Fundamentals of Machine Learning

Chapter 2: Supervised Learning

Target Group: G3SE A & B

Instructor: Melaku M.

What is Supervised learning?

❖ **Approach**:

- Given a collection of records (*training set*)
  - each record contains a set of *attributes,* one of the attributes is the *class attribute (label).*
  - *Dataset* $D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$

❖ **Task:** Find a *model* for class attribute as a function of the values of other attributes.

❖ **Goal**: Previously unseen records should be assigned a class as accurately as possible.

Examples of supervised learning Tasks

- Classifying credit card transactions as legitimate or fraudulent
  - Attributes: your age, income, debts, …
  - Class: are you getting credit by your bank?
- Marketing
  - Attributes: previously bought products, browsing behavior
  - Class: are you a target customer for a new product?
- SPAM Detection: Categorizing email messages
  - Attributes: words and header fields of an e-mail
  - Class: regular e-mail or spam e-mail?
- Identifying tumor cells: Predicting tumor cells as benign or malignant
  - Attributes: features extracted from x-rays or MRI scans
  - Class: malignant or benign cells
- Categorizing news stories as finance, weather, entertainment, sports, etc.

Supervised learning

There are two major types of supervised machine learning problems.

- Classification
- Regression

1. Regression

- Regression Analysis is a statistical approach for modelling the relationships between the **dependent variables** and **one or more independent variables.**
- In Regression, we plot a **graph** between the **variables,** which **best fit** the given data points. In naïve words, "***Regression shows a line or curve that passes through all the data points on a target-predictor graph in such a way that the vertical distance between the data points and the regression line is minimum***".
- Regression is principally used for prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables.

Predicting House Prices

Problem: Given features like the size of a house, and the number of bedrooms of the house, predict its price.

Independent variable

Dependent variable

Size (sq ft)	Bedrooms	Price ($)
1914	4	245679.32
2859	3	338129.41
1465	2	192672.58
1297	5	194902.16
2433	1	251292.31

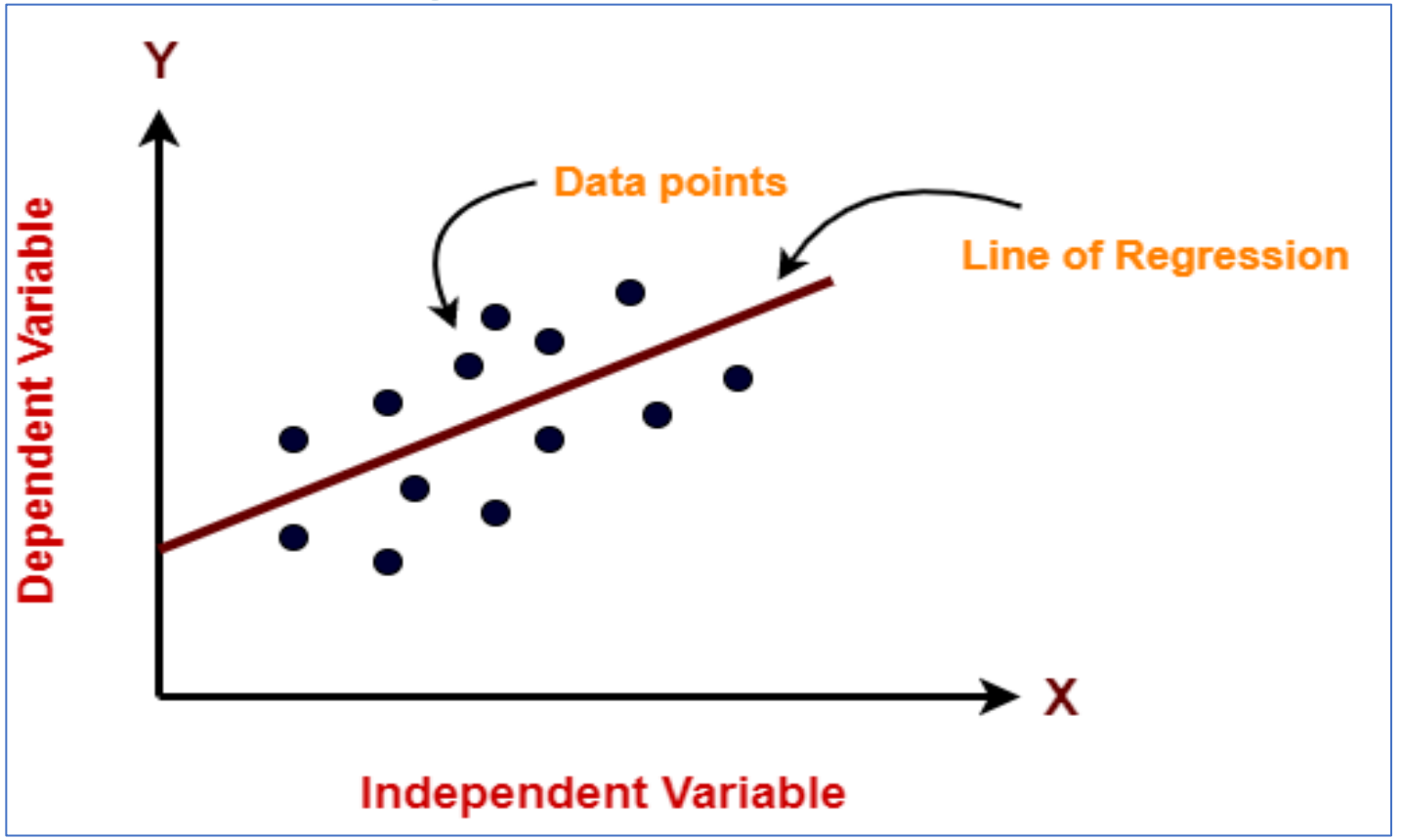
Types of Regression models

- There are several types of regression techniques, each suited for **different types of data** and **different types of relationships**.
- The main types of regression techniques are:
  - Linear Regression
  - Polynomial Regression

Linear Regression

- This is the simplest and most common type of regression algorithm.
- It models the relationship/connection between a dependent variable (what you are trying to predict) and one or more independent variables (the features that influence the dependent variable) using a **linear equation**.
- ***Linear regression*** *is a regression model that uses a straight line to describe the relationship between variables.*
- The goal of the algorithm is to find the **best Fit Line** that passes through most of data points.

Linear Regression



In the figure given above, on **X-axis is the independent variable** and on **Y-axis is the dependent variable(target)**. The regression line (a sloped straight line) is referred to as the best-fit line for a model. And our main objective in this algorithm is to **find this best fit line** based on the given data points.

Types of Linear Regression

1) Simple Linear Regression

- In a simple linear regression, there is one independent variable(x) and one dependent variable(y).
- The model estimates the *regression coefficients(***slope** and **intercept)** of the line of best fit, which represents the relationship between the variables.
- The goal of the linear regression algorithm is to get the **best values for B_0 and B_1** to find the best fit line.

1) Simple Linear Regression

❖**The equation for simple linear regression**

y=mx+b => $\hat{y} = B_0 + B_1 x$

❖where,

y = predicted value. It represents the value that the model tries to predict.

x = feature value

B_0 = bias term or **intercept of the line.**

B_1 = the feature weights or regression coefficient. Slope of best-fit line.

❖B_0, B_1 are known as **Model Parameters**.

2) Multiple Linear Regression

- Multiple linear regression is a technique to model the relationship between a *single* dependent variable and *multiple* independent variables.
- The equation for multiple linear regression is:

  $\hat{y}=\beta_0+\beta_1 x_1+ \beta_2 x_2 +\ldots\ldots\ldots \beta_n x_n$

  where:
  - $\hat{y}$ is the predicted value
  - n is the number of features
  - x_i i^{th} feature value
  - β_0 is the intercept/bias term
  - β_i, i^{th} feature weight value

Hypothesis function in Linear Regression

❖Let us assume that our independent feature is the experience i.e. X and the respective salary Y is the dependent variable. Let's assume there is a linear relationship between X and Y then the salary can be predicted using:

$\hat{Y} = B_0 + B_1 X$ Or $\hat{y}_i = B_0 + B_1 x_i$

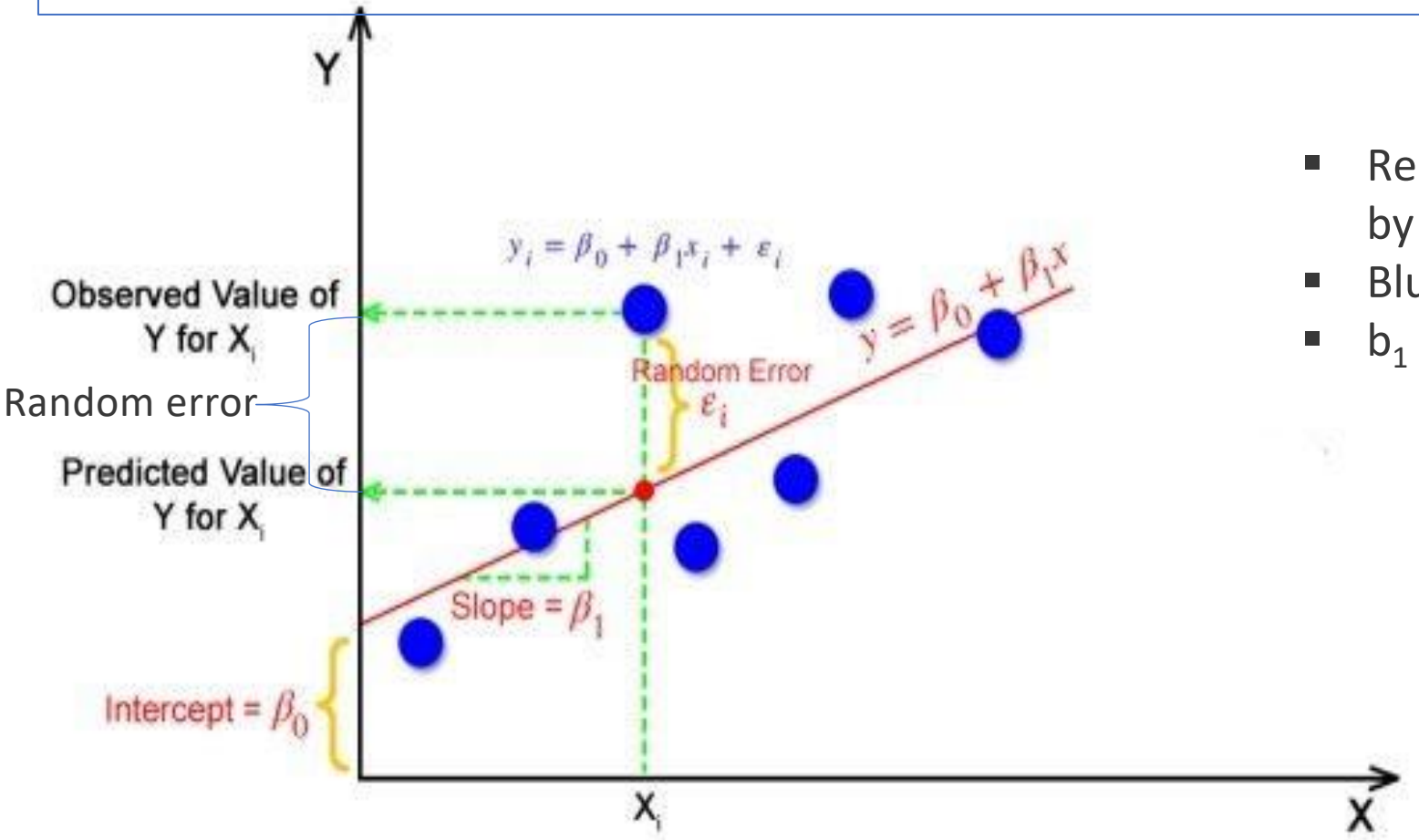
$y_i \epsilon Y (i=1,2,\cdots,n)$ are labels to data

$x_i \epsilon X (i=1,2,\cdots,n)$ are the input independent training data

$\hat{y}_i \epsilon \hat{Y} (i=1,2,\cdots,n)$ are the predicted values.

❖Once we find the best B_0 and B_1 values, we get the best-fit line.

This algorithm explains the linear relationship between the dependent variable y and the independent(predictor) variable X using a straight-line $Y = B_0 + B_1 X$.



- Red line is the best fit line predicted by the model.
- Blue dots are the data points
- b_1 is the slope of the x variable.

The best fit line should have the least error means the error between predicted values and actual values should be minimized.

Cont'd

- **The vertical distance between the data point and the regression line is known as error or residual.** Each data point has one residual and the sum of all the differences is known as **the Sum of Residuals/Errors.**

- **<u>Mathematical Approach:</u>**

- Residual/Error = Actual values – Predicted Values

- Sum of Residuals/Errors = Sum(Actual- Predicted Values)

- Square of Sum of Residuals/Errors = (Sum(Actual- Predicted Values))^2

$$\sum e_i^2 = \sum (Y_i - \hat{Y}_i)^2$$

Cost Function

❖The cost function is nothing but the error or difference between the **actual value** y_i and the **predicted value Y**.

❖In Linear Regression, the **Mean Squared Error (MSE)** cost function is employed, which calculates the **average of the squared errors** between the **actual values** $\hat{y}$ and the **predicted values** y_i.

❖It helps us to figure out **optimal/best values** for B_0 and B_1, which provides the best fit line for the given data points.

- Our MSE tells us how well (or rather how poorly) our function fits our data. This means that we want to make our MSE as low as possible.
- Our MSE is just a function, we can compute the minimum of it directly by taking the first derivative of the MSE.

Cost Function

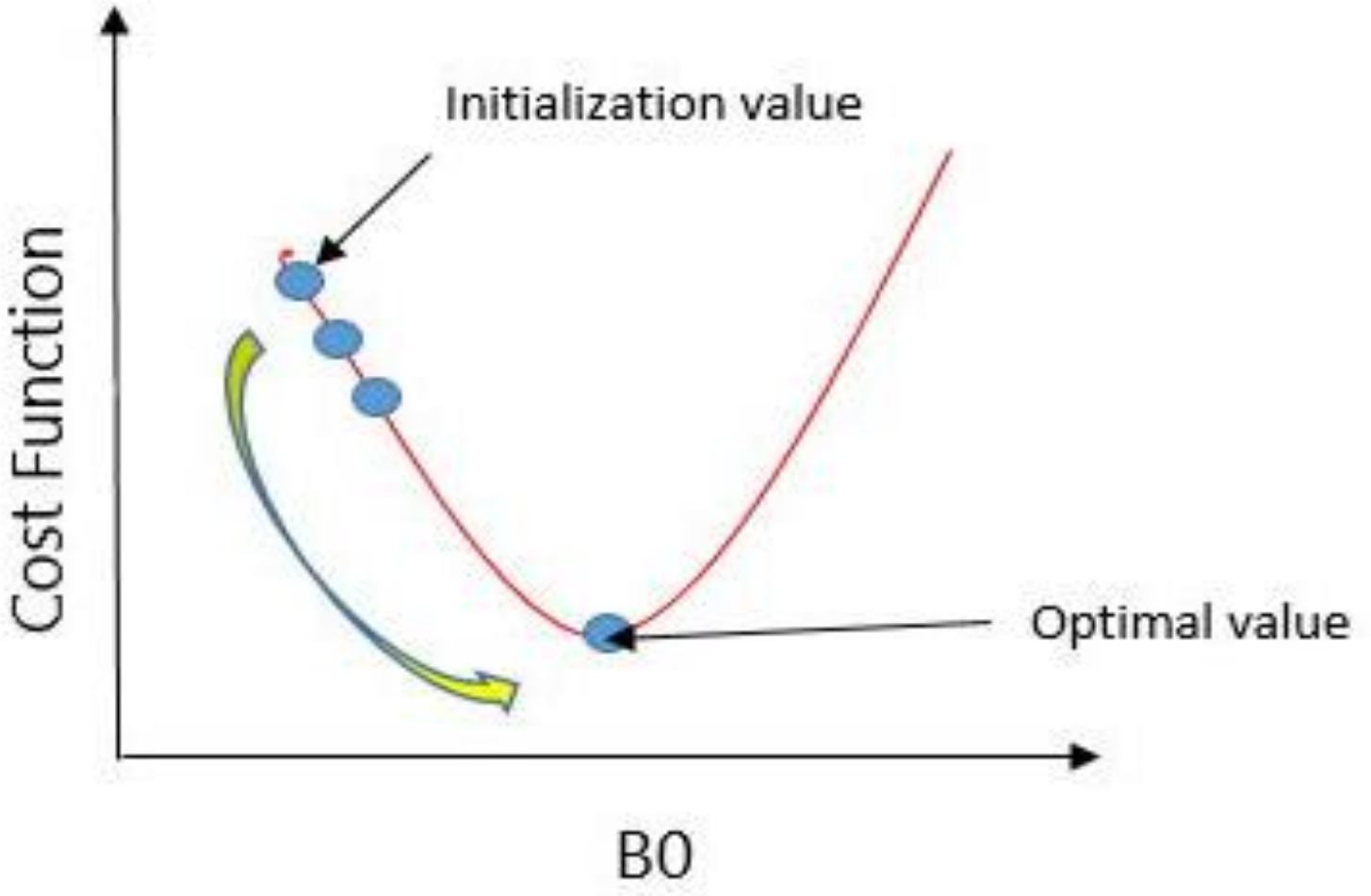
- Utilizing the MSE function, the iterative process of **gradient descent** is applied to update/adjust the values of B_0 and B_1 .This ensures that the MSE value converges to the global minima, signifying the most accurate fit of the linear regression line to the dataset.
- This process involves continuously adjusting the parameters based on the **gradients** calculated from the MSE. The final result is a linear regression line that minimizes the overall squared differences between the predicted and actual values, providing an optimal representation of the underlying relationship in the data.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

Gradient Descent

- Gradient Descent is a generic **optimization algorithm** capable of finding **optimal solutions** to a wide range of problems. The general idea of Gradient Descent is to tweak model parameters iteratively in order to minimize a cost function, to reach the optimal solution.
- A linear regression model can be trained using the optimization algorithm gradient descent by iteratively modifying the model's parameters to reduce the mean squared error (MSE) of the model on a training dataset.

Gradient Descent



❖To update B_0 and B_1, we take gradients from the MSE cost function. To find these gradient, we compute the partial derivatives of the cost function (MSE) with respect to both m and c.

$$B_0 = B_0 - \alpha \cdot \frac{2}{n} \sum_{i=1}^{n} (pred_i - y_i)$$

$$B_1 = B_1 - \alpha \cdot \frac{2}{n} \sum_{i=1}^{n} (pred_i - y_i) \cdot x_i$$

❖These gradients tell us how much we should adjust B_0 and B_1 to reduce the error.

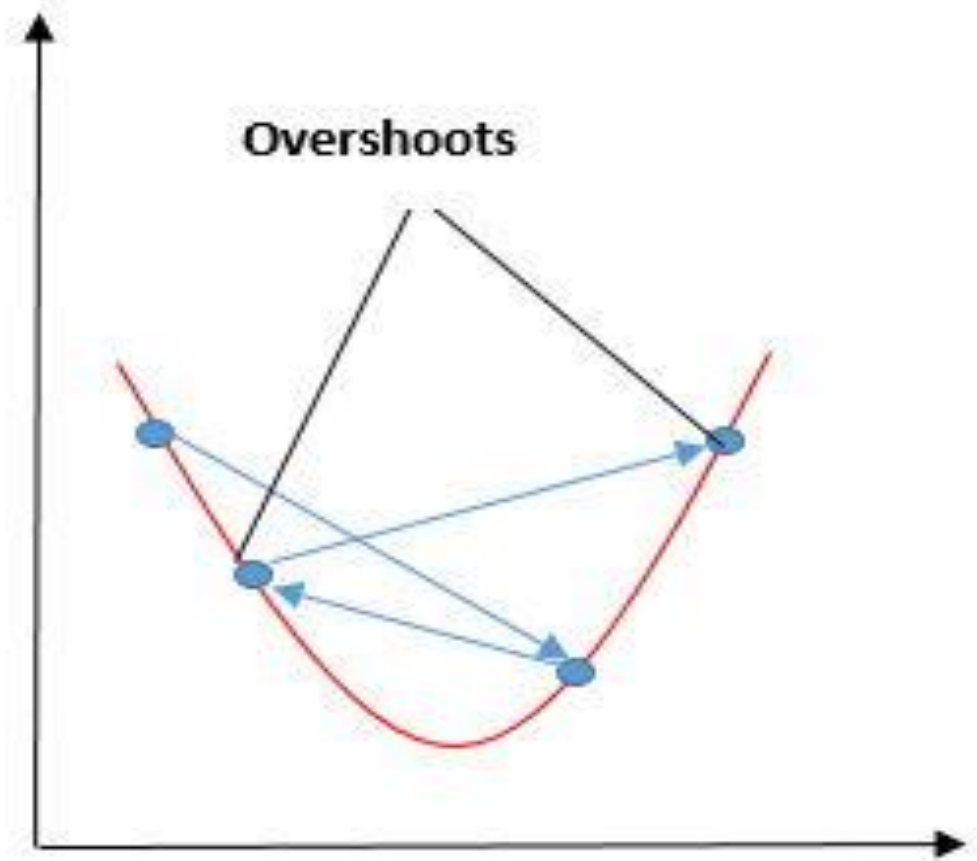
❖Initially, it starts with random B_0 and B_1 values and then iteratively update the values, reaching minimum cost.

❖A **gradient** is nothing but a derivative that defines the effects on outputs of the function with a little bit of variation in inputs.

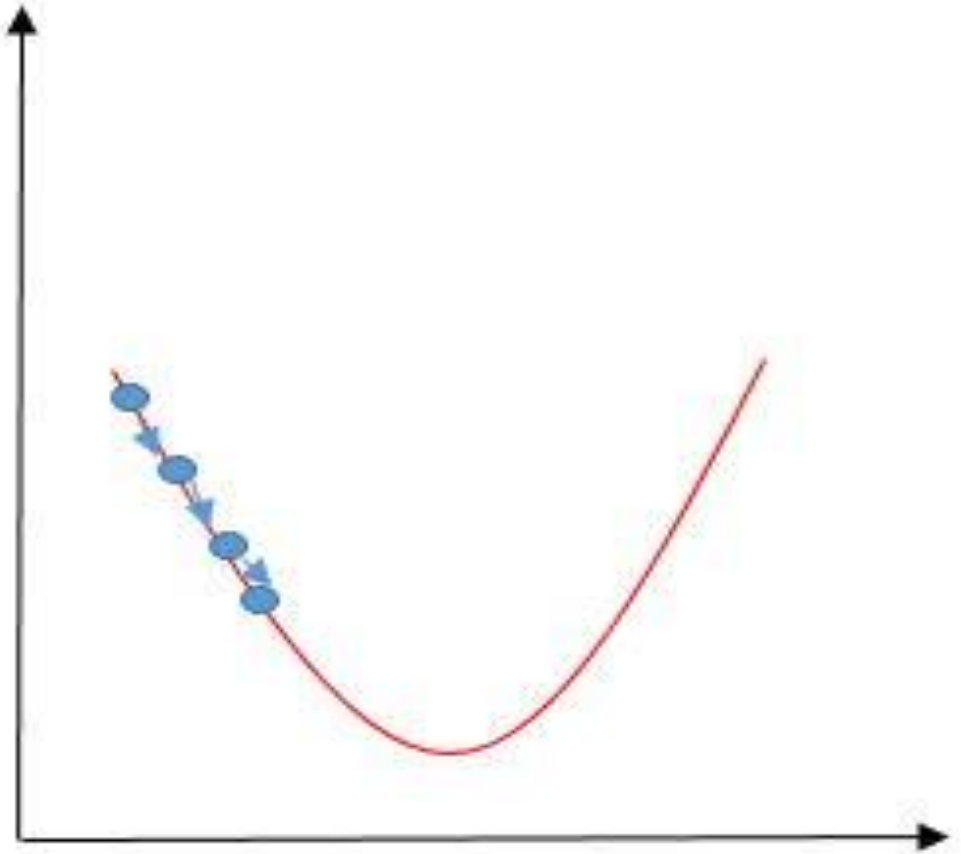
How Linear regression works

1. **Initialize parameters:** Start with initial values for the slope (m) and y-intercept (c).
2. **Calculate predictions:** Use your current parameter values (m and c) to predict the dependent variable ($\hat{y}$) for each data point.
3. **Calculate gradients:** Compute the partial derivatives of the cost function (MSE) with respect to both m and c. These gradients indicate the direction of the steepest descent in the cost function landscape.
4. **Update parameters:** Use the learning rate (α) to determine the step size for the update. Subtract the learning rate multiplied by the gradients from the current values of m and c. This moves the parameters in the direction that minimizes the cost function.
5. **Repeat steps 2-4:** Continue iterating until the cost function converges (stops decreasing significantly) or reaches a certain threshold.

Impact of different values for learning rate



High learning rate



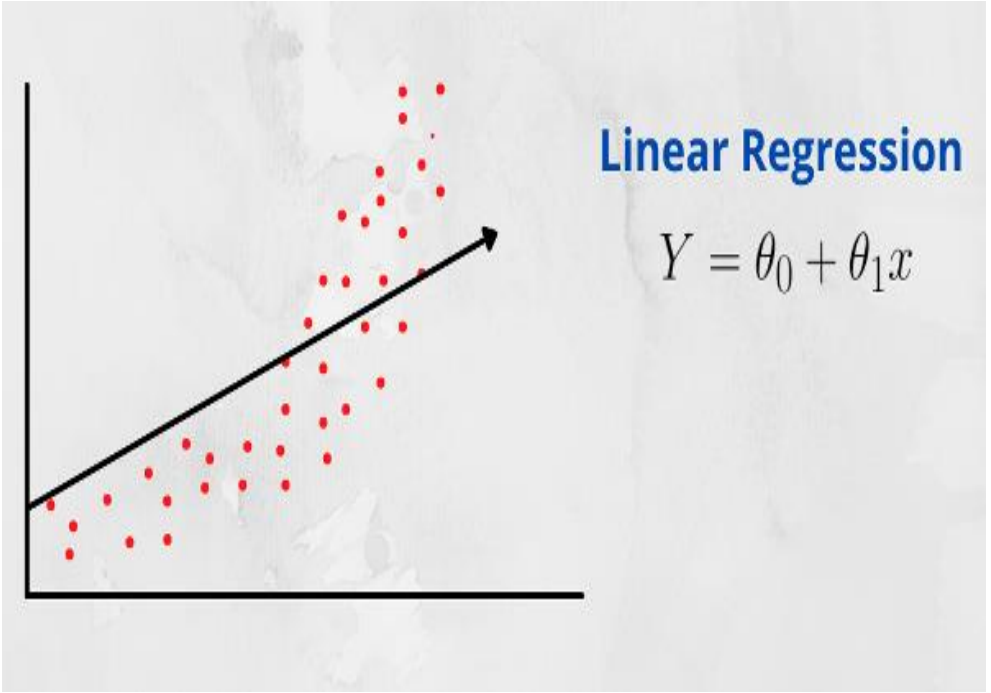
Low learning rate

Polynomial regression

❖**Polynomial regression** helps to build a model over non-linear data.

❖A simple linear regression algorithm only works when the relationship between the data is linear. But suppose we have non-linear data, then linear regression will not be able to draw a best-fit line.

Consider the diagram, which has a non-linear relationship, and you can see the linear regression results on it, which does not perform well, meaning it does not come close to reality.



Polynomial regression

- Polynomial Regression is a regression algorithm that models the relationship between a dependent variable(y) and independent variable(x) as **n^{th} degree polynomial**.
- The Polynomial Regression equation is given below:

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

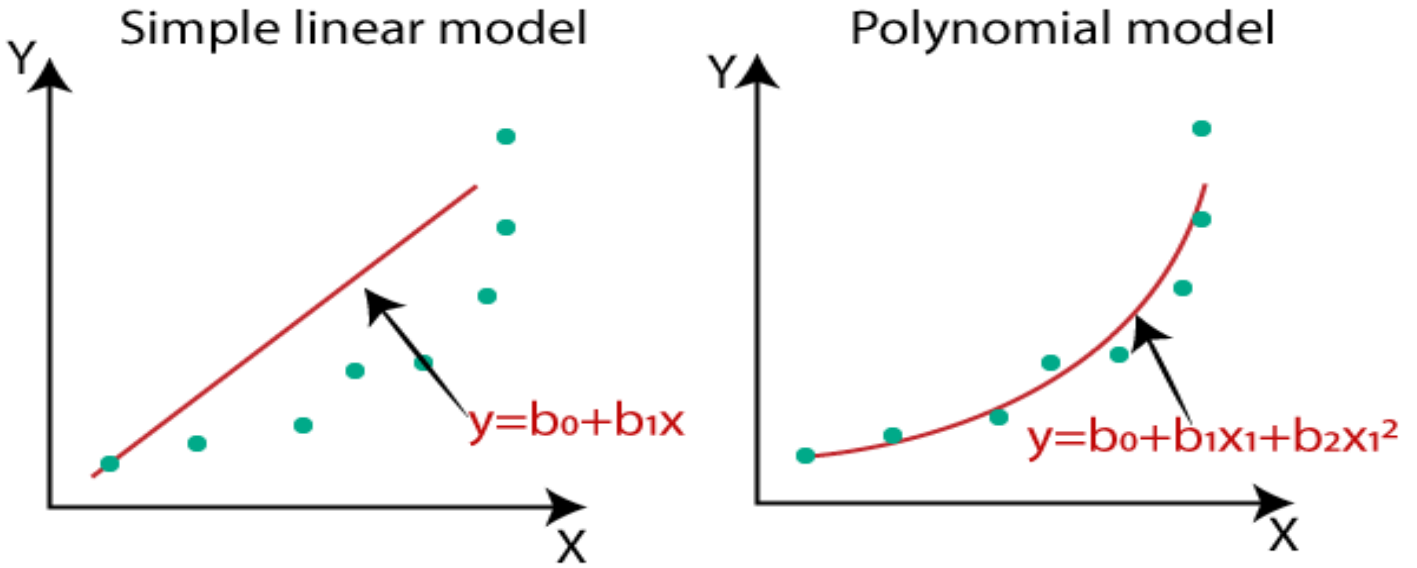
- Polynomial regression is a form of Linear regression where only due to the Non-linear relationship between dependent and independent variables, we add some **polynomial terms** to linear regression to convert it into Polynomial regression.

Types of Polynomial Regression

Polynomials	Form	Degree	Examples
Linear Polynomial	$p(x): ax+b, a \neq 0$	Polynomial with Degree 1	$x + 8$
Quadratic Polynomial	$p(x): ax^2+b+c, a \neq 0$	Polynomial with Degree 2	$3x^2-4x+7$
Cubic Polynomial	$p(x): ax^3+bx^2+cx, a \neq 0$	Polynomial with Degree 3	$2x^3+3x^2+4x+6$

Linear Regression Vs. Polynomial Regression

- We build our model and realize that it performs badly. We examine the difference between the **actual value** and **the best fit line we predicted**, and it appears that the true value has a curve on the graph, but our line is nowhere near cutting the mean of the points. This is where polynomial regression comes into play; it predicts the best-fit line that matches the pattern of the data (curve).



Math Behind Polynomial Regression!

Simple Linear Regression

$$y = b_0 + b_1 x_1$$

Multiple Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

Polynomial Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

Cost Function

- A common cost function employed in polynomial regression is the **mean squared error (MSE)**. It calculates the average of the squared differences between the predicted values and the actual values (y).
- Cost = (1 / n) * sum((y_i - y_predi) ^ 2)
- Where:
  - n represents the total number of training examples.
  - The summation calculates the **squared error** for each data point.
- So, the equation can also be written as:

$$\text{Cost} = 1/n * \text{sum}(\text{square}(y_i - (b_0 + b_1x + b_2x^2 + b_3x^3 .....))$$

Gradient Descent for Polynomial Regression

- **Gradient descent** is an optimization algorithm used to find the values of parameters (coefficients) of a function that minimizes a cost function.

$$m = m - \alpha * derivative\ of\ m$$

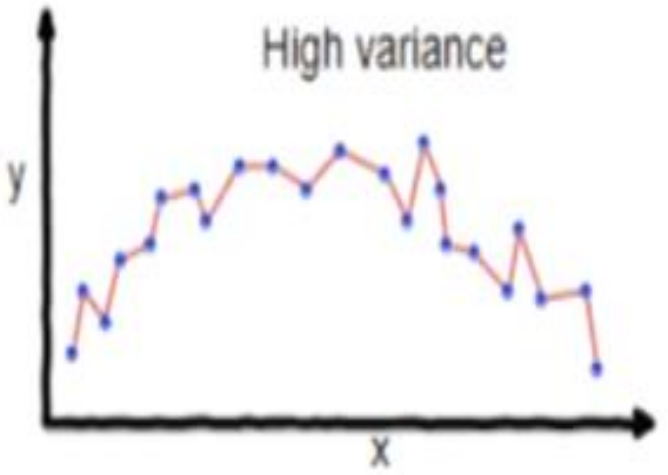
$$b = b - \alpha * derivative\ of\ b$$

The values of slope (m) and slope-intercept (b) will be set to 0 at the start of the function, and the learning rate (α) will be introduced. The learning rate (α) is set to an extremely low number, perhaps between 0.01 and 0.0001. The learning rate is a **tuning parameter** in an optimization algorithm that sets the step size at each iteration as it moves toward the cost function's minimum.

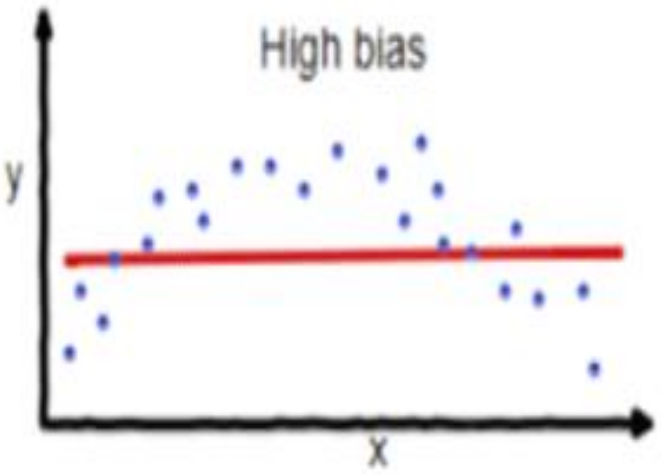
Overfitting Vs Under-fitting

- In linear regression, when analyzing a dataset linearly, we encounter an under-fitting problem.
  - Model is unable to capture data patterns due to the model's simple assumptions in fitting the data.
- In polynomial regression one thing that we face is the problem of overfitting this happens because while we increase the order/degree of the polynomial regression to achieve better and better performance, model gets overfit on the data and does not perform on the new data points.
  - Model memorizes the training data too well and loses generalizability.
- Due to this reason only while using the polynomial regression, do we try to penalize the weights of the model to regularize the effect of the overfitting problem.

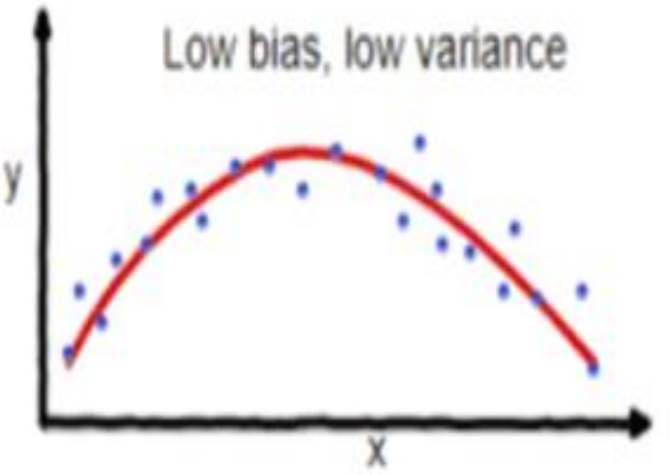
Overfitting Vs Under-fitting



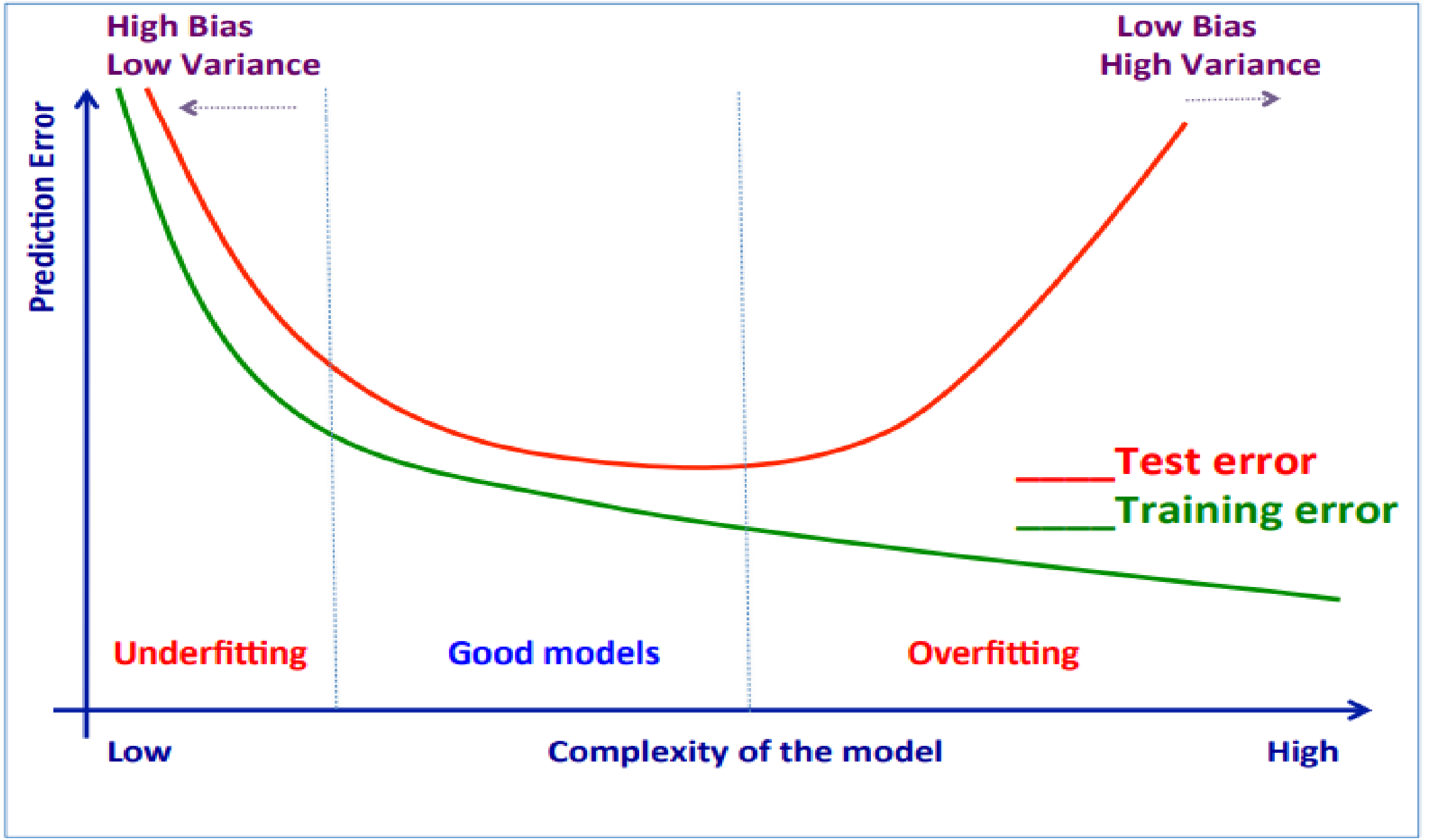
overfitting



underfitting



Good balance


High Bias
Low Variance
Low Bias
High Variance
Prediction Error
Test error
Training error
Underfitting
Good models
Overfitting
Low
Complexity of the model
High

Regularization Technique

- While developing machine learning models you must have encountered a situation in which the training accuracy of the model is high but the validation accuracy is low. This is the case which is popularly known as **overfitting.**
- **Regularization in ML which helps us to solve the problem of overfitting.**
- Regularization is implemented by adding a “penalty” term to the best fit derived from the trained data, in order to achieve a *lesser variance* with the tested data and also restricts the influence of feature variables over the output variable by compressing their coefficients.

Regularization in Machine Learning

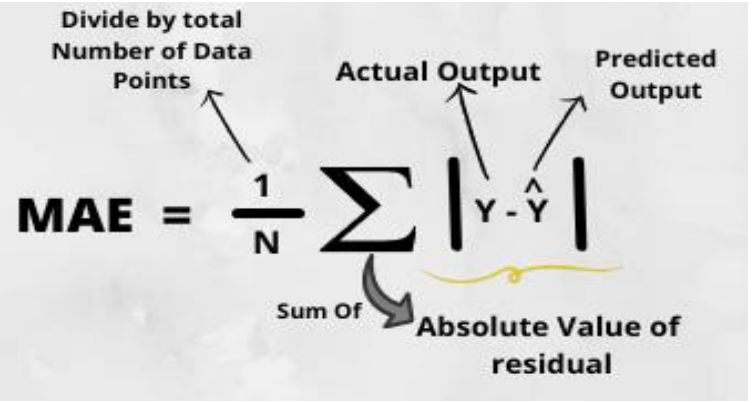
- Regularization is a technique used to prevent overfitting by adding a penalty term to the loss function, discouraging the model from assigning too much importance to individual features or coefficients.

- The commonly used regularization techniques are:
  - Lasso Regularization – L1 Regularization
  - Ridge Regularization – L2 Regularization

Evaluation metrics for regression models

- To build and deploy a generalized model we need to evaluate the model on different metrics which helps us to better optimize the performance, fine-tune it, and obtain a better result.
- Some of the Evaluation metrics used for Regression analysis are:

1. Mean Absolute Error (MAE)



- MAE is simplest and easily comprehensible loss functions, which calculates the absolute difference between actual and predicted values. We aim to get a minimum MAE because this is a loss.

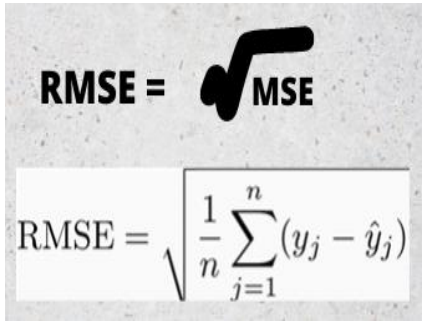
Evaluation metrics for regression models

2. Mean Squared Error (MSE) :

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and}}$$

MSE is a most used and very simple metric
MSE represents the squared distance between actual and predicted values.

3. Root Mean Squared Error (RMSE) :



. As RMSE is clear by the name itself, that it is a simple square root of mean squared error.

4. R-squared (R^2) Score:

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

R2 score is a metric that tells the performance/accuracy of your model. **R^2** Score usually ranges from 0 to 1. **SST** is the total sum of squares and **SSR** is the total sum of squares of residuals.

Classification

- ✓Given: $D^{train=} (x_1, y_1), \ldots, (x_n, y_n) / x_i \in R^d$ and y_i is the label

- **Goal**:
  - ✓Learn a function to predict *y* given *x*
    - *y* is categorical/discrete == classification
  - ✓In other word, Build a model that can accurately predict the class labels of new, unseen instances.

Types of classification tasks

- Types of classification tasks based on the nature of the class labels.
  - **Binary Classification:** The simplest case, where there are only two classes (e.g., classifying an email as spam/not spam(ham)).
  - **Multi-Class Classification:** More than two classes (e.g., classifying images as cat, dog, or horse. Predicting handwritten digits[0-9] where there are 10 classes(response variable can have any value ranging from 0 to 9).
  - **Multi-Label Classification:** A data point can belong to multiple classes simultaneously. For example, an image can contain a cat, a dog, and a person. predicting the presence of multiple diseases in a patient based on medical records.

Classification problem 1: Predicting the Weather

- Suppose we take a real-world example of predicting the weather.
- Let’s keep it simple and say we are trying to predict if the weather is **sunny** or **rainy** based on multiple input data samples consisting of attributes or features like humidity, temperature, pressure, and precipitation.
  - **Input features** = { humidity, temperature, pressure, and precipitation}
  - **Output labels** = {Sunny, runny}
- Since the prediction can be either sunny or rainy, there are a total of two distinct classes in total; hence this problem can also be termed as a **binary classification problem.**

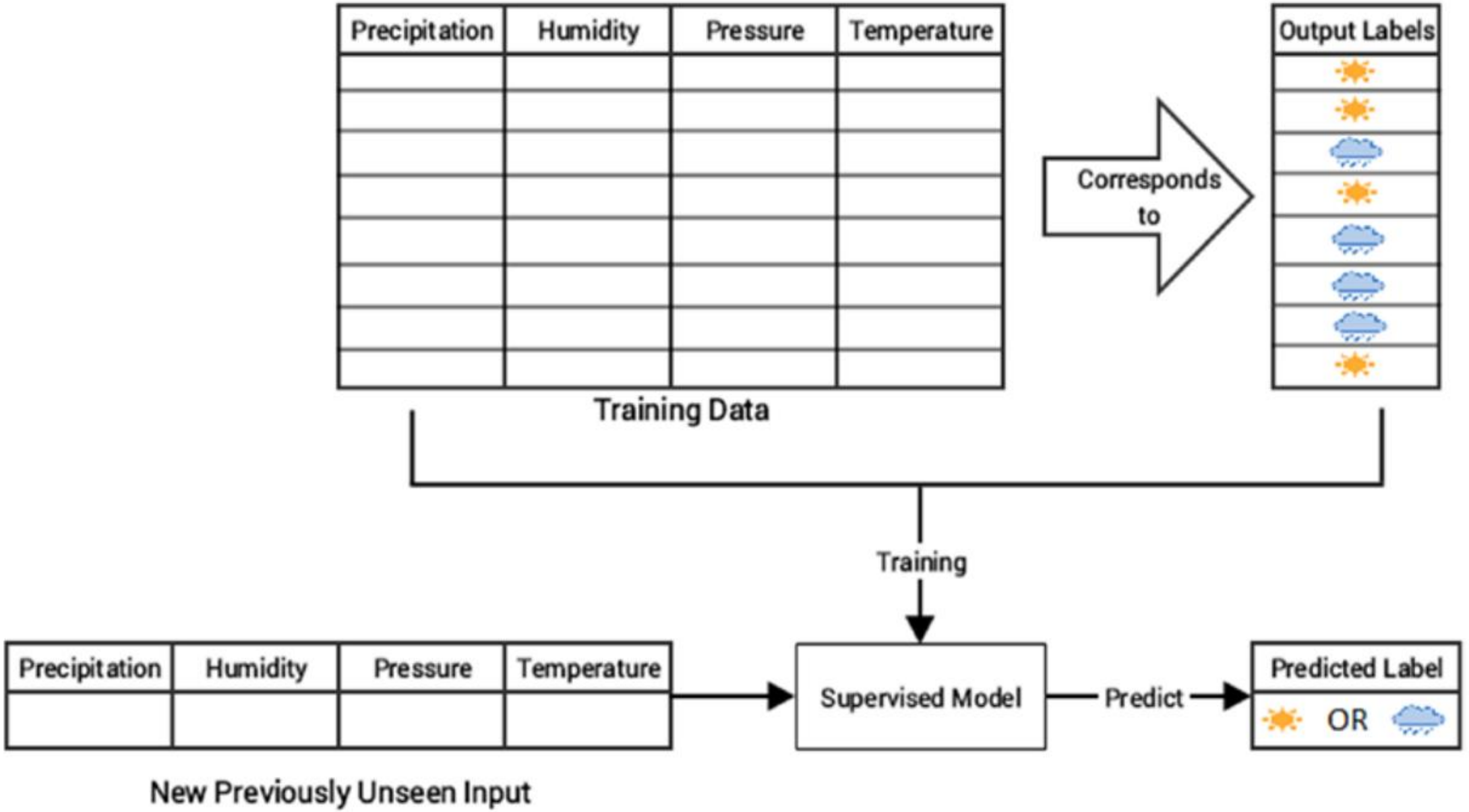


Figure. Supervised learning: binary classification for weather prediction

Classification problem 2: Iris plant classification

Example: classifying iris plants (Anderson 1935).

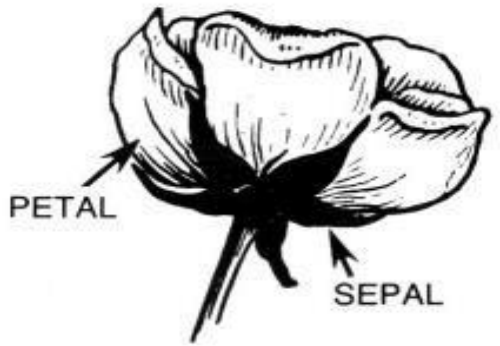
iris setosa

iris versicolor

150 iris plants (50 of each species):

- 3 species:
  setosa, versicolor, virginica
- length and width of sepals (in cm)
- length and width of petals (in cm)

iris virginica



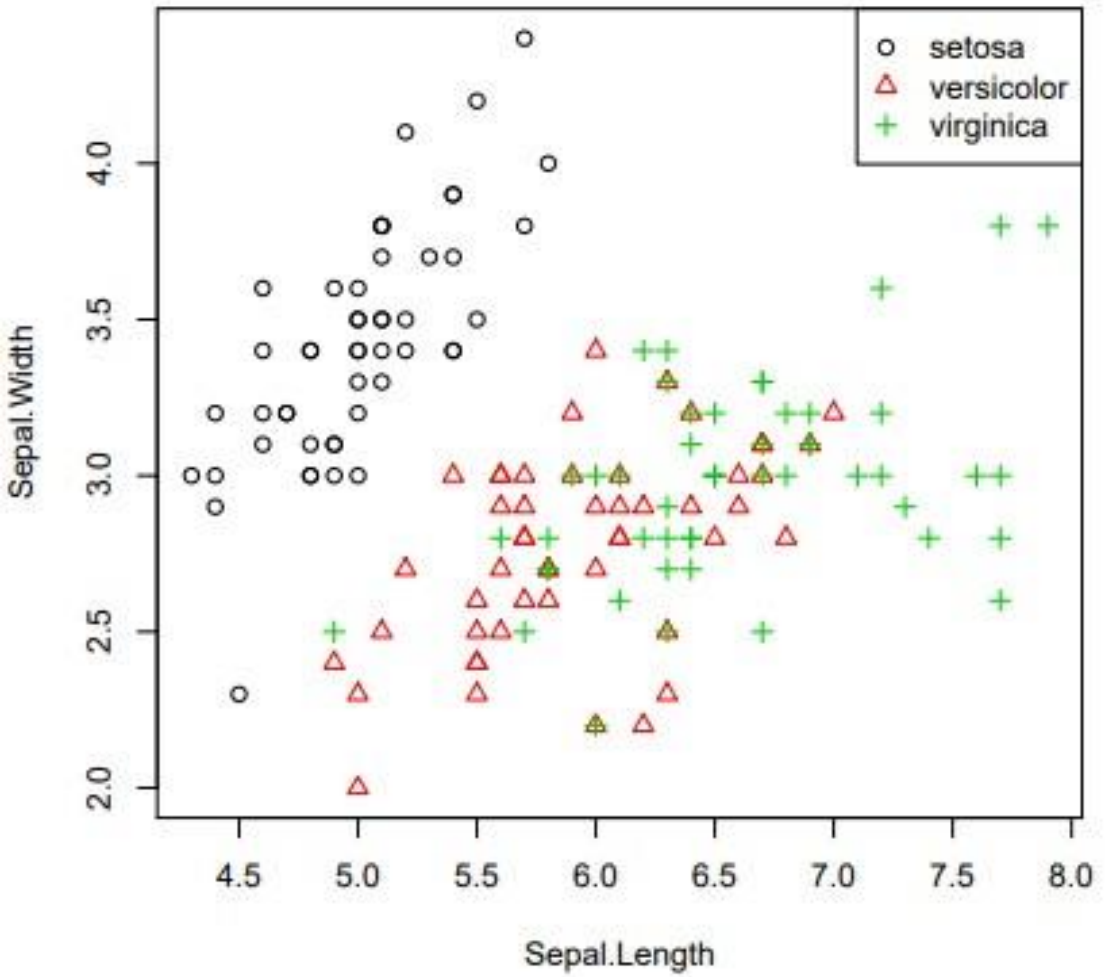
Given the lengths and widths of sepals and petals of an instance, which iris species does it belong to?

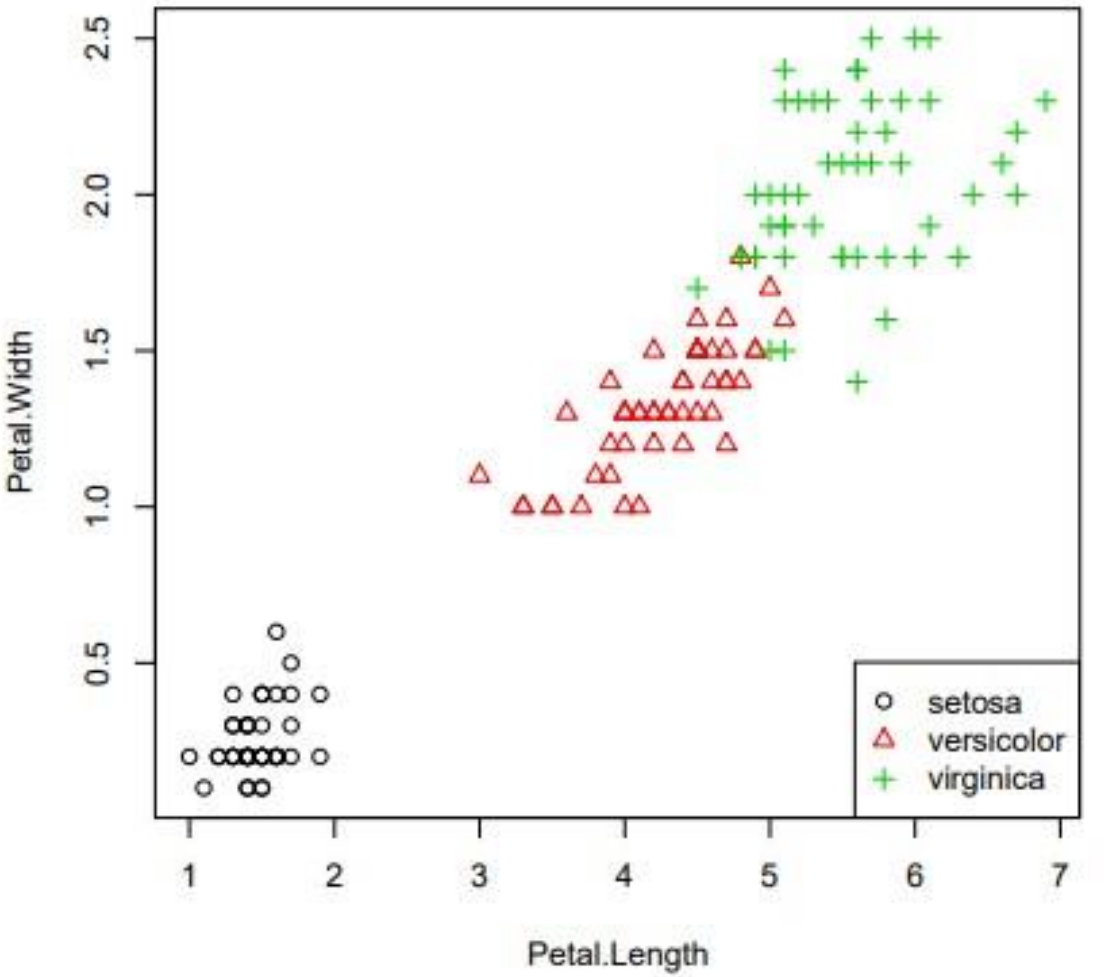
There are three distinct classes in total; hence this problem can also be termed as a **multi-class classification.**

Cont.... the data:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.10	3.50	1.40	0.20	setosa
2	4.90	3.00	1.40	0.20	setosa
3	4.70	3.20	1.30	0.20	setosa
⋮	⋮	⋮	⋮	⋮	
51	7.00	3.20	4.70	1.40	versicolor
52	6.40	3.20	4.50	1.50	versicolor
53	6.90	3.10	4.90	1.50	versicolor
⋮	⋮	⋮	⋮	⋮	
101	6.30	3.30	6.00	2.50	virginica
⋮	⋮	⋮	⋮	⋮	
150	5.90	3.00	5.10	1.80	virginica

Cont....


setosa
versicolor
virginica
Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


setosa
versicolor
virginica
Petal.Width
Petal.Length
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7

Classification has a vast array of real-world applications

1) **Fraud Detection**: Classifying financial transactions as fraudulent or legitimate based on patterns, anomalies, or historical data.
2) **Sentiment analysis:** Categorizing customer product reviews as positive, negative, or neutral.
3) **Image recognition:** Classifying images into various categories like cat, dog, horse etc.
4) **Object recognition:** Identifying objects in an image, like a picture containing a cat, a person, and a tree.
5) **News article/document classification:** Categorizing a news article by topic, such as politics, sports, and technology (based on their content).
6) **Cybersecurity threat detection:** Identifying malicious network activity from a vast amount of regular network traffic.
7) **Credit Risk Assessment**: Classifying loan applicants as low risk or high risk based on their credit history, income, and other relevant factors.
8) **Customer Churn Prediction**: Classifying customers as likely to churn(cancel service) or not churn from a subscription or service based on their behavior, usage patterns, or demographics.

Typical classification models

Typical classification models include the following major types of methods; however, the list is not exhaustive.

- Linear models like logistic regression, Naïve Bayes, and support vector machines
- Non-parametric models like K-nearest neighbors(KNN)
- Tree based methods like decision trees
- Ensemble methods like random forests (bagging) and gradient boosted machines (boosting)
- Neural networks (MLPs)

Decision Tree

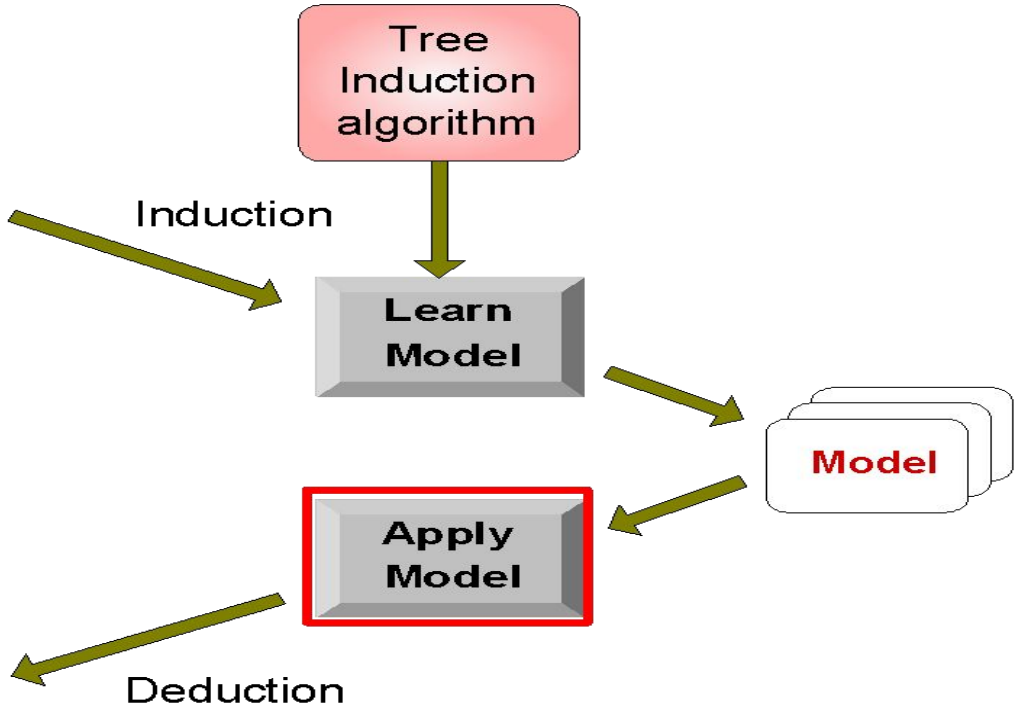
- Decision Trees are versatile supervised machine Learning algorithms that can perform both classification and regression tasks. They are powerful algorithms, capable of fitting complex datasets.

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

Training Set

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

Test Set



Sample Dataset

- Columns denote **features** X_i,
- Rows denote labeled **instances** (x_i, y_i)
- Class label denotes whether customer buys a computer.

Training Data Set: ***buys_computer***

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

(x_i, y_i)

Attributes/feature

Class/label

Decision Tree Induction

- **Decision tree induction** is the learning of decision trees from class-labeled training tuples.
- To build optimal decision tree select an attribute that produces purest subset of records.

select **best attribute**

that splits this data set.

Training Data Set: ***buys_computer***

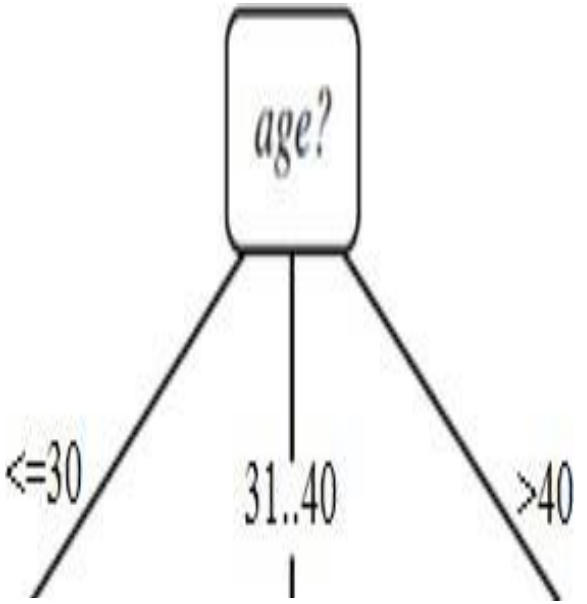
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent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

Attributes/feature

Class/label

Examples of Decision Tree Induction:

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



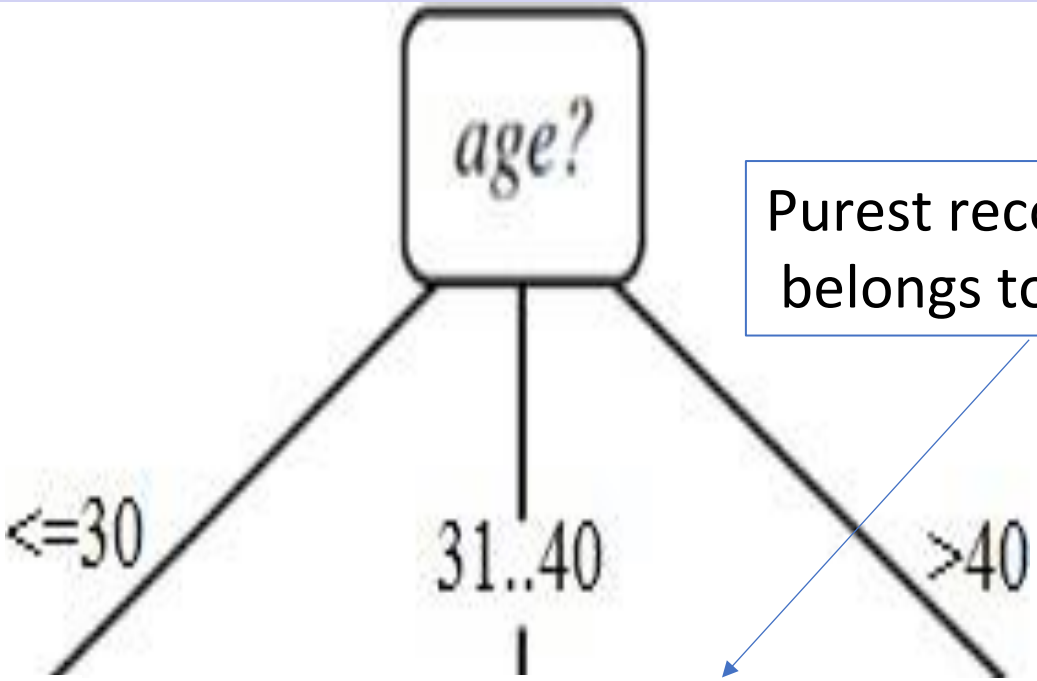
age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

age	income	student	cr	buys
31...40	high	no	fair	yes
31...40	low	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

- A decision tree is grown by recursively **partitioning** the **training records** successively into **purer subsets.**

Cont'd



Purest record because all instances belongs to the same class

age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

age	income	student	cr	buys
31…40	high	no	fair	yes
31…40	low	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes

yes

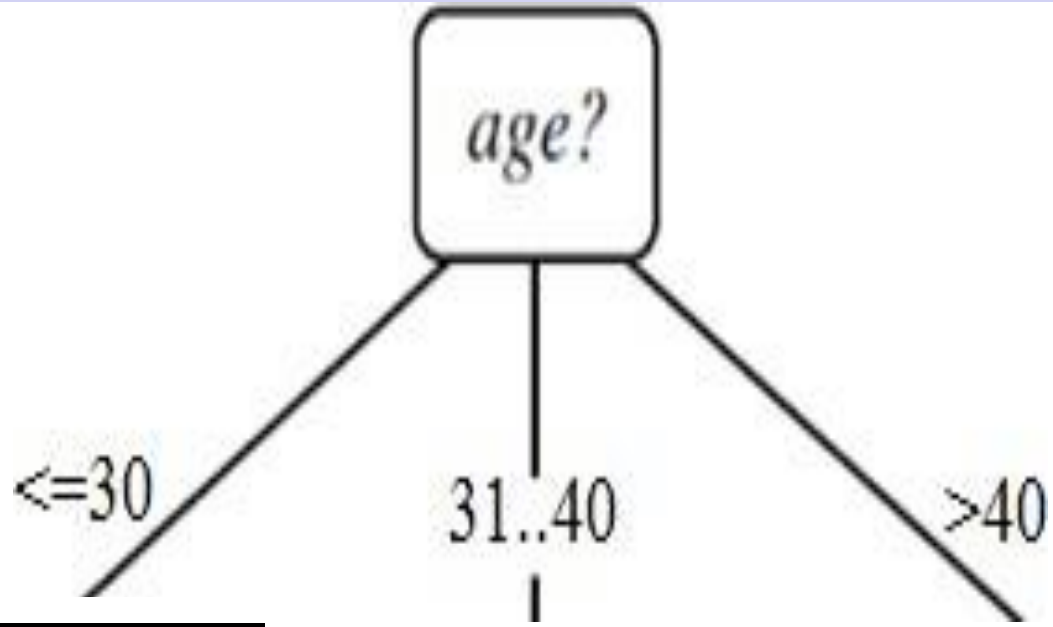
all examples are yes

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

Cont'd

select **best attribute**

that splits this data set.



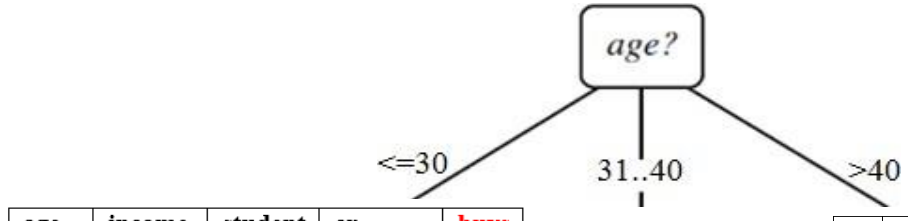
age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

Yes

all examples are yes,
No further split is required

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

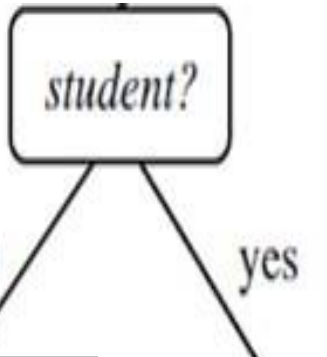
Decision Tree Induction Algorithm: Example



age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

yes

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no



age	income	student	cr	buys
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no

no

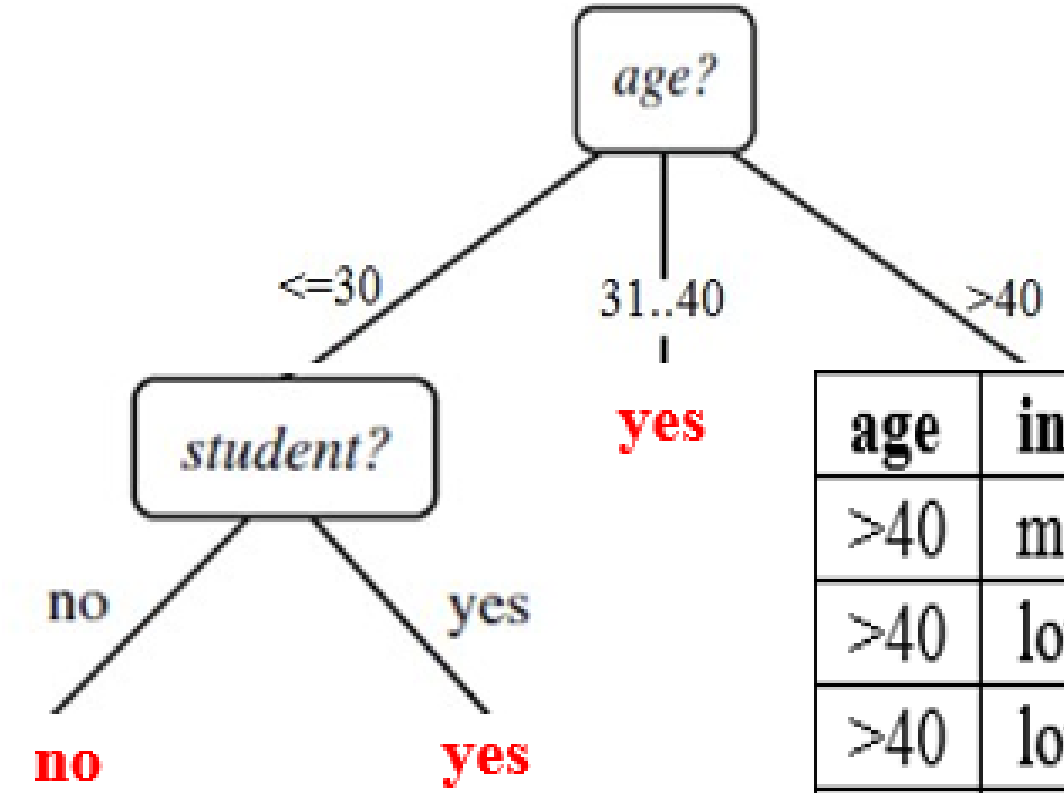
all examples are no

age	income	student	cr	buys
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

yes

all examples are yes

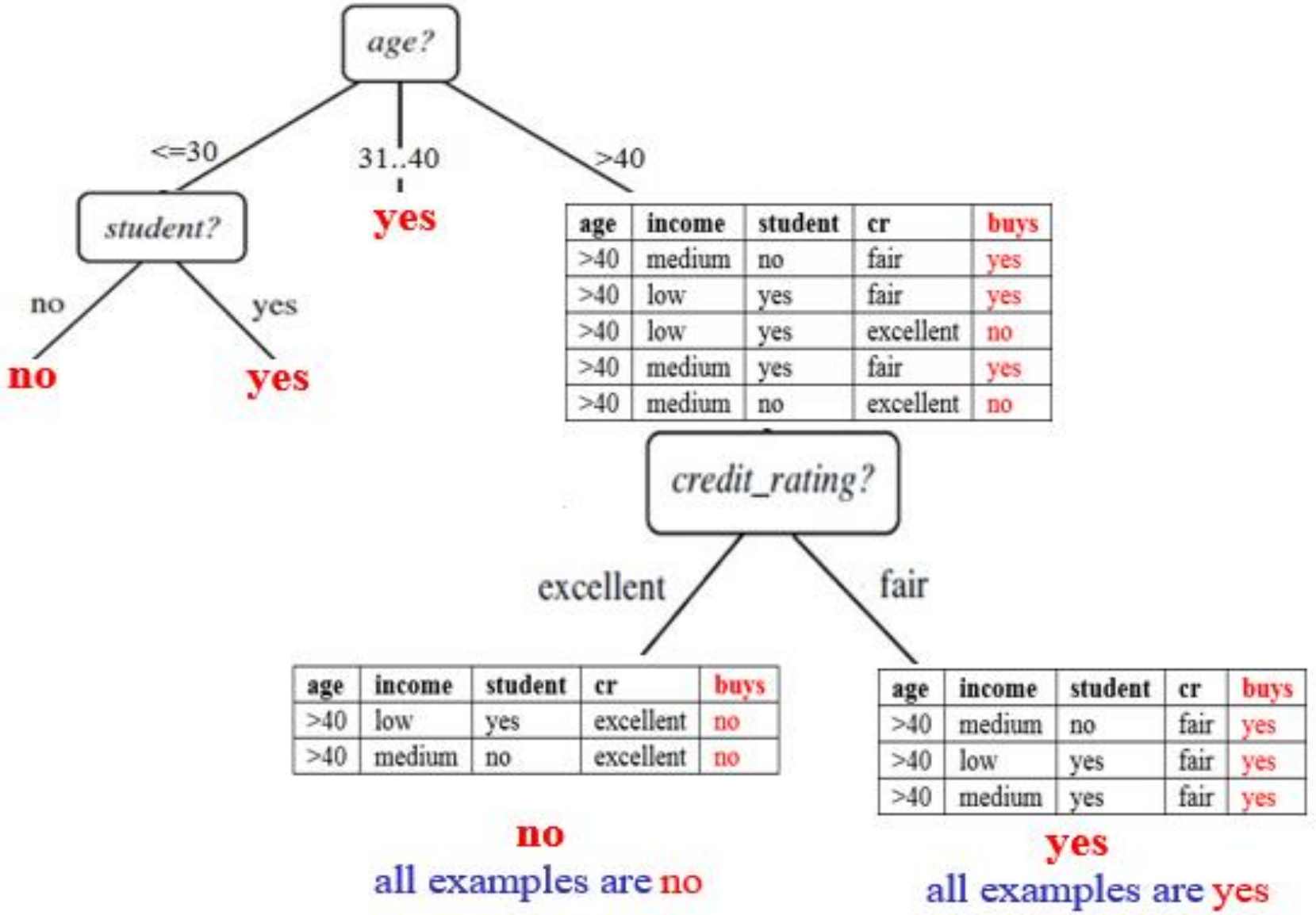
Decision Tree Induction Algorithm: Example



age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

select **best attribute** that splits this data set.

Decision Tree Induction Algorithm: Example



age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

age	income	student	cr	buys
>40	low	yes	excellent	no
>40	medium	no	excellent	no

age	income	student	cr	buys
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	medium	yes	fair	yes

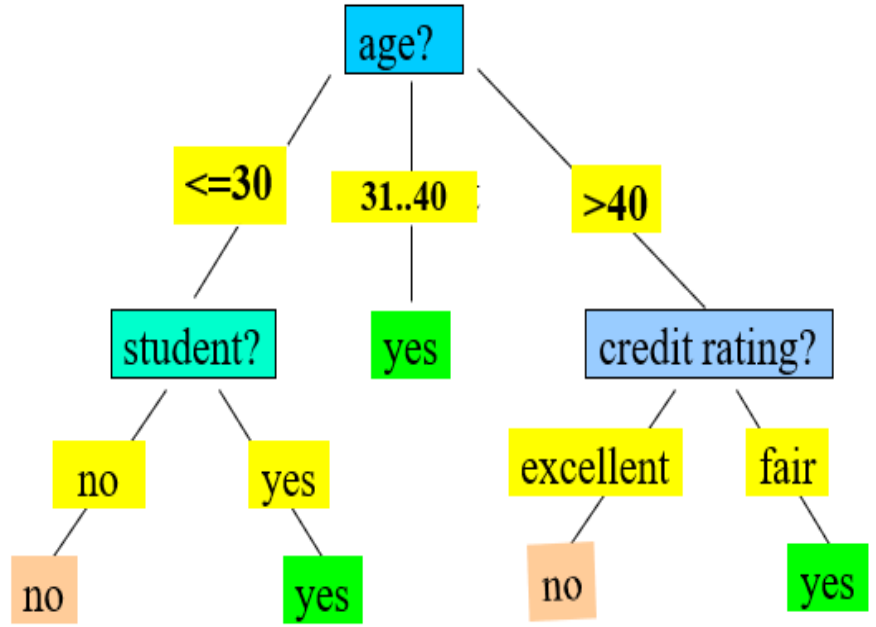
Training and Testing Decision Tree Model

Training Data

Training Data Set: ***buys_computer***

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

Model: Decision Tree



Test Data

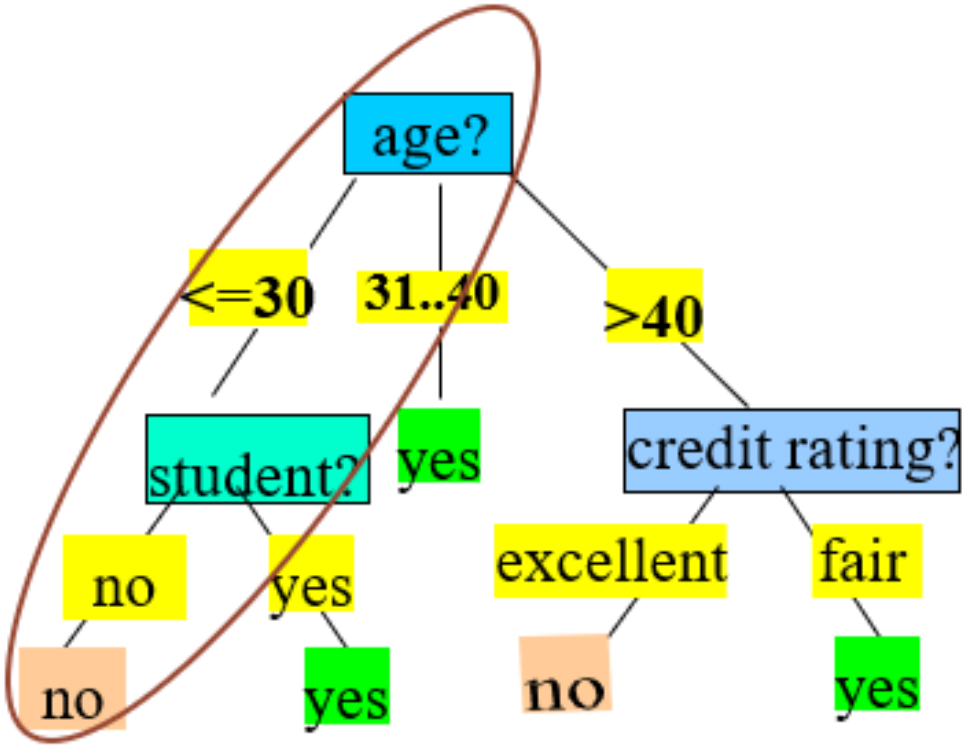
Age	Income	Student	Credit_rating	Buys_computer
<=30	medium	No	fair	?
>40	high	yes	excellent	?

Apply Model

Tree-based Models is Easy to Interpret

- A path from root to a leaf node corresponds to a rule
- E.g., **if** age<=30 and student=no **then** target value, buys=no

Set of rules can be visualized as a tree.



✓ Each path from the tree root to a leaf corresponds to a conjunction of attribute tests.

✓ The instance (age <=30, income = low, student = no, credit_rating = fair) is classified as a **negative** instance.

Decision Tree Induction Algorithm: Example 2

Features **Class/Label**

. outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

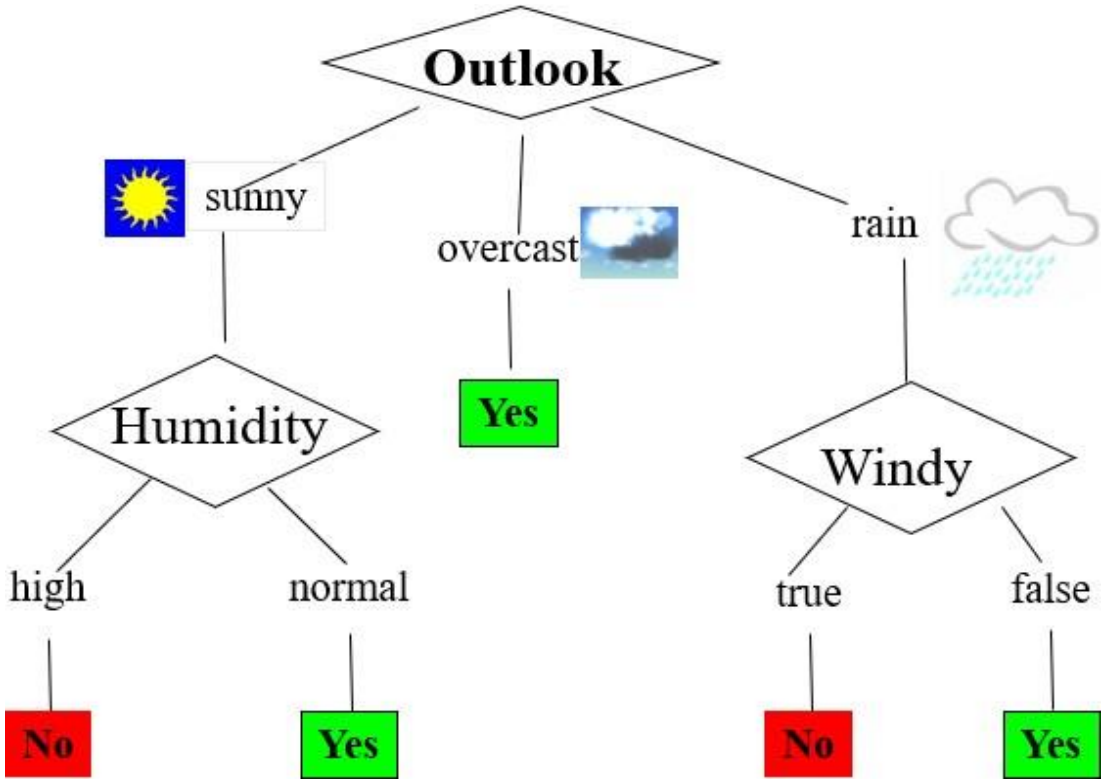
Build a decision tree model for the given dataset

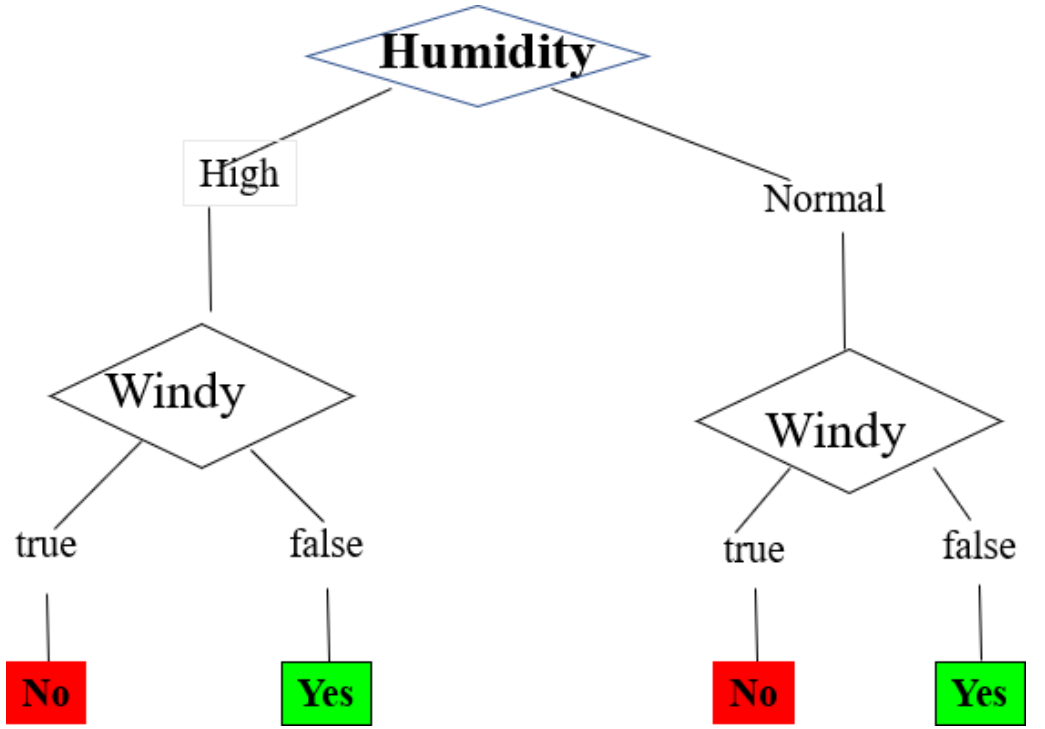
Given : <outlook=sunny, temperature=hot, humidity=high, wind=weak> ?

Predict, if there will be a match?

Many possible Decision Trees

There could be more than one tree that fits the samedata!





Which Decision Tree is the Best?

Decision Tree Induction Algorithms

❖ How to learn a decision tree from training data?

❖ Which feature should be used to break the dataset?

- ✓ finding an optimal decision tree is NP-hard
- ✓ tree building algorithms thus use a greedy, top-down, recursive partitioning strategy to induce a reasonable solution

Number of Algorithms:

- Hunt's: Hunt's Algorithm (1966)
- Quinlan's: Iterative Dichotomizer 3 (ID3) (1975) uses Entropy
  - C4.5 / 4.8 / 5.0 (1993) (Successor of ID3) uses Entropy
- Brieman's: CART: Classification and Regression Trees (1984) uses Gini

Design Issues for Decision Tree induction

1. How should training records be split?
   - ❖ How to specify the attribute test condition (Splitting Criterion)?
     - ◆ Depending on attribute types(Nominal, Ordinal, Continuous)
   - ❖ How to determine the best split?
     - Measure for evaluating the goodness of a **test condition**
     - Different purity measures can be used:
       - information gain, , Gini index, gain ratio misclassification error, …
2. When should the splitting procedure stop?
   - –Stop splitting if all the records belong to the same class.
   - Early termination depending on the results of a statistical test, b/c fully grown trees might overfit training data

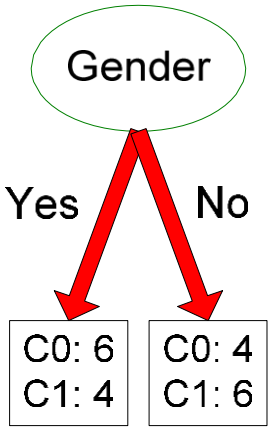
How to Determine the Best Split?

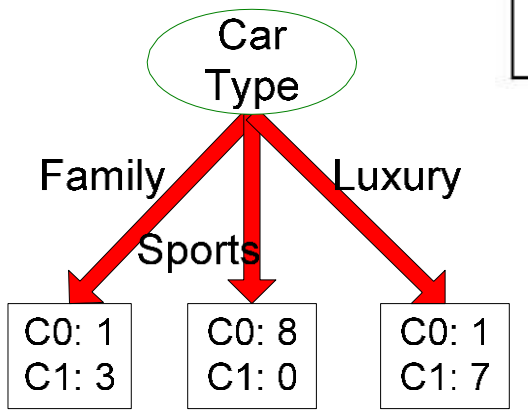
Before splitting the dataset contains:

- 10 records of class C0 and
- 10 records of class C1

Which attribute test is the best?

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1





How to determine the Best Split?

- **Key problem**: choosing which attribute to split a given set of training records.
- Greedy approach:
  - Nodes with purer/homogeneous class distribution are preferred
- Need a measure of node impurity:

C0: 5 C1: 5

Non-homogeneous

High degree of node impurity

C0: 9 C1: 1

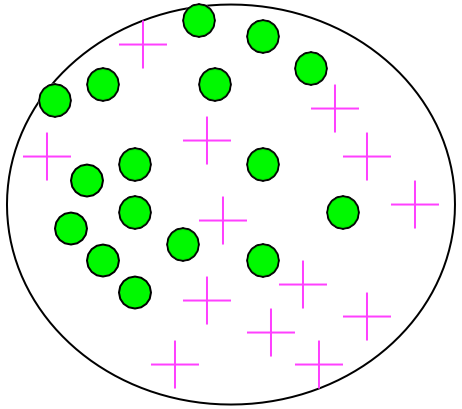
Homogeneous

Low degree of node impurity

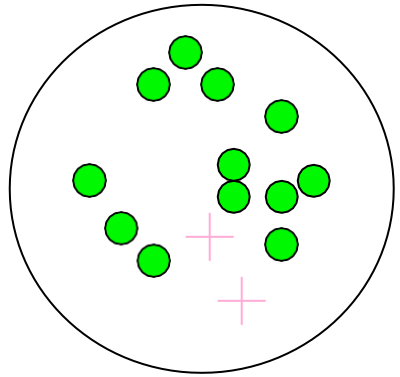
How to determine the Best Split?

Node Impurity

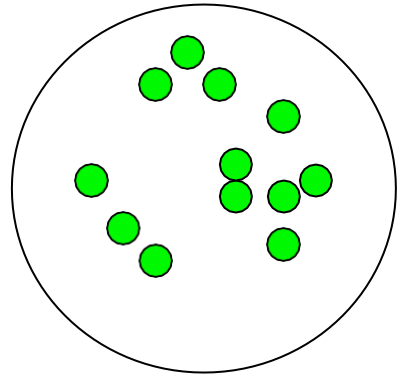
Very impure group



Less impure



Minimum impurity



❖ **Common measures of node impurity**

1. GINI Index
2. *Information gain**:-follows max-gain***

- The ID3 algorithm uses the Max-Gain method of selecting the best attribute

Splitting Based on Information Gain

Weighted impurity measure of children

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node p is split into k partitions; n_i is number of records in partition i

- Information gain measures the entropy reduction of a split
- We choose the split with the largest reduction (maximal GAIN)
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure (split by ID attribute?)

How to determine the Best Split?

Splitting Based on Information gain

- Information gain relies on the entropy of each node
- Entropy of a given node t:

$$Entropy\,(t) = -\sum_{j} p(j \mid t) \log_2 p(j \mid t)$$

$p(j \mid t)$ is the relative frequency of class j at node t

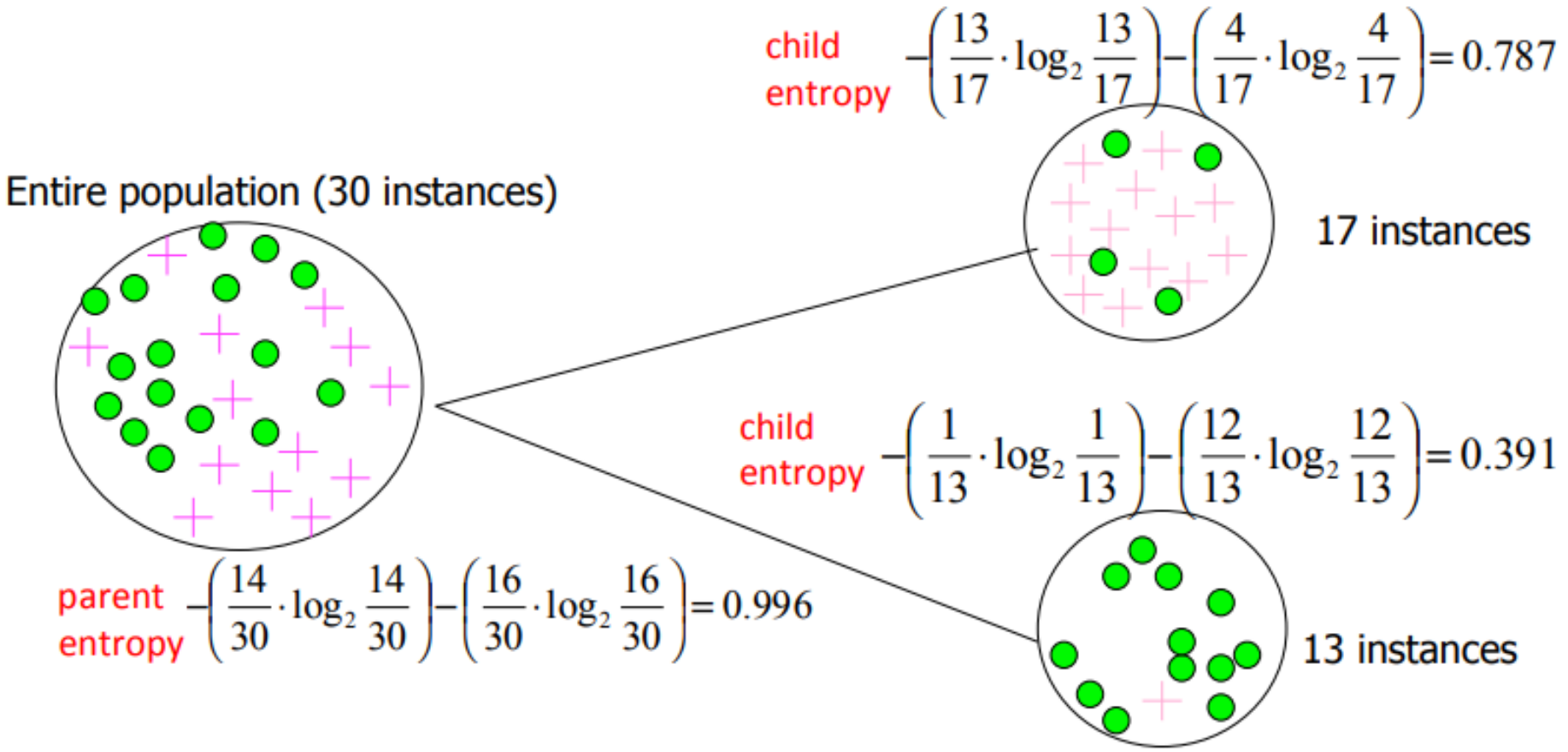
- Entropy measures homogeneity of a node(impurity of a node)

Splitting Based on Information Gain

1. Calculate the entropy of the total dataset
2. Choose an attribute and Split the dataset by an attribute
3. Calculate the entropy of each branch
4. Calculate Information Gain of the split
5. Repeat 2, 3, 4 for all Attributes
6. The attribute that yields the largest IG is chosen for the decision node.
7. Repeat 1 to 6 for all sub-databases till we get sub-databases with single class

Calculating Information Gain

Information Gain = entropy(parent) – [average entropy(children)]



(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

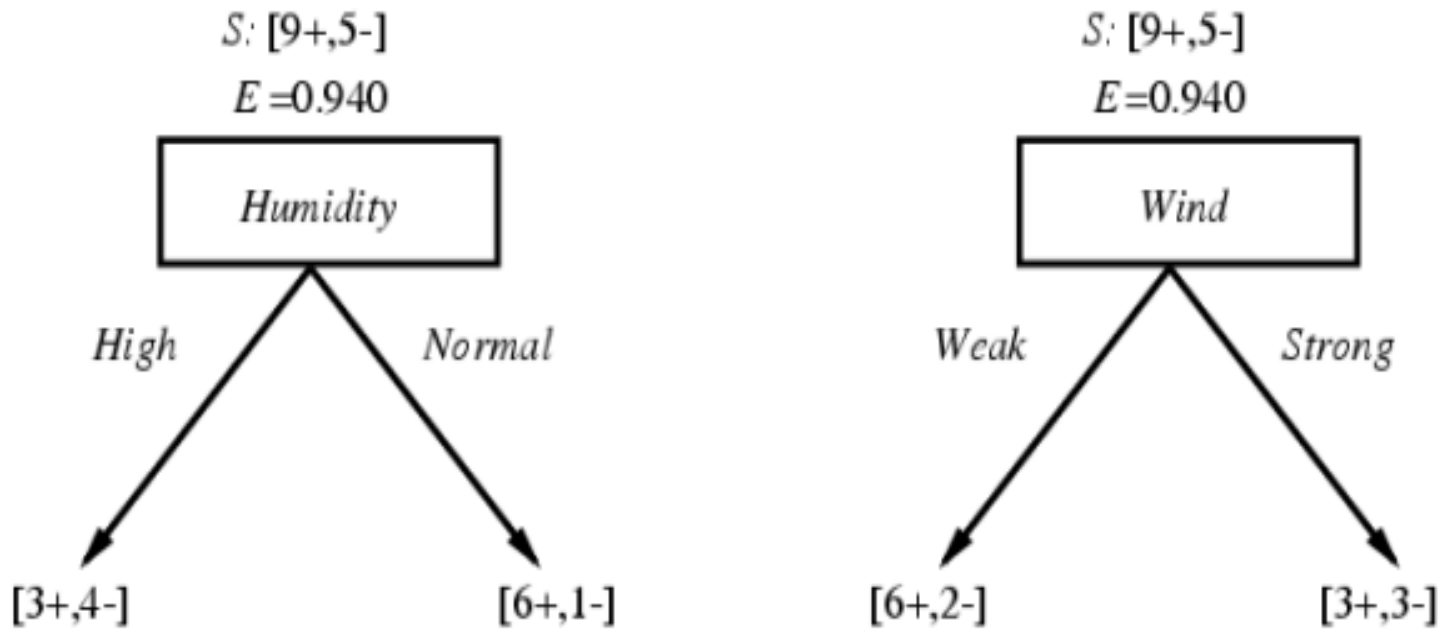
Information Gain= 0.996 - 0.615 = 0.38

Training Examples

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

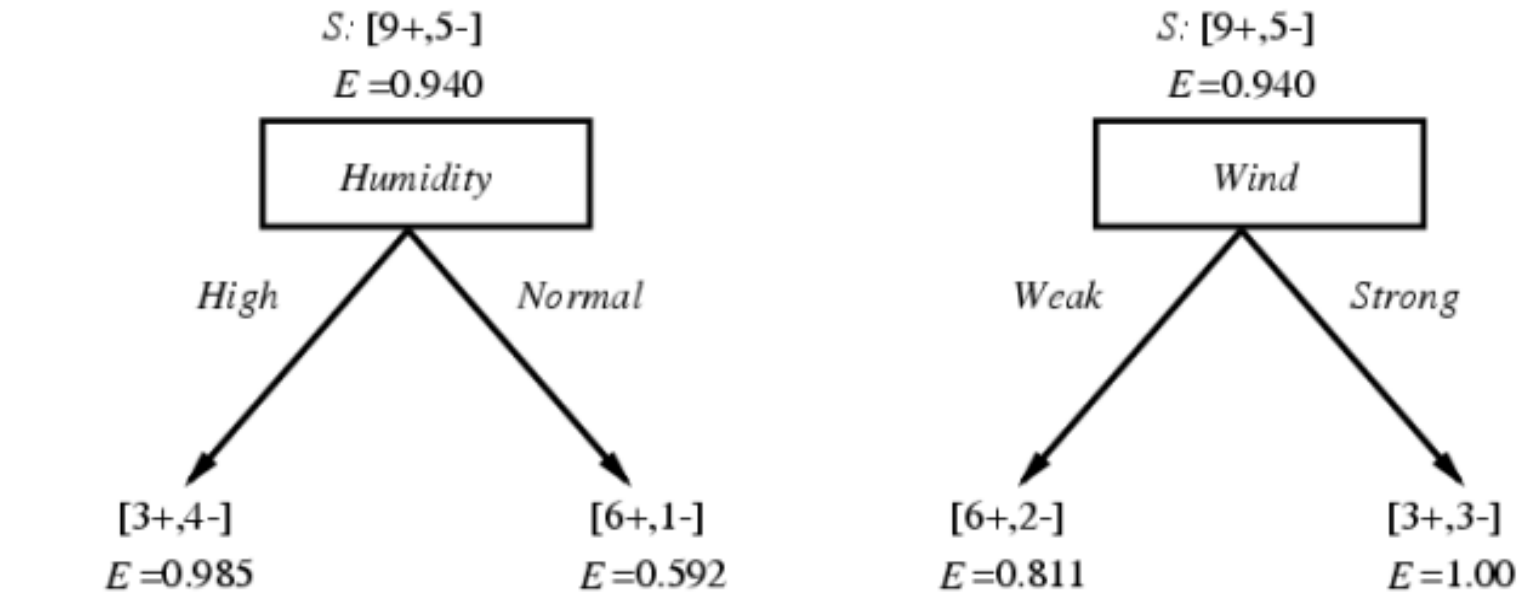
Selecting the Next Attribute

Which attribute is the best classifier?



Selecting the Next Attribute

Which attribute is the best classifier?



Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

Parametric and non-parametric methods

These methods **approach learning** from data in fundamentally different ways:

Parametric methods:

- ❖Parametric methods make assumptions about the underlying statistical distribution of your data and functional forms(eg. Y=mx+b).
- ❖Parametric method assumes/follows a specific **predefined form for model**, and summarizes the given data with a fixed number of parameters (independent of the number of training examples).
- ❖Imagine a straight line for linear regression - that's the form, and the parameters are the slope and intercept.

Parametric and non-parametric methods

Parametric methods:

❖The algorithms involve two steps:

- *Select a form for the function.*
- *Learn the coefficients for the function from the training data.*

A. Linear Regression: It assumes a linear relationship between the input variables and the target variable and estimates the coefficients of the linear equation.

B. Naive Bayes: It assumes that the features are conditionally independent given the class label and estimates the class probabilities using Bayes' theorem.

Non-parametric Methods

- These methods make minimal assumptions about the data's underlying structure. They learn the model(functional form) directly from the data, allowing for more complex relationships.
- Non-parametric methods don't rely on **predefined forms**, and often have a flexible number of parameters that grow with the amount of training data. The model complexity can grow with more data, making them adaptable.
- If your data is complex or you're unsure about its distribution(no prior knowledge), non-parametric methods offer more flexibility.

Feature	Parametric Methods	Non-parametric Methods
Model form	Predefined with a fixed number of parameters	Data-driven learning: -Learn from the data
Assumptions about data distribution	Strong assumptions (e.g., normal distribution)	Weak or no assumptions
Data needs	Since they have a predefined structure, they can be trained effectively with smaller datasets.	Since they don't have a rigid structure, they often need more data for effective training to capture the complexities.
Interpretability	The parameters in a parametric model have clear meanings, making it easier to understand how the model arrives at its predictions.	The inner workings of non-parametric models might be less transparent and challenging to understand.
Statistical power	Potentially higher (if assumptions hold)	Lower
Example	Linear regression, logistic regression, Naïve Bayes, Simple Neural network	KNN, SVM, Random Forest, Decision Trees like CART and C4.5

Choosing the right method:

The best method depends on your specific problem and data:

- If you have a strong understanding of your data and can make reasonable assumptions about its distribution, parametric methods can be a good choice, especially when data is limited.


thank
you!