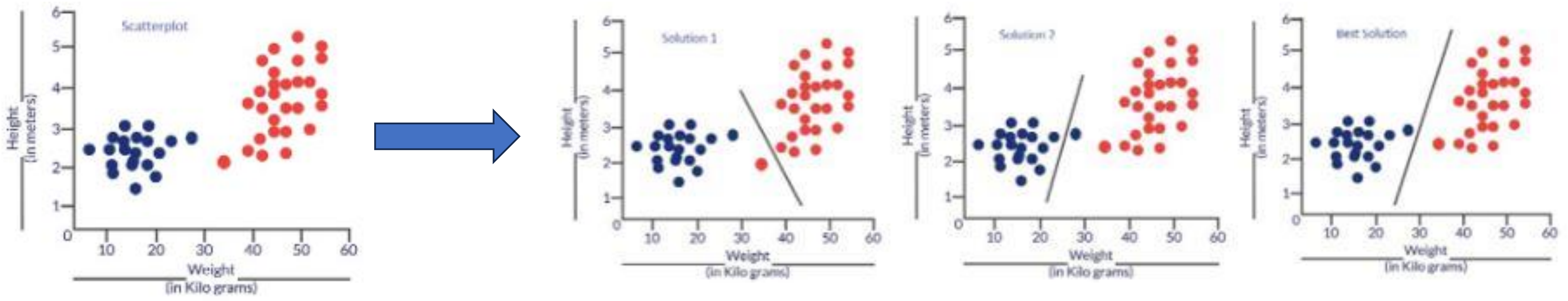
- **A cost function, also referred to as a loss function or objective function**, is a key concept in machine learning.
- **It quantifies the difference between predicted and actual values, serving as a metric to evaluate the performance of a model.**
- The primary objective is to **minimize the cost function, indicating better alignment between predicted and observed outcomes.**
- In essence, the cost function in machine learning guides the model toward optimal predictions by measuring its accuracy against the training data.

Loss function: Used when we refer to the **error for a single training example.**
Cost function: Used to refer to an **average of the loss functions over an entire training dataset.**

Why to use a Cost function

Consider a scenario where we wish to classify data. Suppose we have the **height & weight details of some cats & dogs. Let us use these 2 features to classify them correctly. If we plot these records, we get the following scatterplot:**

Blue dots are cats & red dots are dogs. Following are some solutions to the classification problem.



- Essentially all three classifiers have very high accuracy **but the third solution is the best because it does not misclassify any point.** The reason why it classifies all the points perfectly is that the line is almost exactly in between the two groups, and not closer to any one of the groups. This is where the concept of cost function comes in. **Cost function helps us reach the optimal solution. The cost function is the technique of evaluating "the performance of our algorithm/model".**

- It takes both predicted outputs by the model and actual outputs and calculates **how much wrong the model was in its prediction**. **It outputs a higher number if our predictions differ a lot from the actual values**. As we **tune our model to improve the predictions**, the cost function acts as an indicator of how the model has improved. **This is essentially an optimization problem. The optimization strategies always aim at "minimizing the cost function".**

Cost Function in Machine Learning

1. Regression cost Function:

- Regression models deal with **predicting a continuous value for example salary of an employee, price of a car, loan prediction, etc**. A cost function used in the regression problem is called "**Regression Cost Function**". They are calculated on the distance-based error as follows:

  Error = y-y'

- Where, Y – Actual Input and Y' – Predicted output

The most used Regression cost functions are below,

1.1 Mean Error (ME)

- In this cost function, **the error for each training data is calculated and then the mean value of all these errors is derived.**
- Calculating the mean of the errors is the **simplest and most intuitive** way possible.
- **The errors can be both negative and positive. So they can cancel each other out during summation giving zero mean error for the model. - Thus this is not a recommended cost function** but it does **lay the foundation for other cost functions of regression models**.

1.2 **Mean Squared Error (MSE)**

- This improves the drawback of the Mean Error **- Here a square of the difference between the actual and predicted value is calculated to avoid any possibility of negative error.**

- It is measured as the **average of the sum of squared differences between predictions and actual observations.**

$$MSE = \frac{\sum_{i=0}^{n}(y-y')^2}{n}$$

MSE = (sum of squared errors)/n

•It is also known as **L2 loss.**

•In **MSE, since each error is squared, it helps to penalize even small deviations in prediction when compared to MAE.**

•**But if our dataset has outliers that contribute to larger prediction errors, then squaring this error further will magnify the error many times more and also lead to higher MSE error.**

•Hence one can say that it is **less robust to outliers**

1.3 Mean Absolute Error (MAE)

- This cost **function also addresses the shortcoming of mean error differently. Here an absolute difference between the actual and predicted value is calculated to avoid any possibility of negative error.**

- So in this cost function, MAE is measured as the average of the sum of absolute differences between predictions and actual observations.

$$\text{MAE} = \frac{\sum_{i=0}^{n} |y - y\prime|}{n}$$

MAE = (sum of absolute errors)/n

•It is also known **as L1 Loss.**

•It is **robust to outliers** thus it will give better results even when our dataset has noise or outliers.

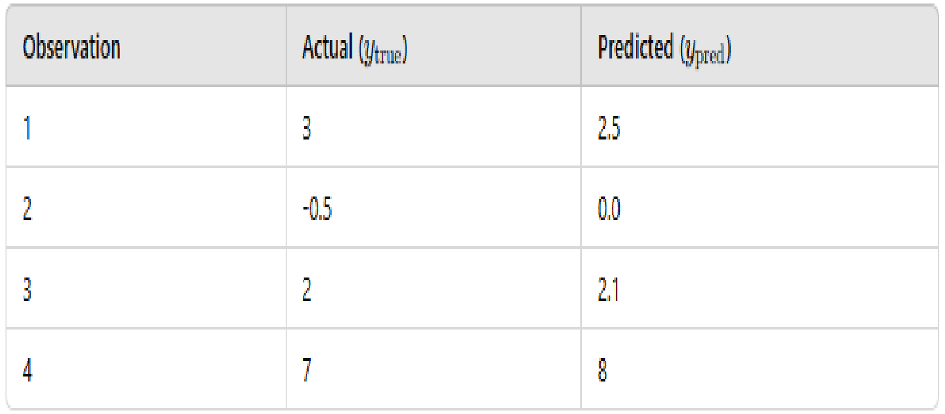
2. Cost functions for Classification problems:

Cost functions used in classification problems are different than what we use in the regression problem**. A commonly used loss function for classification is the cross-entropy loss.**
Cross-entropy – example - Consider that we have a classification problem of 3 classes as follows.

Class(Orange,Apple,Tomato)

Question 1: Calculating Mean Error, MAE and MSE

- **Given:**



Observation	Actual (y_{true})	Predicted (y_{pred})
1	3	2.5
2	-0.5	0.0
3	2	2.1
4	7	8

- **Mean Error (ME) = -0.275**

Since the ME is negative, it suggests that the model tends to overpredict on average.

- **MAE – 0.525**

- **MSE - 0.377**

- The machine learning model will give a probability distribution of these 3 classes as output for a given input data. **The class with the highest probability is considered as a winner class for prediction.**

Output = [P(Orange),P(Apple),P(Tomato)]

The actual probability distribution for each class is shown below.

- ***Orange = [1,0,0]***
- ***Apple = [0,1,0]***
- **Tomato = [0,0,1]**

- If during the training phase, the input class is Tomato, **the predicted probability distribution should tend towards the actual probability distribution of Tomato. If the predicted probability distribution is not closer to the actual one, the model has to adjust its weight**. This is where cross-entropy becomes a tool to calculate how much far the predicted probability distribution from the actual one is.
- In other words, **Cross-entropy can be considered as a way to measure the distance between two probability distributions. The following image illustrates the intuition behind cross-entropy:**

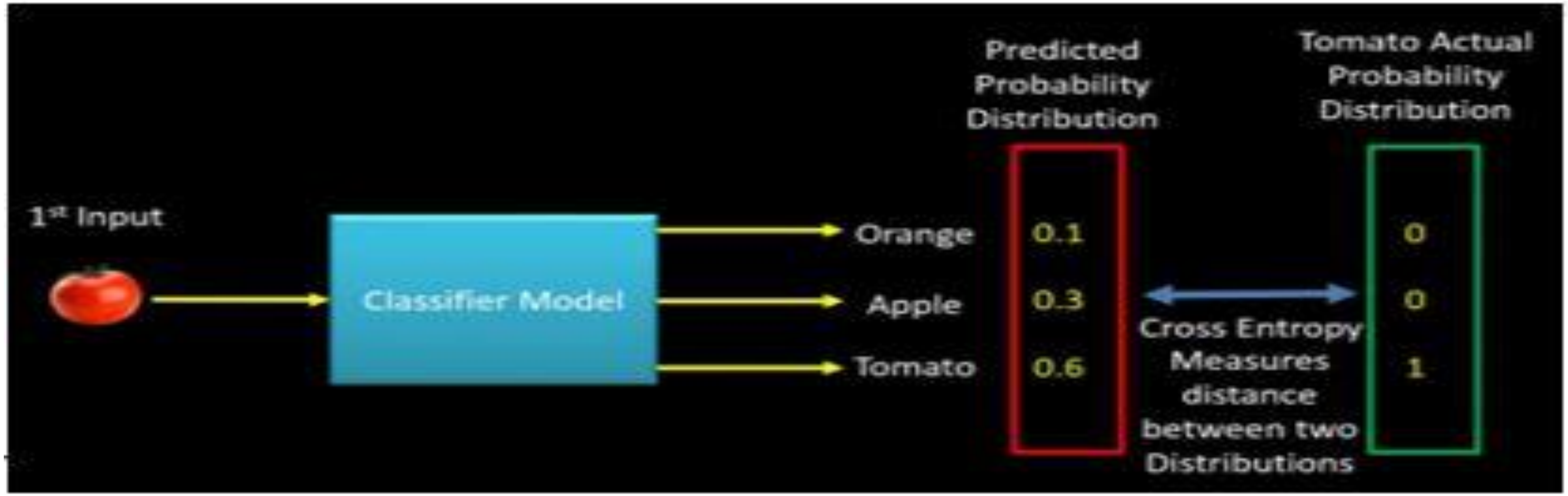

Predicted Probability Distribution
Tomato Actual Probability Distribution
1st Input
Classifier Model
Orange
Apple
Tomato
0.1
0.3
0.6
0
0
1
Cross Entropy Measures distance between two Distributions

Fig. Intuition behind cross-entropy

2.1 Multi-class Classification cost Functions

This cost function is used in the classification problems where there are multiple classes and input data belongs to only one class - how **cross-entropy** is calculated - Let us assume that the model gives the **probability distribution as below for ‘n’ classes & for a particular input data D.**

$P(D) = \begin{bmatrix} p1 \\ p2 \\ pn \end{bmatrix}$ And **the actual or target probability distribution of the data D is** $Y(D) = \begin{bmatrix} y1 \\ y2 \\ yn \end{bmatrix}$

Then cross-entropy for that particular data D is calculated as

Cross-entropy loss(y,p) = – y^T log(p)

= -(y_1 log(p_1) + y_2 log(p_2) + ……y_n log(p_n))

$$\therefore \text{Cross-entropy loss}(y,p) = -\,[y1\ y2\ yn] \begin{bmatrix} \log(p1) \\ \log(p2) \\ \log(pn) \end{bmatrix}$$

- Let us now define the **cost function using the above example (Refer cross entropy image),**

p(Tomato) = [0.1, 0.3, 0.6]

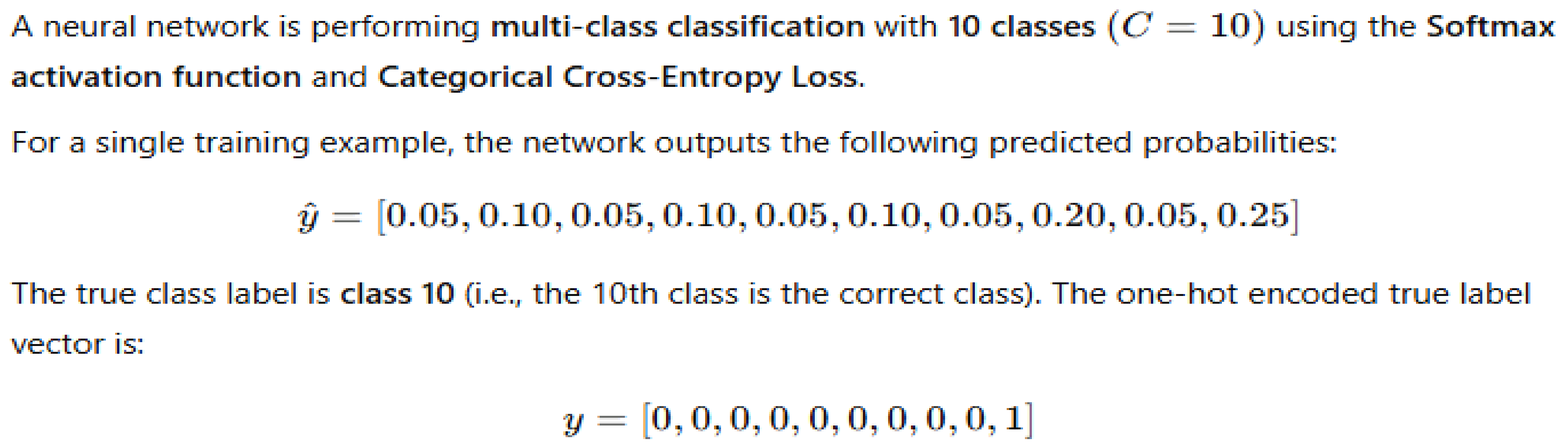
y(Tomato) = [0, 0, 1]

Cross-Entropy(y,P) = – (0*Log(0.1) + 0*Log(0.3)+1*Log(0.6)) = 0.51

- The above formula just measures the **cross-entropy for a single observation or input data**. **The error in classification for the complete model is given by categorical cross-entropy which is nothing but the mean of cross-entropy for all N training data.**

Categorical Cross-Entropy = (Sum of Cross-Entropy for N data)/N

Question :


A neural network is performing **multi-class classification** with **10 classes** $(C = 10)$ using the **Softmax activation function** and **Categorical Cross-Entropy Loss**.

For a single training example, the network outputs the following predicted probabilities:

$$\hat{y} = [0.05, 0.10, 0.05, 0.10, 0.05, 0.10, 0.05, 0.20, 0.05, 0.25]$$

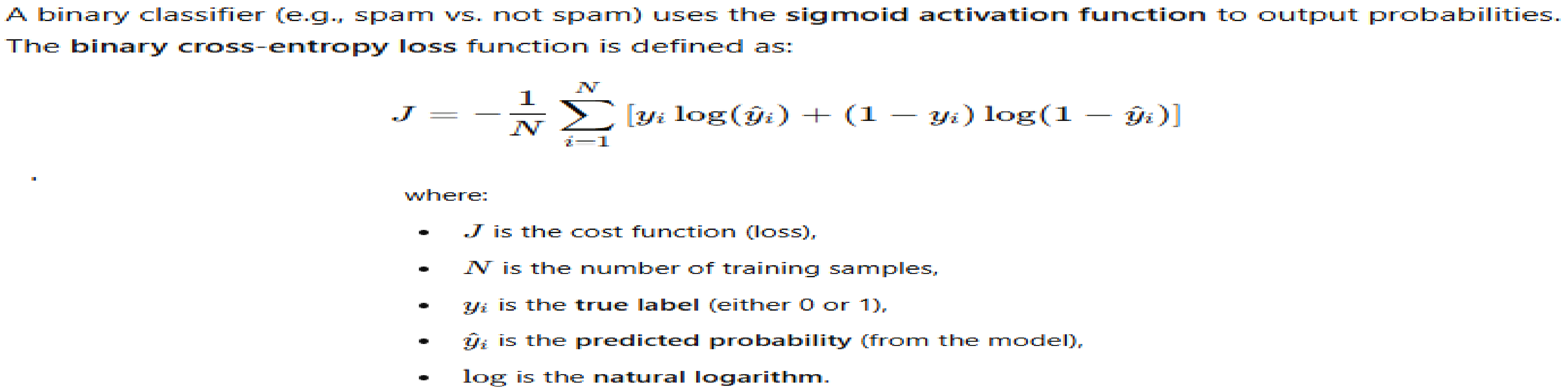
The true class label is **class 10** (i.e., the 10th class is the correct class). The one-hot encoded true label vector is:

$$y = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$$

- Calculate the Cross-Entropy loss
- Calculate the Categorical Cross-Entropy

2.2 Binary Cross Entropy(BCE) - Cost Function

- **Binary cross-entropy is a special case of categorical cross-entropy when there is only one output that just assumes a binary value of 0 or 1 to denote negative and positive class respectively.** For example-classification between cat & dog.



A binary classifier (e.g., spam vs. not spam) uses the **sigmoid activation function** to output probabilities. The **binary cross-entropy loss** function is defined as:

$$J = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

where:

- J is the cost function (loss),
- N is the number of training samples,
- y_i is the **true label** (either 0 or 1),
- $\hat{y}_i$ is the **predicted probability** (from the model),
- log is the **natural logarithm**.

- The **error in binary classification for the complete model is given by binary cross-entropy** which is nothing but the mean of cross-entropy for all N training data.

  Binary Cross-Entropy = (Sum of Cross-Entropy for N data)/N

- **Cost Function in Machine learning ,is a way to measure how well a model predicts outcomes. It’s crucial because it tells us how accurate our predictions are and guides us in improving the model’s performance**.

Problem:

Step 1: Given Data

We have the **true labels** (y) and the **predicted probabilities** ($\hat{y}$) from the model:

Sample	True Label (y)	Predicted Probability ($\hat{y}$)
1	1	0.9
2	0	0.2
3	1	0.7
4	0	0.4
5	1	0.1
6	0	0.8

Computation of Loss for each samples:

Sample 1: $y_1 = 1, \hat{y}_1 = 0.9$

$$J_1 = -\log(0.9) \approx 0.105$$

Sample 2: $y_2 = 0, \hat{y}_2 = 0.2$

$$J_2 = -\log(1 - 0.2) = -\log(0.8) \approx 0.223$$

Sample 3: $y_3 = 1, \hat{y}_3 = 0.7$

$$J_3 = -\log(0.7) \approx 0.357$$

Sample 4: $y_4 = 0, \hat{y}_4 = 0.4$

$$J_4 = -\log(1 - 0.4) = -\log(0.6) \approx 0.511$$

Sample 5: $y_5 = 1, \hat{y}_5 = 0.1$

$$J_5 = -\log(0.1) \approx 2.302$$

Sample 6: $y_6 = 0, \hat{y}_6 = 0.8$

$$J_6 = -\log(1 - 0.8) = -\log(0.2) \approx 1.609$$

Compute the Final Loss (Average BCE)

$$J = \frac{1}{6}(0.105 + 0.223 + 0.357 + 0.511 + 2.302 + 1.609)$$

$$J = \frac{5.107}{6} = 0.851$$

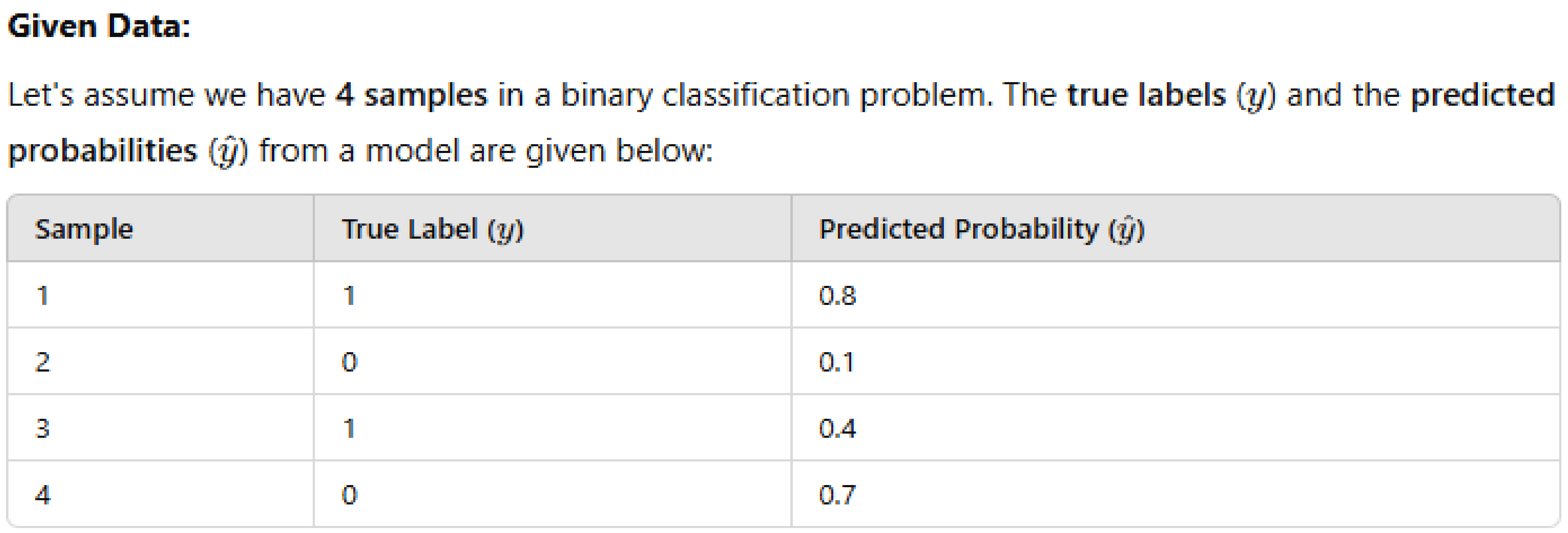
Key Takeaways

1. **BCE is lower when predictions are confident and correct** (i.e., close to 1 for $y = 1$ and close to 0 for $y = 0$).
2. **BCE is higher when predictions are uncertain or incorrect** (e.g., predicting 0.5 for any class leads to a higher loss).
3. **It is commonly used in logistic regression and deep learning for binary classification problems.**
4. **The logarithm function ensures that incorrect predictions are penalized more strongly** than slightly inaccurate predictions.

Key Observations

1. **Sample 5 has the highest loss (2.302)** since the model predicted 0.1 for a true label of 1 (very incorrect).
2. **Sample 1 has the lowest loss (0.105)** since the model predicted 0.9 for a true label of 1 (very accurate).
3. The **total BCE loss is an average of all samples**, and a lower value indicates better model performance.

Question: Calculate the Loss and **Binary Cross Entropy(BCE) and write the inference – for the following data.**



Given Data:

Let's assume we have **4 samples** in a binary classification problem. The **true labels** (y) and the **predicted probabilities** ($\hat{y}$) from a model are given below:

Sample	True Label (y)	Predicted Probability ($\hat{y}$)
1	1	0.8
2	0	0.1
3	1	0.4
4	0	0.7

Nearest neighbor methods

Nearest neighbor methods are a set of **non-parametric machine learning algorithms used for classification and regression.** These methods rely on the idea that **similar data points exist in close proximity to each other in feature space.** The most common nearest neighbor method is **k-Nearest Neighbors (k-NN)**.

Key Concepts of Nearest Neighbor Methods

1. **Instance-Based Learning**: Nearest neighbor methods do not build an explicit model during training. Instead, they **store the training data and use it directly for making predictions.**
2. **Distance Metric**: The algorithm measures similarity using a **distance metric**, such as:
    1. Euclidean Distance (most common)
    2. Manhattan Distance
    3. Minkowski Distance
    4. Cosine Similarity (useful for text and high-dimensional data)
3. **Choice of k (in k-NN)**:
    1. **k = 1** → The prediction is based on the **single closest neighbor**.
    2. **Higher k values** → **More robust to noise** but may **lose finer details**.

Types of Nearest Neighbor Methods

1. **k-Nearest Neighbors (k-NN)**:
    1. For **classification**: Assigns the most common label among k nearest neighbors.
    2. For **regression**: Takes the **average (or weighted average) of the target values** of k nearest neighbors.
2. **Radius Nearest Neighbors (RNN)**:
    1. Considers all points within a **fixed radius instead of a fixed number (k).**
    2. Useful when **data density varies** across different regions.
3. **Weighted k-NN**:
    1. Gives **more weight to closer neighbors** using functions like inverse distance weighting.

Advantages of Nearest Neighbor Methods

- Simple and easy to implement.
- Works well with **small datasets.**
- **No training phase**, making it useful for online learning.

Disadvantages

- **Computationally expensive for large datasets** (since it requires storing and searching the entire dataset).
- Sensitive to irrelevant or redundant features.
- **Performance heavily depends on the choice of distance metric and k.**

K-Nearest Neighbor(KNN) Algorithm

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on **Supervised Learning technique.**

- K-NN algorithm **stores all the available data and classifies a new data point based on the similarity**. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

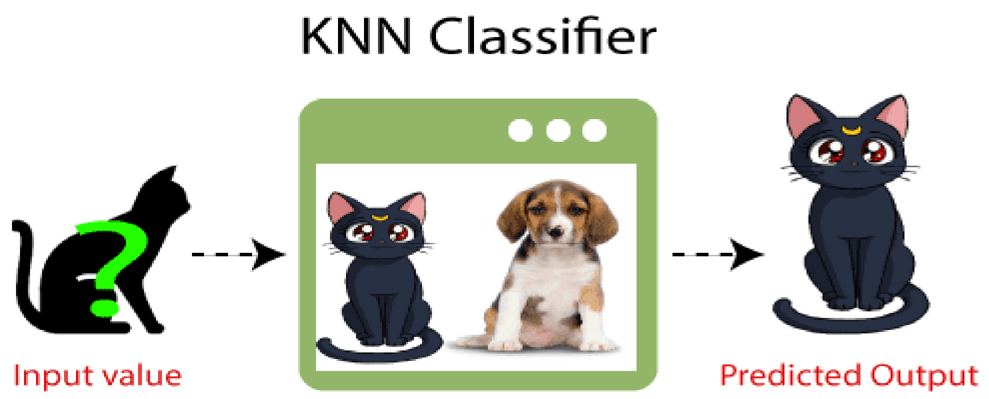
- **K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.**

- **It is also called 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**.

- **"k" represents the number of nearest neighbors that are considered when classifying a new data point**; essentially, it defines how many neighboring data points are used to make a prediction about the class of a new data point.

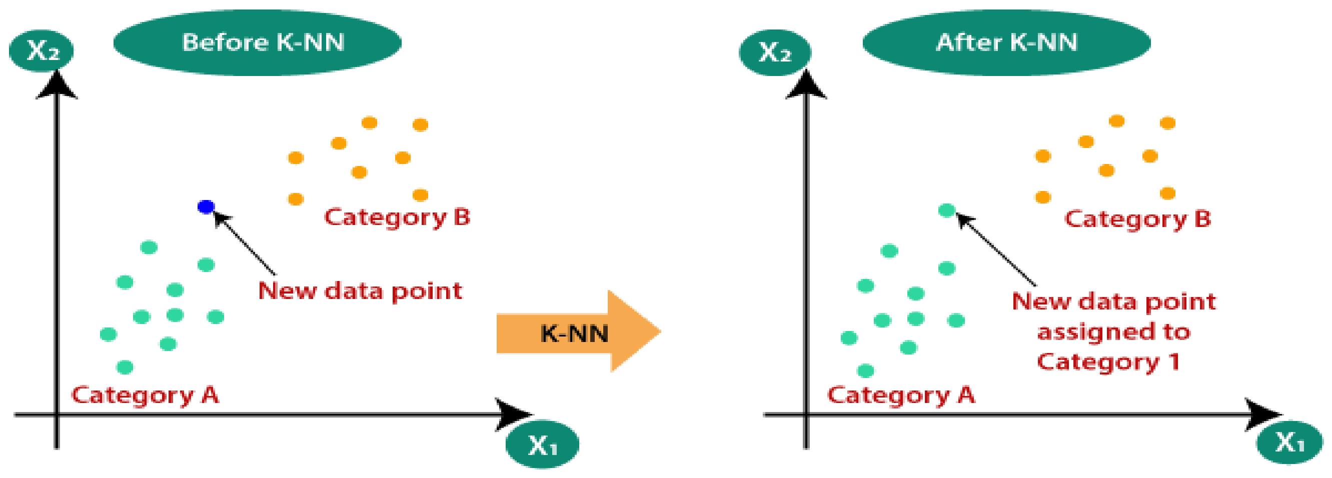
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

- **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.**



Why do we need a K-NN Algorithm?

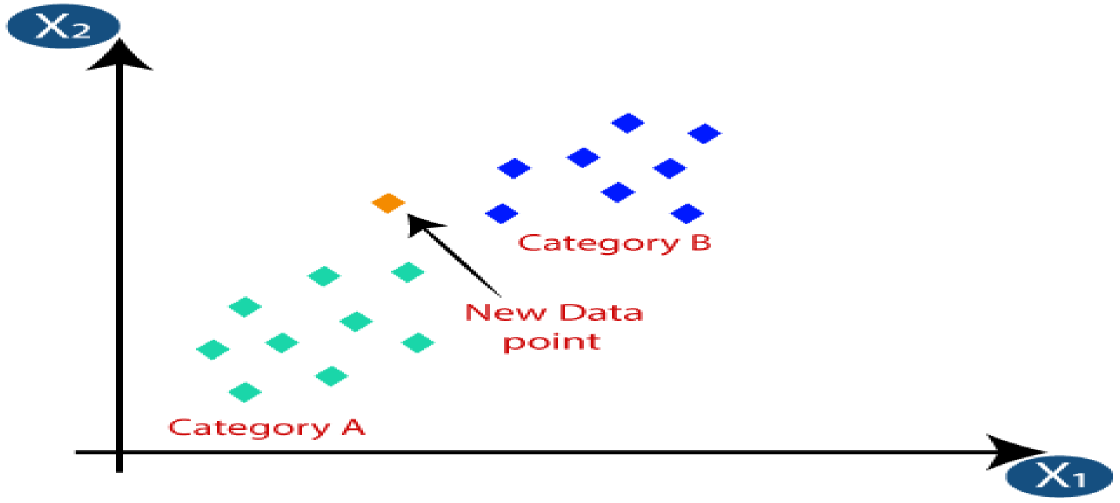
Suppose there are two categories, i.e., **Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories.** To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the diagram:



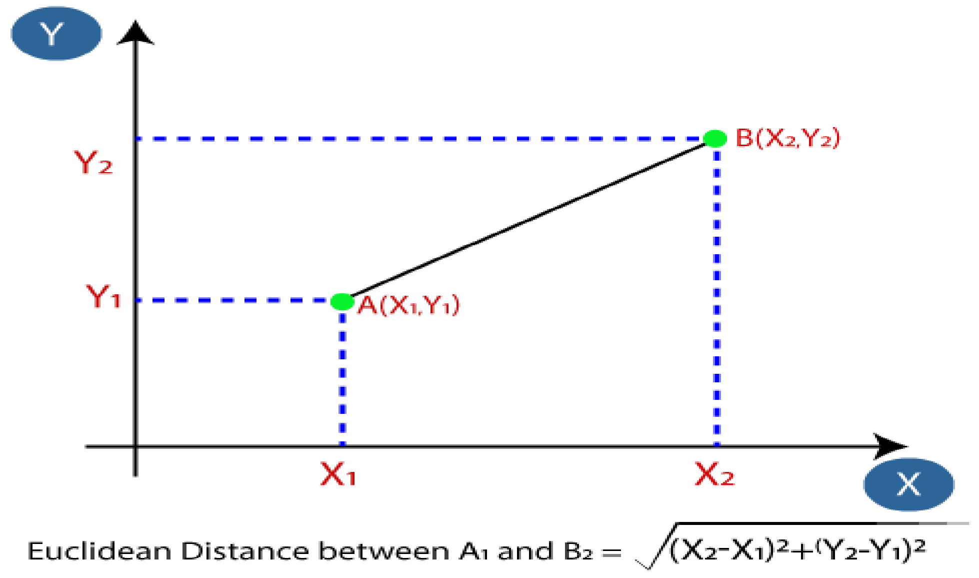
Working of K-NN algorithm:

- **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:



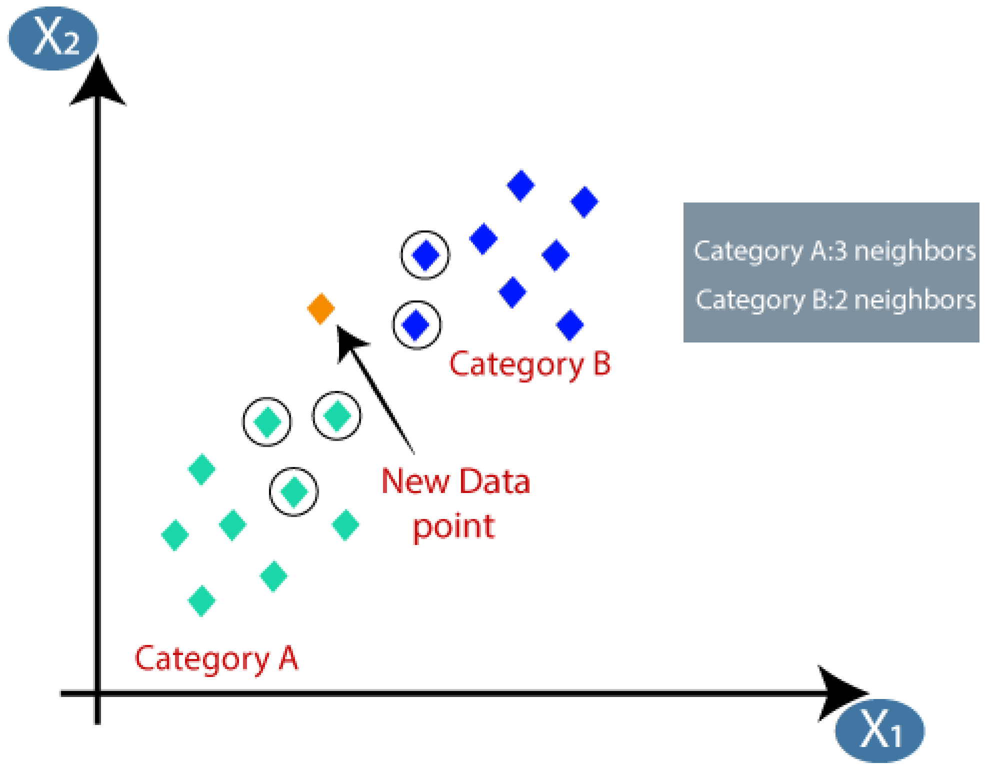
- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



Y
Y_2
Y_1
$B(X_2,Y_2)$
$A(X_1,Y_1)$
X_1
X_2
X
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.

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?

- **The k value in the k-NN algorithm defines how many neighbors will be checked to determine the classification of a specific query point**.
- **There is 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.**

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.

Example 1 :

Perform KNN classification algorithm on the following dataset and predict the class for, X= (Maths =6, CS=8), K=3

S.No	Maths	CS	Result
(i)	4	3	Fail
(ii)	6	7	Pass
(iii)	7	8	Pass
(iv)	5	5	Fail
(v)	8	8	Pass

1. Calculation of Euclidean distance

d= $\sqrt{(x_{o1} \_ x_{a1})^2} + (x_{o1} \_ x_{a1})^2$

(i) $\sqrt{(6-4)^2} + (8-3)^2$

= $\sqrt{29} = 5.38$

(ii) 1

(iii) 1

(iv) 3.16

(v) 2

2. Based on the distance calculation [(ii), (iii), (v)] are nearby to each other – with k=3 (since the value of K=3 – derive 3 nearest neighbors)

3 (pass) > 0 (fail)

Therefore, X= (Maths =6, CS=8) = Pass

Question 1:

S.No	P1	P2	Class
(i)	7	7	False
(ii)	7	4	False
(iii)	3	4	True
(iv)	1	4	True

X (p1=3 and p2 =7) and k =3

Question 2: Apply KNN and predict the type of food it belongs to, Tomato(sweet = 6, crunch =4)

Ingredient	Sweet	Chrunch	Food type
Grape	8	5	Fruit
Greenbean	3	7	Vegetables
Nuts	3	6	Protein
Orange	7	3	Fruit

D(Tomato,Grape)

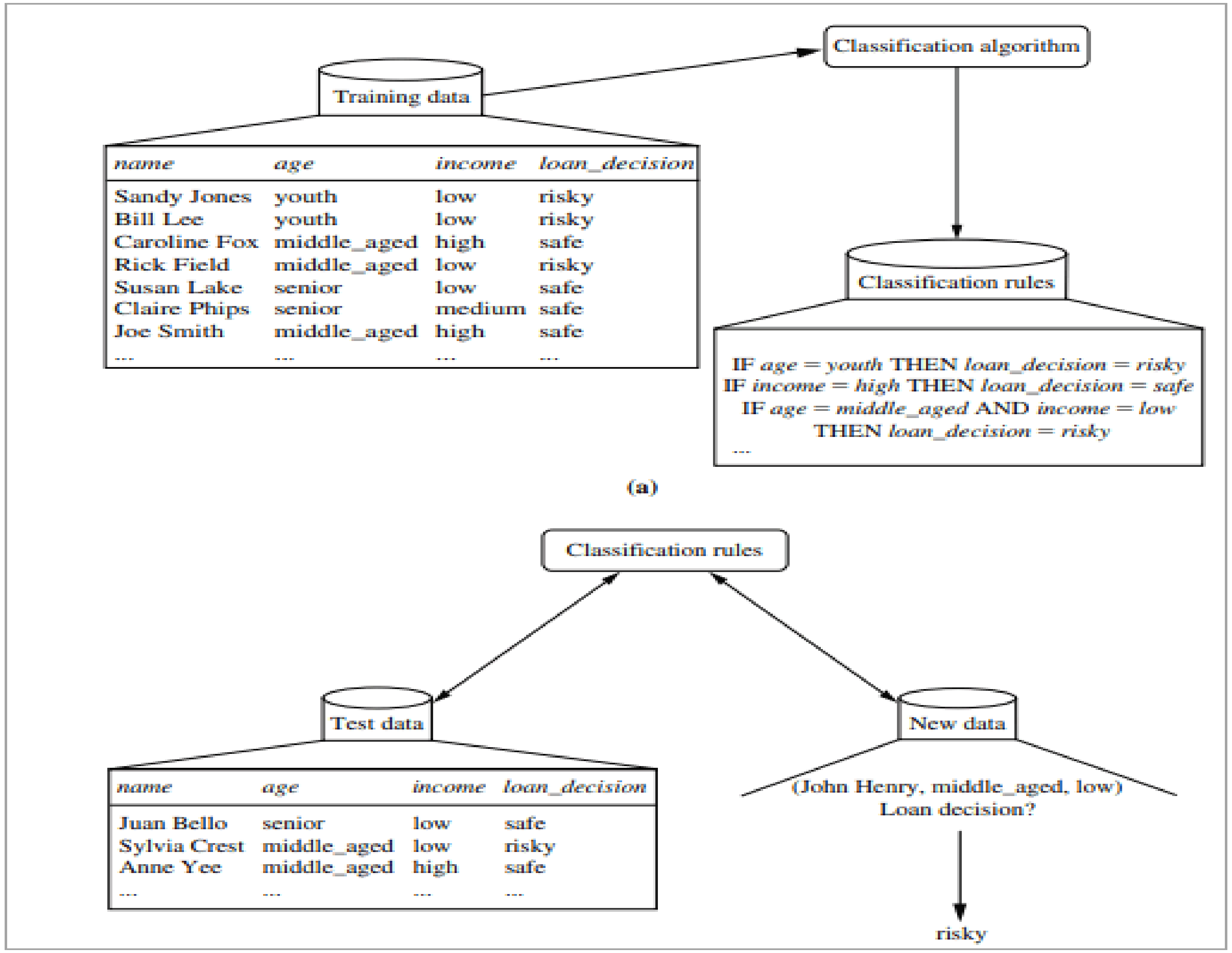
D(Tomato,Greenbean) $\longrightarrow$ Find – whose distance is Minimum

D(Tomato, Nuts)

D(Tomato, Orange)

Classification

- Classification, where a model or classifier is constructed to predict class (categorical) Labels
- Training Phase & Testing Phase
- Supervised learning and unsupervised learning
- The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier



name	*age*	*income*	*loan_decision*
Sandy Jones	youth	low	risky
Bill Lee	youth	low	risky
Caroline Fox	middle_aged	high	safe
Rick Field	middle_aged	low	risky
Susan Lake	senior	low	safe
Claire Phips	senior	medium	safe
Joe Smith	middle_aged	high	safe
...	...	...	...

IF *age* = *youth* THEN *loan_decision* = *risky*
IF *income* = *high* THEN *loan_decision* = *safe*
IF *age* = *middle_aged* AND *income* = *low* THEN *loan_decision* = *risky*
...

name	*age*	*income*	*loan_decision*
Juan Bello	senior	low	safe
Sylvia Crest	middle_aged	low	risky
Anne Yee	middle_aged	high	safe
...	...	...	...

- **Classification** is the process of organizing objects or events into groups or categories. It helps us understand the relationships between things.

The Need of Classification

Taking an example of Species and Organisms – for classifications:

- **Identify and differentiate**: Classification helps us identify and differentiate closely related species.
- **Understand variation**: Classification helps us understand the variation among species.
-
- **Understand evolution**: Classification helps us understand how species evolved from simpler organisms.
- **Study organisms**: Classification makes it easier to study different kinds of organisms.
- **Understand relationships**: Classification helps us understand the inter-relationships among organisms.
- **Find objects**: Classification helps us find an individual object quickly based on its kind or group.
- **Detect duplicates**: Classification makes it easier to detect duplicate objects.
- **Improve utility**: Classification brings out the similarity in different sets of data, which enhances its utility.

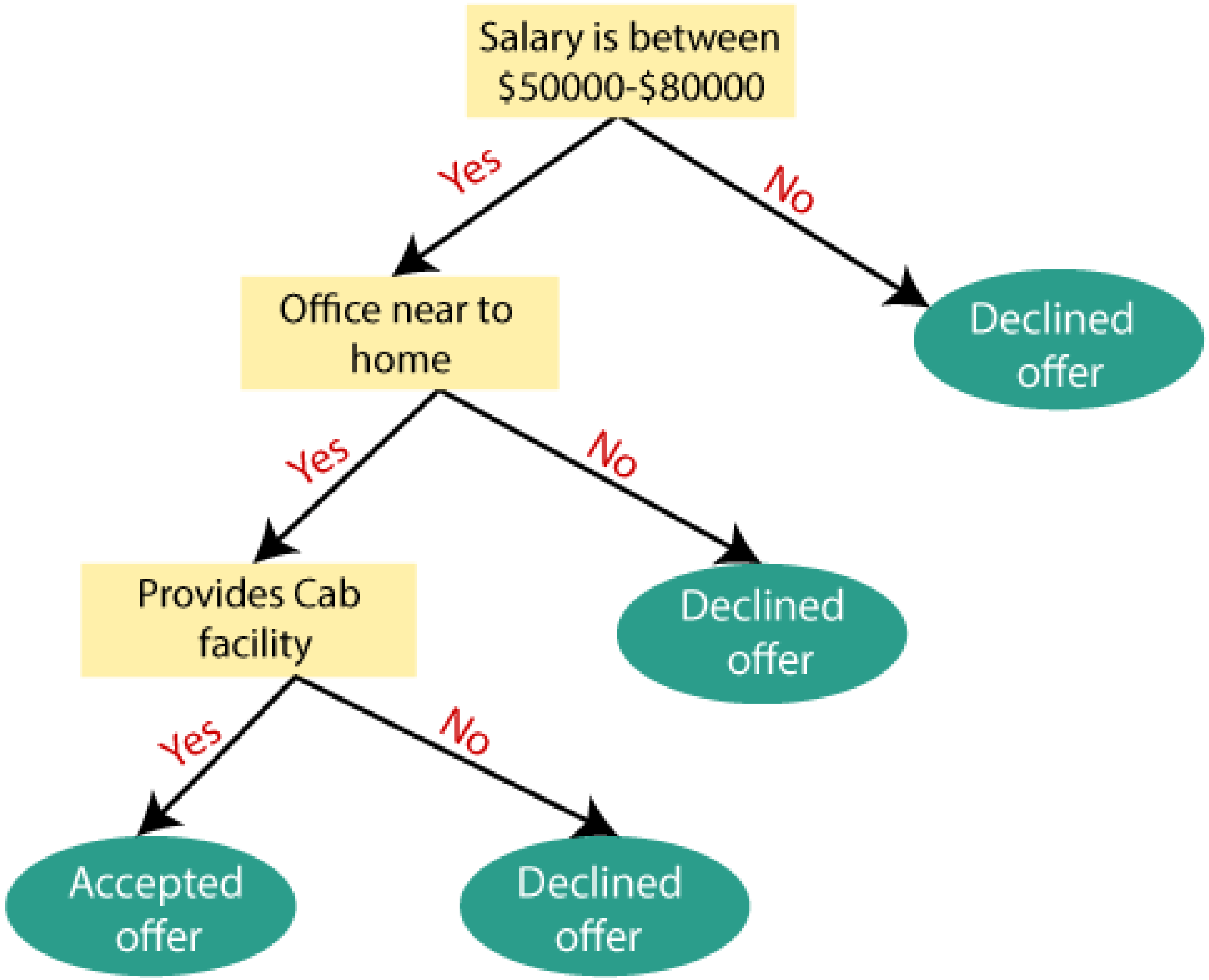
Decision Tree Induction

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

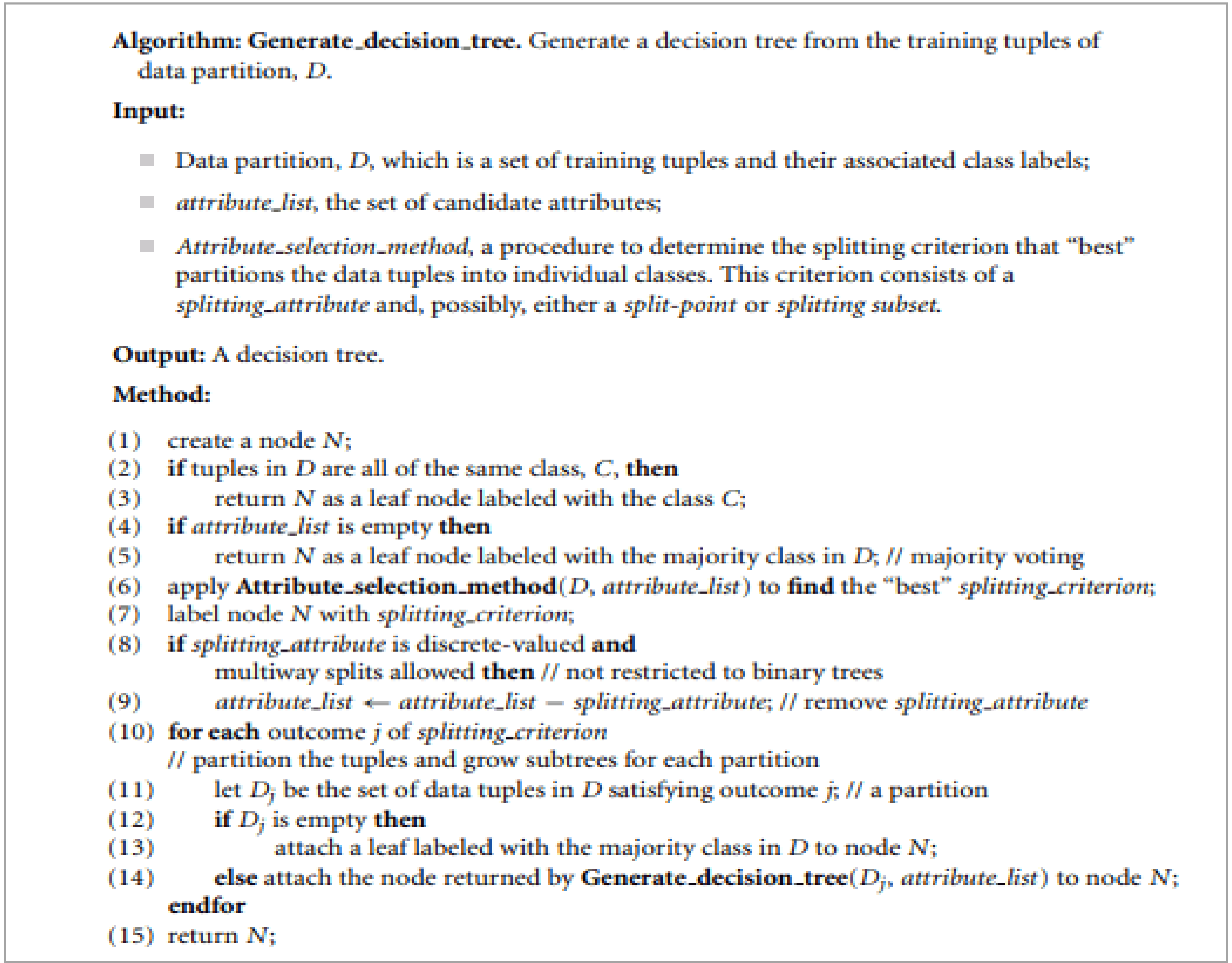
- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. **It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.**

- A decision tree simply asks a question, and **based on the answer (Yes/No), it further split the tree into subtrees.**

Example: Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not.



Algorithm:



Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset*.

Output: A decision tree.

Method:

```
(1)  create a node $N$;
(2)  if tuples in $D$ are all of the same class, $C$, then
(3)       return $N$ as a leaf node labeled with the class $C$;
(4)  if attribute_list is empty then
(5)       return $N$ as a leaf node labeled with the majority class in $D$; // majority voting
(6)  apply Attribute_selection_method($D$, attribute_list) to find the "best" splitting_criterion;
(7)  label node $N$ with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
          multiway splits allowed then // not restricted to binary trees
(9)       attribute_list $\leftarrow$ attribute_list $-$ splitting_attribute; // remove splitting_attribute
(10) for each outcome $j$ of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)      let $D_j$ be the set of data tuples in $D$ satisfying outcome $j$; // a partition
(12)      if $D_j$ is empty then
(13)           attach a leaf labeled with the majority class in $D$ to node $N$;
(14)      else attach the node returned by Generate_decision_tree($D_j$, attribute_list) to node $N$;
     endfor
(15) return $N$;
```

Attribute Selection Measures

- While implementing a Decision tree, the **main issue arises that how to select the best attribute for the root node** and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. **The techniques for ASM, is**

- **Information Gain**

Information Gain:

- Information gain is the **measurement of changes in entropy, after the segmentation of a dataset based on an attribute.**
- **It calculates how much information a feature provides us about a class.**
- **According to the value of information gain, we split the node and build the decision tree.**
- **A decision tree algorithm always tries to maximize the value of information gain**, and a node/attribute having the **highest information gain is split first.**

Decision tree :

Example 1:

Age	Competition	Type	Profit
Old	Yes	S/W	Down
Old	No	S/W	Down
Old	No	H/W	Down
Mid	Yes	S/W	Down
Mid	Yes	H/W	Down
Mid	No	H/W	Up
Mid	No	S/W	Up
New	Yes	S/W	Up
New	No	H/W	Up
New	No	S/W	Up

1. Calculate the Information Gain – for target class/variables (P-Down, N-Up)

IG = $-P/P+N \log_2 (P/P+N) - N/P+N \log_2(N/P+N)$

$= - [5/10.\log_2 (5/10) + 5/10.\log_2 (5/10)]$

$= - [0.5.\log_2 2^{-1} + 0.5.\log_2 2^{-1}]$

$= -[-0.5 \ -0.5] = -[-1]$ $\boxed{IG = 1}$

2. Calculate the entropy (for rest of the attributes) - Multiplication of IG (ex. Age) and its probability

	Down	Up
Old	3	0
Mid	2	2
New	0	3

IG(old) = $- [3/3. \log_2 (3/3) + 0/3 \log_2 (0/3)] = 0 \text{ X } 3/10 = 0$

IG(Mid) = $- [2/4. \log_2 (2/4) + 2/4 \log_2 (2/4)] = 1 \text{ X } 4/10 = 0.4$

IG(New) = $- [0/3. \log_2 (0/3) + 3/3 \log_2 (3/3)] = 0 \text{ X } 3/10 = 0$

Entropy (add)– E(Age) = 0.4

3. Gain = IG-E(Age)=1-0.4 = 0.6

Dr.Priya Govindarajan

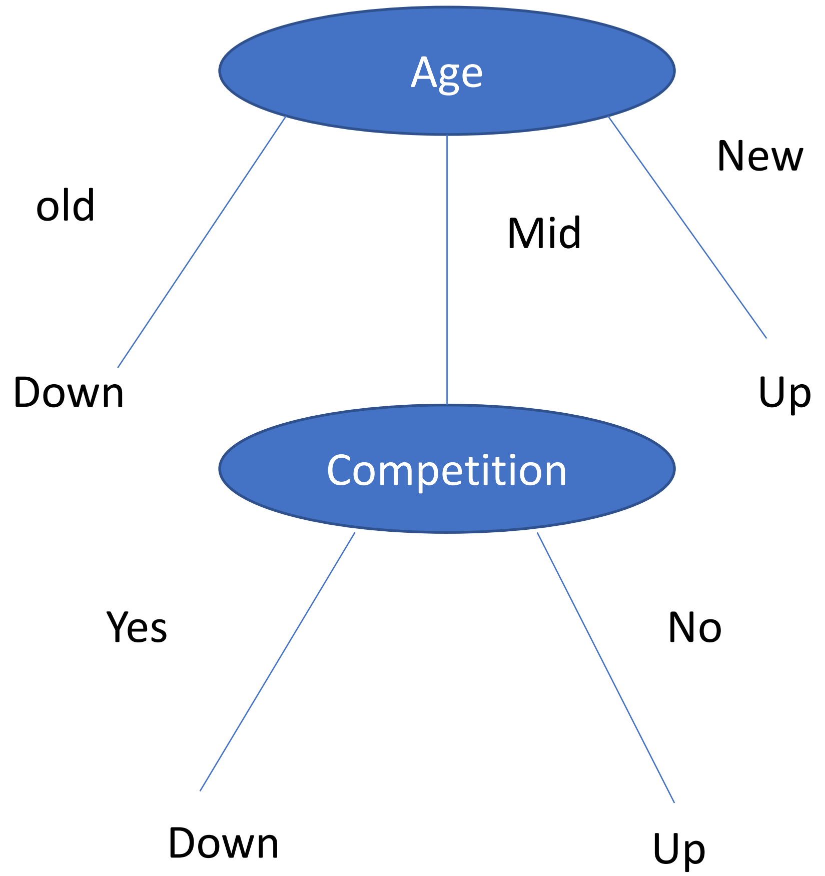
- Calculate the entropy and gain for rest of the attributes (i.e., competition and type)

- Whichever attribute is having the **highest gain, will be the root node of the DT**

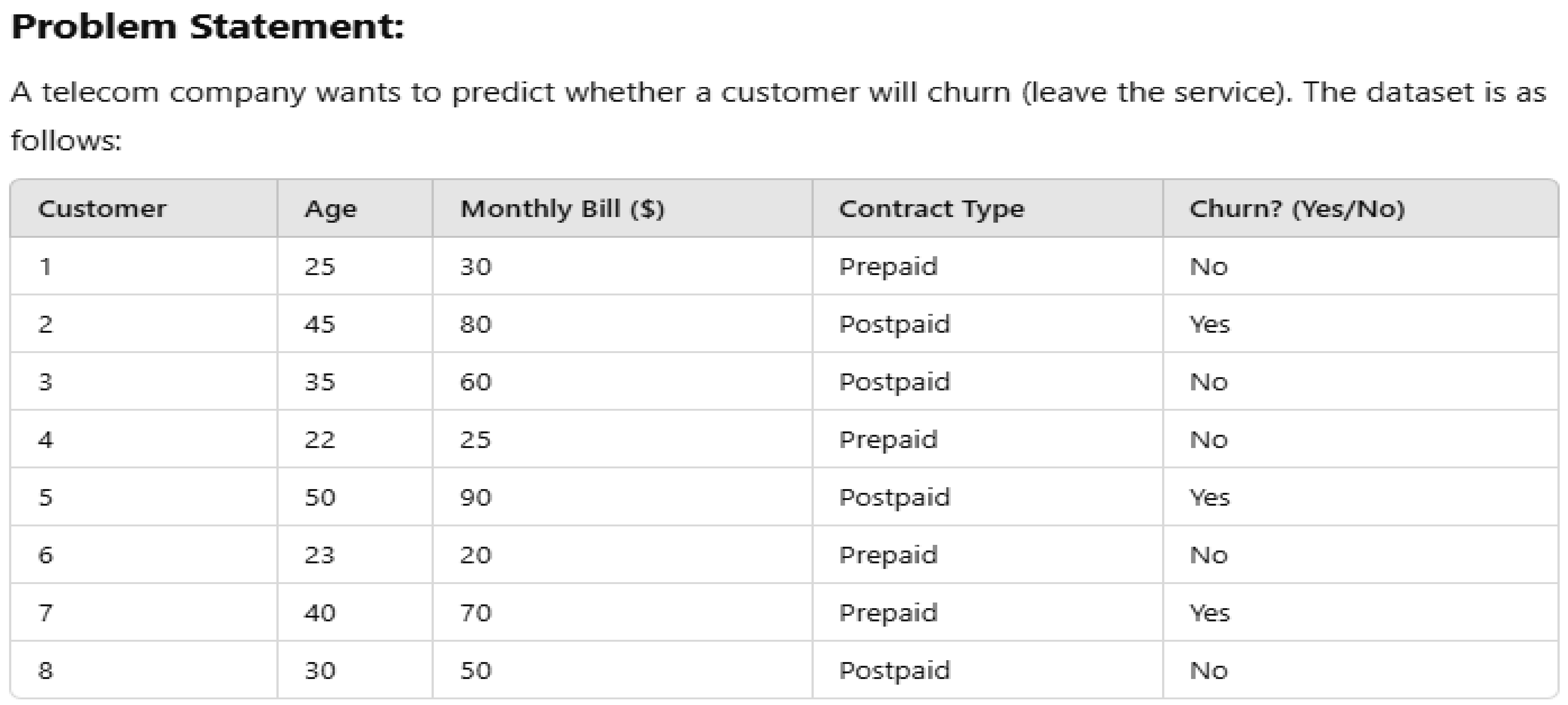
- Construction of Decision tree

- When an attribute – belongs to different set of variants, then bring in another attribute – based on the (highest) Gain value.

- In terms of competition,
  Yes – pointing to one variant (down)
  No – pointing to one variant (up)
  - in this case, no need to bring in Type
  - (And) also Type- Gain – is zero (0)



Questions :

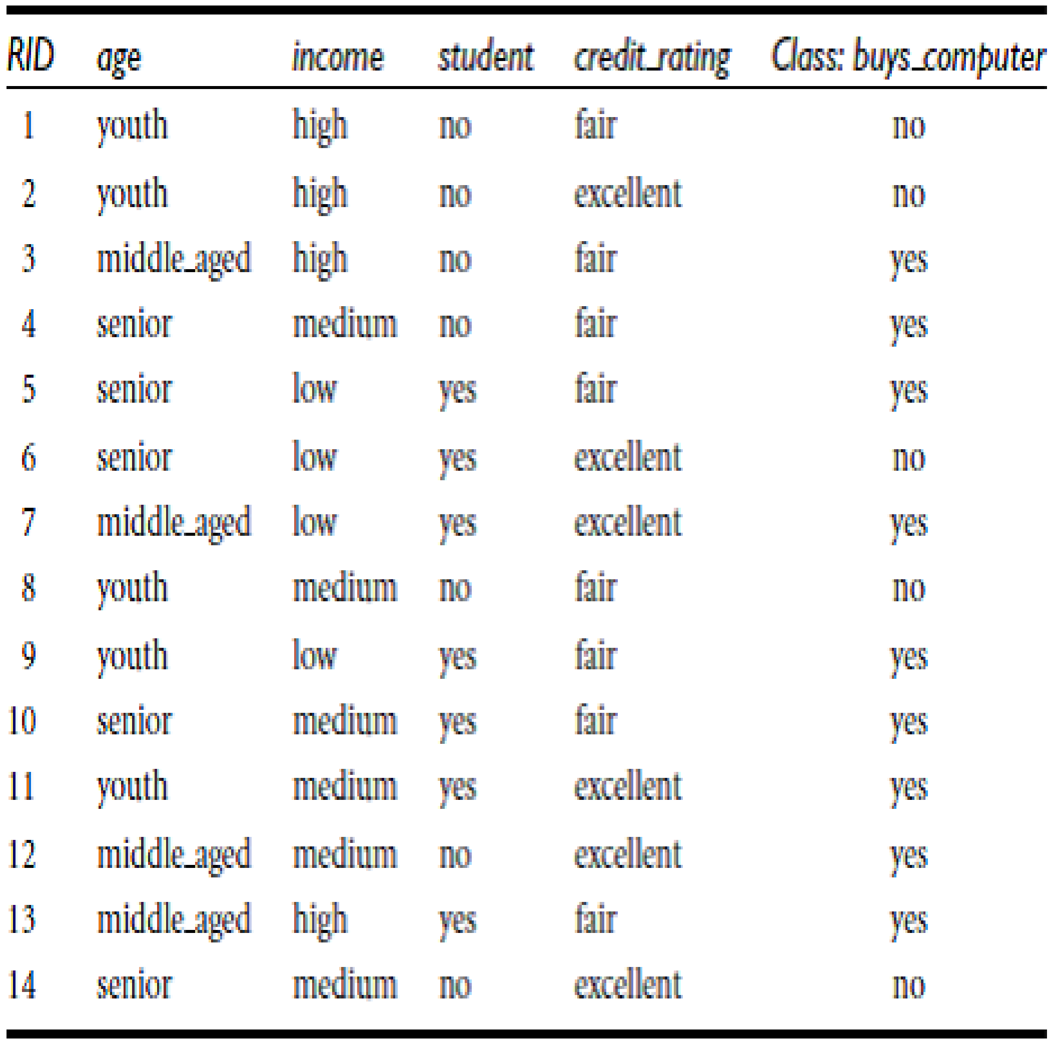


Problem Statement:

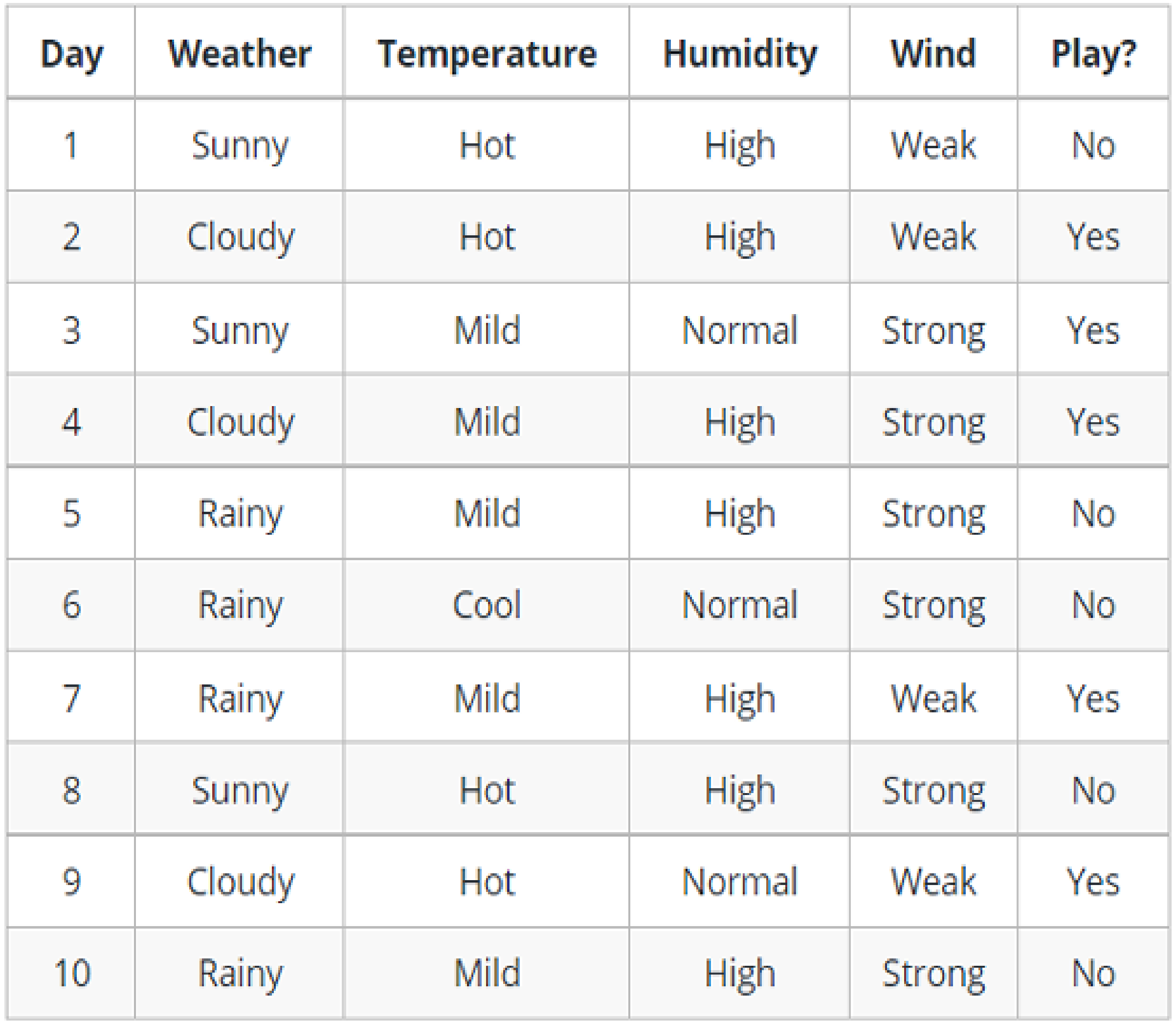
A telecom company wants to predict whether a customer will churn (leave the service). The dataset is as follows:

Customer	Age	Monthly Bill ($)	Contract Type	Churn? (Yes/No)
1	25	30	Prepaid	No
2	45	80	Postpaid	Yes
3	35	60	Postpaid	No
4	22	25	Prepaid	No
5	50	90	Postpaid	Yes
6	23	20	Prepaid	No
7	40	70	Prepaid	Yes
8	30	50	Postpaid	No

Questions:



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



Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No

Tree Pruning

Pruning is the procedure that decreases the size of decision trees. It can **decrease the risk of overfitting** by defining the size of the tree or **eliminating areas of the tree that support little power**. Pruning - supports by **trimming the branches that follow anomalies in the training information because of noise or outliers and supports the original tree in a method that enhances the generalization efficiency of the tree.**

Pre-pruning Approach

In the pre-pruning approach, **a tree is "pruned" by laboring its construction early** (e.g., **by determining not to further divide or partition the subset of training samples at a provided node). Upon halting, the node turns into a leaf**. The leaf can influence the most common class between the **subset samples, or the probability distribution of those samples.**

If partitioning the samples at a node can result in a split that declines **below a pre-specified threshold, then partitioning of the given subset is halted. There are problems in selecting an appropriate threshold. High thresholds can result in oversimplified trees, while low thresholds can result in very little simplification.**

Post-pruning Approach

- **The post-pruning approach eliminates branches from a “completely grown” tree**. **A tree node is pruned by eliminating its branches.** The price complexity pruning algorithm is an instance of the post-pruning approach. **The pruned node turns into a leaf and is labeled by the most common class between its previous branches.**

- For each **non-leaf node in the tree, the algorithm computes the expected error rate** that can appear if the subtree at that node were shortened. Next, **the expected error rate appearing if the node were not pruned is computed using the error rates for each branch**, connected by weighting according to the dimension of observations along each branch. If pruning the node leads to a higher expected error rate, then the subtree is preserved. Therefore, it is pruned.

- After creating a set of increasingly pruned trees, an independent test set can estimate the efficiency of each tree. **The decision tree that diminishes the expected error cost is preferred**.