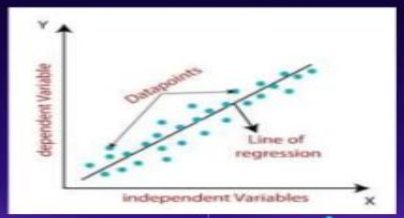
# What is Regression?

Regression is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables.

It is mainly used for prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables. Regression analysis helps in the prediction of a continuous variable.

# What is Linear Regression?

- Linear regression analysis is used to predict the value of a variable based on the value of another variable
- Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between a dependent variable and one or more independent features.

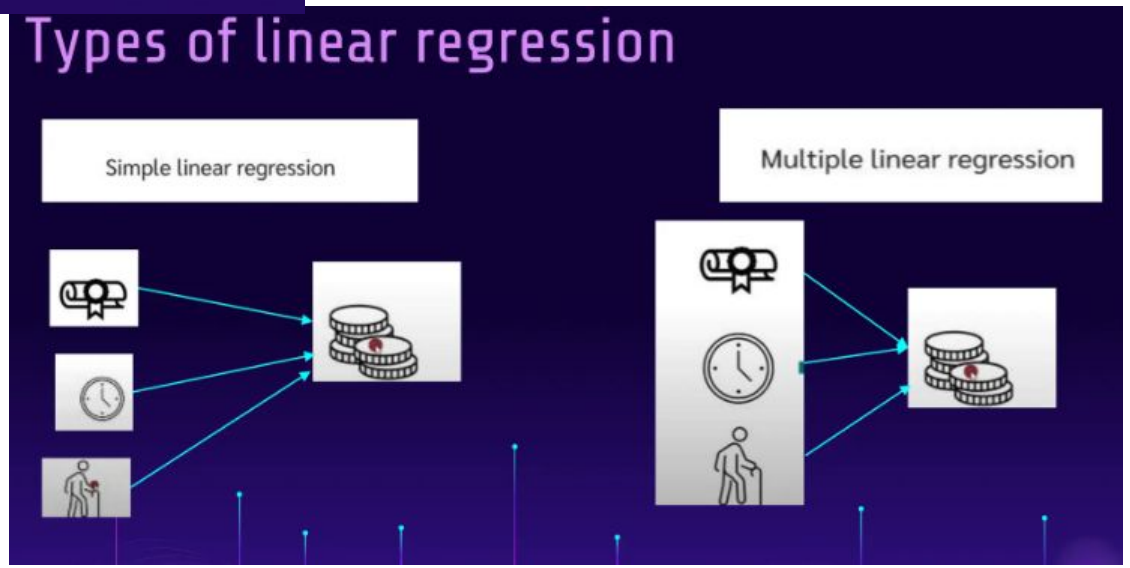


# Types of linear regression

- **Simple linear regression :** The goal of a simple linear regression is to predict the value of a dependent variable based on an independent variable. Simple linear regression is used to model the relationship between two continuous variables. Often, the objective is to predict the value of an output variable (or response) based on the value of an input (or predictor) variable.

- **Multiple linear regression: Multiple linear regression is used to estimate the relationship between two or more independent variables and one dependent variable.**


# Types of linear regression

# Equation of Simple linear regression

$$Y = mX + b$$

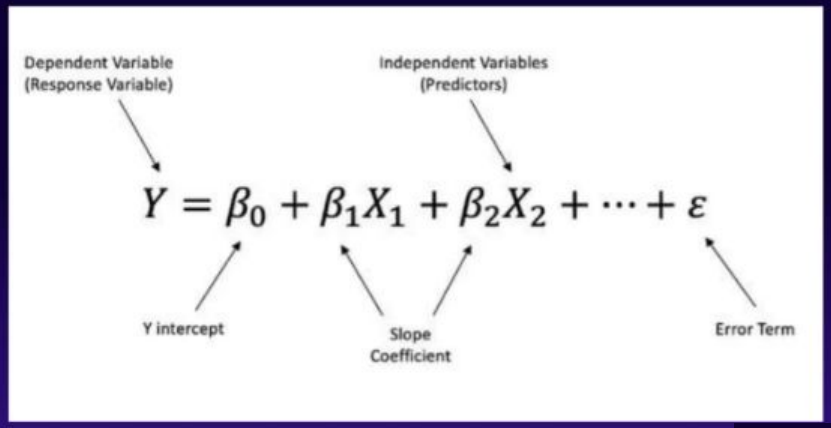
- Y represents the dependent variable
- X represents the independent variable
- m is the slope of the line( how much Y changes for a unit change in X)
- b is the intercept ( the value of Y when X is 0)

# Example : Predicting Pizza Prices

| Diameter (X) in Inches | Price(Y) in Dollars |
|---|---|
| 8 | 10 |
| 10 | 13 |
| 12 | 16 |

What will be the 20-inch pizza price?

# Multiple linear regression formula




# Example of Multiple Linear Regression

| x1 Product 1 Sales | x2 Product 2 Sales | Y Weekly Sales |
|---|---|---|
| 1 | 4 | 1 |
| 2 | 5 | 6 |
| 3 | 8 | 8 |
| 4 | 2 | 12 |

Here, the Matrices for Y and X are given as follows

$$X = \begin{pmatrix} 1 & 1 & 4 \\ 1 & 2 & 5 \\ 1 & 3 & 8 \\ 1 & 4 & 2 \end{pmatrix} \text{ and } Y = \begin{pmatrix} 1 \\ 6 \\ 8 \\ 12 \end{pmatrix}$$

# EVALUATING PERFORMANCE OF Supervised learning – Regression

- A well-fitted regression model churns out **predicted values close to actual values**.
- Regression model which **ensures that the difference between predicted and actual values is low** can be considered as a good model.

$$y = \alpha + \beta x$$

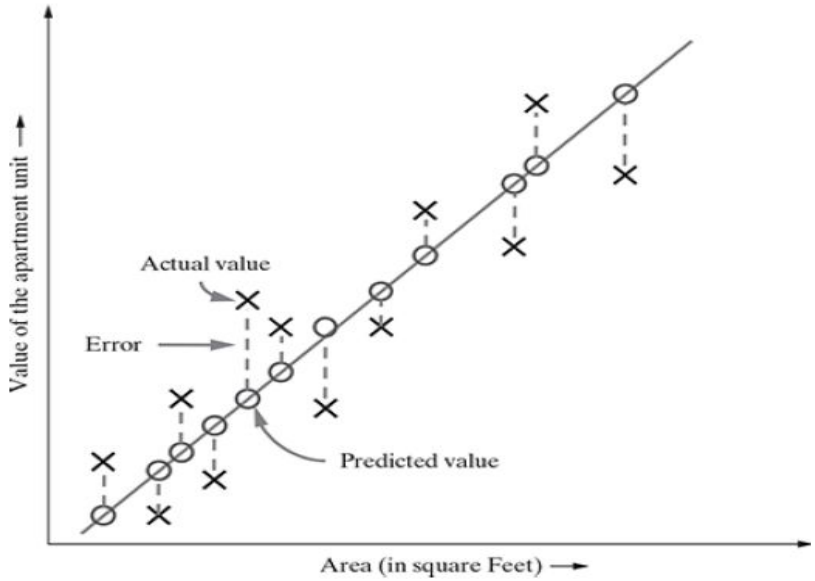



FIG. 3.9 Error – Predicted vs. actual value

- **Sum of Squared Errors (SSE)-**

sum of the squared residuals $= \sum_{i=1}^{n} (Y_i - \hat{y})^2$

Where

$\hat{y}_i$ is the predicted value of $y$

$Y_i$ is the actual value

- **Sum of Squares Total (SST)-**

squared differences of each observation from the overall mean

**SST** $= \sum_{i=1}^{n} (y_i - \bar{y})^2$ Where $\bar{y}$ is the mean of y.

**R-squared-**good measure to evaluate the model fitness. It is also known as the "coefficient of determination for multiple regression" The R-squared value lies between 0 to 1 (0%–100%) with a larger value representing a better fit.

$$R^2 = \frac{SST - SSE}{SST}$$

# Types of Cost Function in Linear Regression

Root mean squared error (RMSE): The Root Mean Squared Error (RMSE) is one of the two main performance indicators for a regression model. It measures the average difference between values predicted by a model and the actual values.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Mean Absolute Error: MSE is a variation of MAE that squares the difference instead of taking the absolute value of the difference. There is no possibility of negative errors.

I = index of sample
Ŷ = predicted value
Y = expected value
M = number of samples in the data set

$$MAE = \frac{1}{m} \sum_{i=1}^{m} |\hat{y}^{(i)} - y^{(i)}|$$

**Mean error:** These errors can be negative or positive. Therefore, they can cancel each other out during the summation, and the average error of the model will be zero.

Mean Squared Error : MSE represents the average squared difference between the predictions and expected results.

$$MSE = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

I = index of sample
Ŷ = predicted value
Y = expected value
M = number of samples in the data set

# Advantages of Linear Regression

- Linear Regression is simple to implement and easier to interpret the output coefficients
- When you know the relationship between the independent and dependent variable have a linear relationship, this algorithm is the best to use because of it's less complexity compared to other algorithms.
- Linear Regression is susceptible to over-fitting but it can be avoided using some dimensionality reduction techniques, regularization (L1 and L2) techniques and cross-validation.

A second hand car dealer has 10 cars for sale. She decides to investigate the link between the age of the cars, $x$ years, and the mileage, $y$ thousand miles. The data collected from the cars are shown in the table below.

| Age, $x$ (years) | 2 | 2.5 | 3 | 4 | 4.5 | 4.5 | 5 | 3 | 6 | 6.5 |
|---|---|---|---|---|---|---|---|---|---|---|
| Mileage, $y$ (thousands) | 22 | 34 | 33 | 37 | 40 | 45 | 49 | 30 | 58 | 58 |

[You may assume that $\sum x = 41, \sum y = 406, \sum x^2 = 188, \sum xy = 1818.5$]

(b) Find the equation of the least squares regression line in the form $y=a+bx$. Give the values of $a$ and $b$ to 2 decimal places.

(d) Using your answer to part (b), find the mileage predicted by the regression line for a 5 year old car.

**And evaluate the performance using SST, SSE and R-square value**

# DECISION TREE

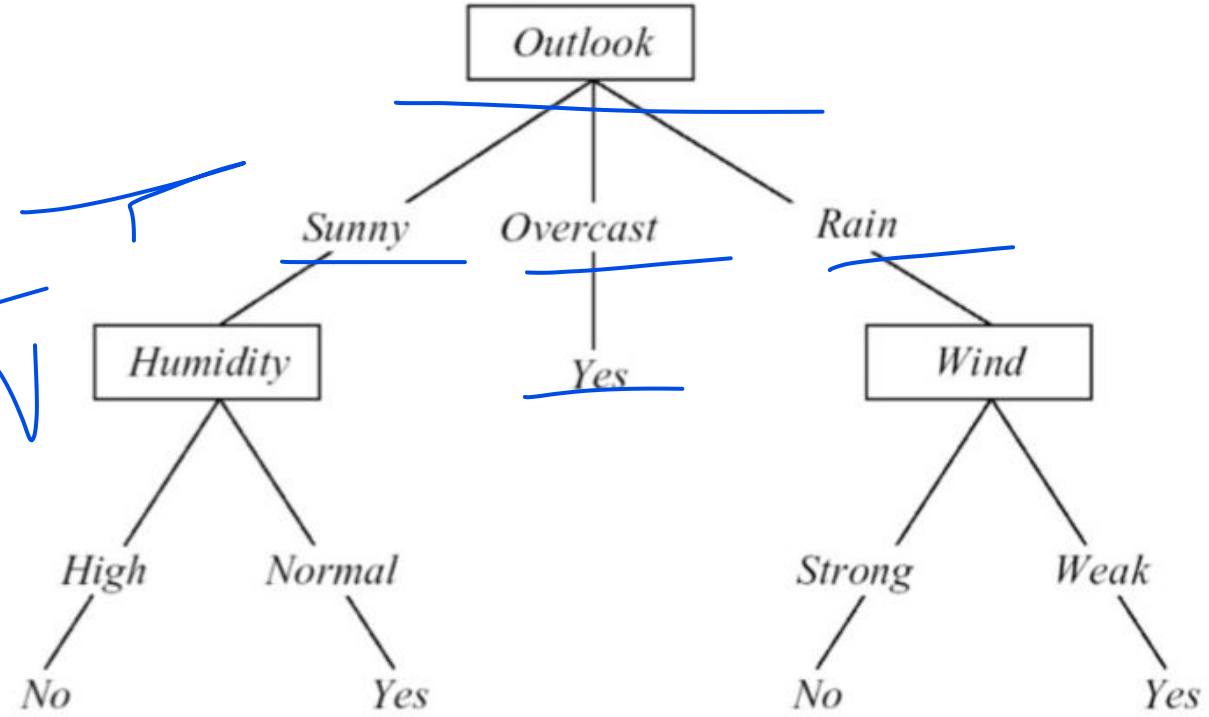
# Abstract

- Decision tree representation
- ID3 learning algorithm
- Entropy, Information gain
- Overfitting

# Decision Trees

- Decision tree is a simple but powerful learning Paradigm.
- It is a type of classification algorithm for supervised learning.
- A decision tree is a tree in which each branch node represents a choice between a number of alternatives and each leaf node represents a decision.
- A node with outgoing edges is called an internal or test node. All other nodes are called leaves (also known as terminal or decision nodes).

# Decision Tree for *PlayTennis*




- Decision tree representation:
  - Each internal node tests an attribute
  - Each branch corresponds to attribute value
  - Each leaf node assigns a classification

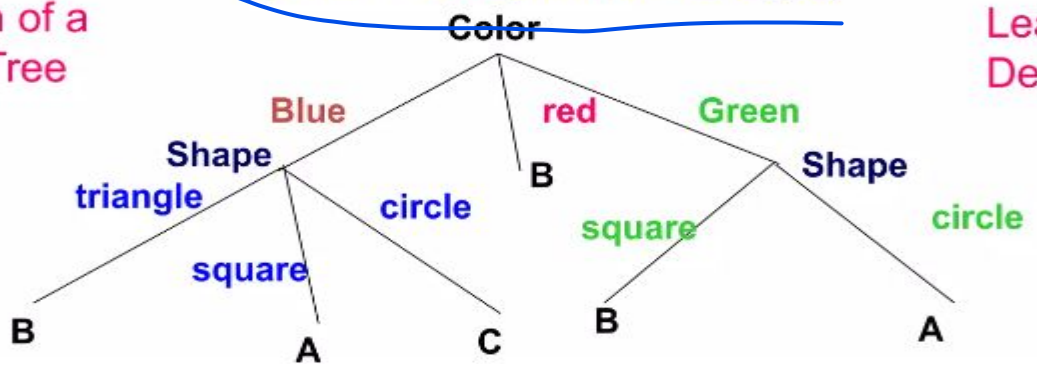
# DECISION TREE REPRESENTATION

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.

- Decision Trees are classifiers for instances represented as features vectors. (color = ;shape = ;label = )
- Nodes are tests for feature values;
- There is one branch for each value of the feature
- Leaves specify the categories (labels)
- Can categorize instances into multiple disjoint categories – multi-class

(color = RED ;shape = triangle)

Evaluation of a Decision Tree

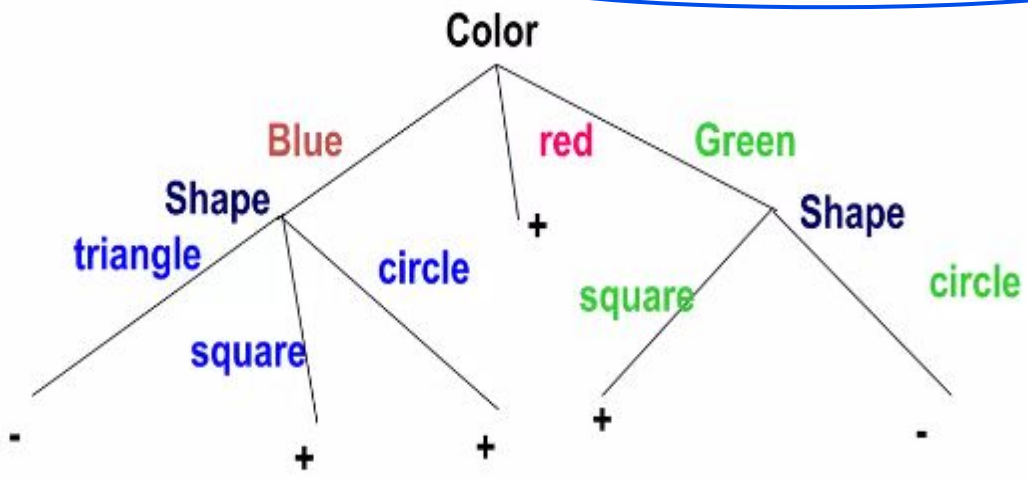



Learning a Decision Tree

# Binary classified Decision Trees

They can represent any Boolean function

- Can be rewritten as rules in Disjunctive Normal Form (DNF)
- green ∧ square →positive
- blue ∧ circle → positive
- blue ∧ square → positive
- The disjunction of these rules is equivalent to the Decision Tree



# When to Consider Decision Trees

- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:

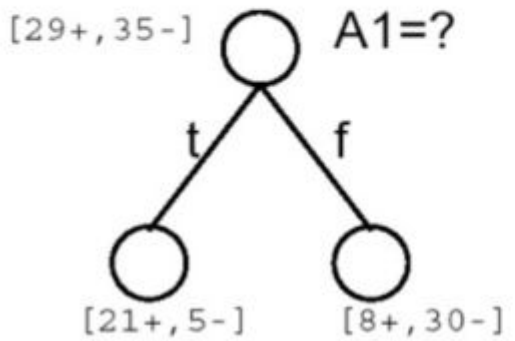
- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

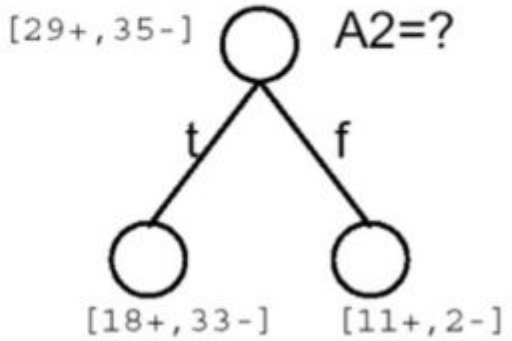
# ID3 (Iterative Dichotomiser 3)

- Invented by J.Ross Quinlan in 1975.
- Used to generate a decision tree from a given data set by employing a top-down, greedy search, to test each attribute at every node of the tree.
- The resulting tree is used to classify future samples.

# Top-Down Induction of Decision Trees

- Main loop:
  1. *A* ← the "best" decision attribute for next *node*
  2. Assign *A* as decision attribute for *node*
  3. For each value of *A*, create new descendant of *node*
  4. Sort training examples to leaf nodes
  5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes
- Which attribute is best?



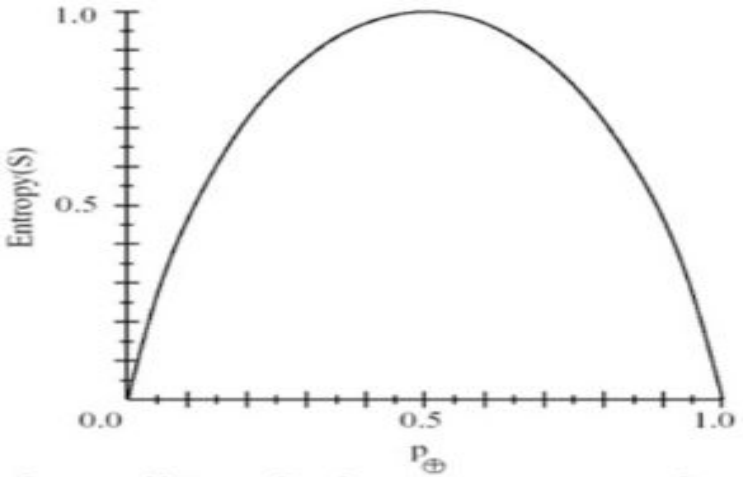



# Algorithm

- Calculate the Entropy of every attribute using the data set.
- Split the set into subsets using the attribute for which entropy is minimum (or, equivalently, information gain is maximum).
- Make a decision tree node containing that attribute.
- Recurse on subsets using remaining attributes.

# Entropy

- In order to define Information Gain precisely, we need to discuss Entropy first.
- A formula to calculate the homogeneity of a sample.
- A completely homogeneous sample has entropy of 0 (leaf node).
- An equally divided sample has entropy of 1.




- $S$ is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in $S$
- $p_\ominus$ is the proportion of negative examples in $S$
- Entropy measures the impurity of $S$

$$Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$

# Entropy

- Example 1

If S is a collection of 14 examples with 9 YES and 5 NO examples then

$$\text{Entropy}(S) = -(9/14)\,\text{Log}_2(9/14) - (5/14)\,\text{Log}_2(5/14)$$

$$= 0.940$$

# Information Gain (IG)

- The information gain is based on the decrease in entropy after a dataset is split on an attribute.
- The formula for calculating information gain is:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Where:

- $S_v$ = subset of S for which attribute A has value v
- $|S_v|$ = number of elements in $S_v$
- $|S|$ = number of elements in S

Training Set

| Attributes | | | | Classes |
|---|---|---|---|---|
| Gender | Car Ownership | Travel Cost | Income Level | Transportation |
| Male | 0 | Cheap | Low | Bus |
| Male | 1 | Cheap | Medium | Bus |
| Female | 1 | Cheap | Medium | Train |
| Female | 0 | Cheap | Low | Bus |
| Male | 1 | Cheap | Medium | Bus |
| Male | 0 | Standard | Medium | Train |
| Female | 1 | Standard | Medium | Train |
| Female | 1 | Expensive | High | Car |
| Male | 2 | Expensive | Medium | Car |
| Female | 2 | Expensive | High | Car |

# Calculate the entropy of the total dataset

- First compute the Entropy of given training set.

Probability
Bus : 4/10 = 0.4
Train: 3/10 = 0.3
Car:3/10 = 0.3

$$E(S) = -P(I)\ log_2\ P(I)$$
$$E(S) = -(0.4)\ log_2\ (0.4) - (0.3)log_2\ (0.3) - (0.3)log_2\ (0.3) = 1.571$$

| Attributes | | | | Classes |
|---|---|---|---|---|
| Gender | Car Ownership | Travel Cost | Income Level | Transportation |
| Male | 0 | Cheap | Low | Bus |
| Male | 1 | Cheap | Medium | Bus |
| Female | 1 | Cheap | Medium | Train |
| Female | 0 | Cheap | Low | Bus |
| Male | 1 | Cheap | Medium | Bus |
| Male | 0 | Standard | Medium | Train |
| Female | 1 | Standard | Medium | Train |
| Female | 1 | Expensive | High | Car |
| Male | 2 | Expensive | Medium | Car |
| Female | 2 | Expensive | High | Car |

| Attributes | Classes |
|---|---|
| Gender | Transportation |
| Male | Bus |
| Male | Bus |
| Female | Train |
| Female | Bus |
| Male | Bus |
| Male | Train |
| Female | Train |
| Female | Car |
| Male | Car |
| Female | Car |

# Split the dataset on 'Gender' attribute

| Attributes | Classes |
|---|---|
| Gender | Transportation |
| Male | Bus |
| Male | Bus |
| Male | Bus |
| Male | Train |
| Male | Car |

Probability
Bus : 3/5 = 0.6
Train: 1/5 = 0.2
Car:1/5 = 0.2

$E(S_v) = -0.6 \log_2(0.6) - 0.2 \log_2(0.2) - 0.2 \log_2(0.2)$
$= 1.522$

| Attributes | Classes |
|---|---|
| Gender | Transportation |
| Female | Train |
| Female | Bus |
| Female | Train |
| Female | Car |
| Female | Car |

Probability
Bus : 1/5 = 0.2
Train: 2/5 = 0.4
Car:2/5 = 0.4

$E(S_v) = -0.2 \log_2(0.2) - 0.4 \log_2(0.4) - 0.4 \log_2(0.4)$
$= 1.371$

$Gain(S,A) = E(S) - I(S,A)$
$I(S,A) = 1.522 * (5/10) + 1.371 * (5/10)$

$Gain(S,A) = 1.571 - 1.447$
$= 0.12$

| Attributes | | | | Classes |
|---|---|---|---|---|
| Gender | Car Ownership | Travel Cost | Income Level | Transportation |
| Male | 0 | Cheap | Low | Bus |
| Male | 1 | Cheap | Medium | Bus |
| Female | 1 | Cheap | Medium | Train |
| Female | 0 | Cheap | Low | Bus |
| Male | 1 | Cheap | Medium | Bus |
| Male | 0 | Standard | Medium | Train |
| Female | 1 | Standard | Medium | Train |
| Female | 1 | Expensive | High | Car |
| Male | 2 | Expensive | Medium | Car |
| Female | 2 | Expensive | High | Car |

| Attributes | Classes |
|---|---|
| Car Ownership | Transportation |
| 0 | Bus |
| 1 | Bus |
| 1 | Train |
| 0 | Bus |
| 1 | Bus |
| 0 | Train |
| 1 | Train |
| 1 | Car |
| 2 | Car |
| 2 | Car |

# Split the dataset on 'ownership' attribute

| Attributes | Classes |
|---|---|
| Car Ownership | Transportation |
| 0 | Bus |
| 0 | Bus |
| 0 | Train |

| Attributes | Classes |
|---|---|
| Car Ownership | Transportation |
| 1 | Bus |
| 1 | Train |
| 1 | Bus |
| 1 | Train |
| 1 | Car |

| Attributes | Classes |
|---|---|
| Car Ownership | Transportation |
| 2 | Car |
| 2 | Car |

*Gain(S,A) = E(S) – I(S,A)*
*I(S,A) = 0.918 *(3/10) + 1.522*(5/10) + 0*(2/10)*

*Gain(S,A) = 1.571 – 1.0364 = 0.534*

Probability
Bus : 2/3= 0.6
Train: 1/3 = 0.3
Car:0/3 = 0
Entropy = 0.918

Probability
Bus : 2/5 = 0.4
Train: 2/5 = 0.4
Car:1/5 = 0.2
Entropy = 1.522

Probability
Bus : 0/2 = 0
Train: 0/2 = 0
Car:2/2 = 1
Entropy = 0

- If we choose Travel Cost as splitting attribute,

-Entropy for Cheap = 0.722

Standard = 0

Expensive = 0

IG = 1.21

- If we choose Income Level as splitting attribute,

-Entropy for Low = 0

Medium = 1.459

High = 0

IG = 0.695

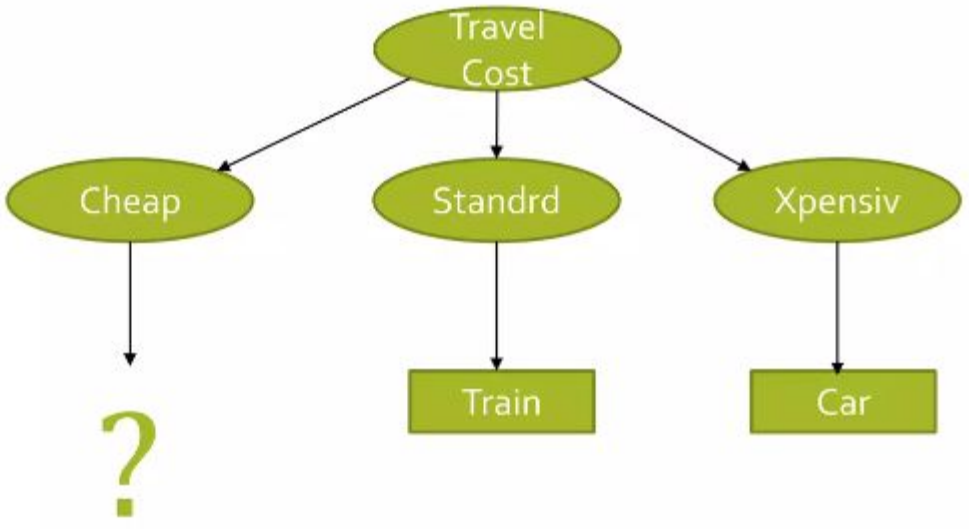
| Attribute | Information Gain |
|---|---|
| Gender | 0.125 |
| Car Ownership | 0.534 |
| Travel Cost | 1.21 |
| Income Level | 0.695 |

| Attributes | | | | Classes |
|---|---|---|---|---|
| Travel Cost | Gender | Car Ownership | Income Level | Transportation |
| Cheap | Male | 0 | Low | Bus |
| Cheap | Male | 1 | Medium | Bus |
| Cheap | Female | 1 | Medium | Train |
| Cheap | Female | 0 | Low | Bus |
| Cheap | Male | 1 | Medium | Bus |

| Attributes | | | | Classes |
|---|---|---|---|---|
| Travel Cost | Gender | Car Ownership | Income Level | Transportation |
| Standard | Male | 0 | Medium | Train |
| Standard | Female | 1 | Medium | Train |

| Attributes | | | | Classes |
|---|---|---|---|---|
| Travel Cost | Gender | Car Ownership | Income Level | Transportation |
| Expensive | Female | 1 | High | Car |
| Expensive | Male | 2 | Medium | Car |
| Expensive | Female | 2 | High | Car |

# Diagram : Decision Tree



# Iteration on Subset of Training Set

| Attributes | | | | Classes |
|---|---|---|---|---|
| **Travel Cost** | **Gender** | **Car Ownership** | **Income Level** | **Transportation** |
| Cheap | Male | 0 | Low | Bus |
| Cheap | Male | 1 | Medium | Bus |
| Cheap | Female | 1 | Medium | Train |
| Cheap | Female | 0 | Low | Bus |
| Cheap | Male | 1 | Medium | Bus |

Probability
Bus : 4/5 = 0.8
Train: 1/5 = 0.2
Car:0/5 = 0

$$E(S) = -P(I) \log_2 P(I)$$
$$E(S) = -(0.8) \log_2 (0.8) - (0.2) \log_2 (0.2) = 0.722$$

- If we choose Gender as splitting attribute,

-Entropy for Male = 0

Female = 1

IG = 0.322

- If we choose Car Ownership as splitting attribute,

-Entropy for 0 = 0

1 = 0.918

IG = 0.171

- If we choose Income Level as splitting attribute,

-Entropy for Low = 0

Medium = 0.918

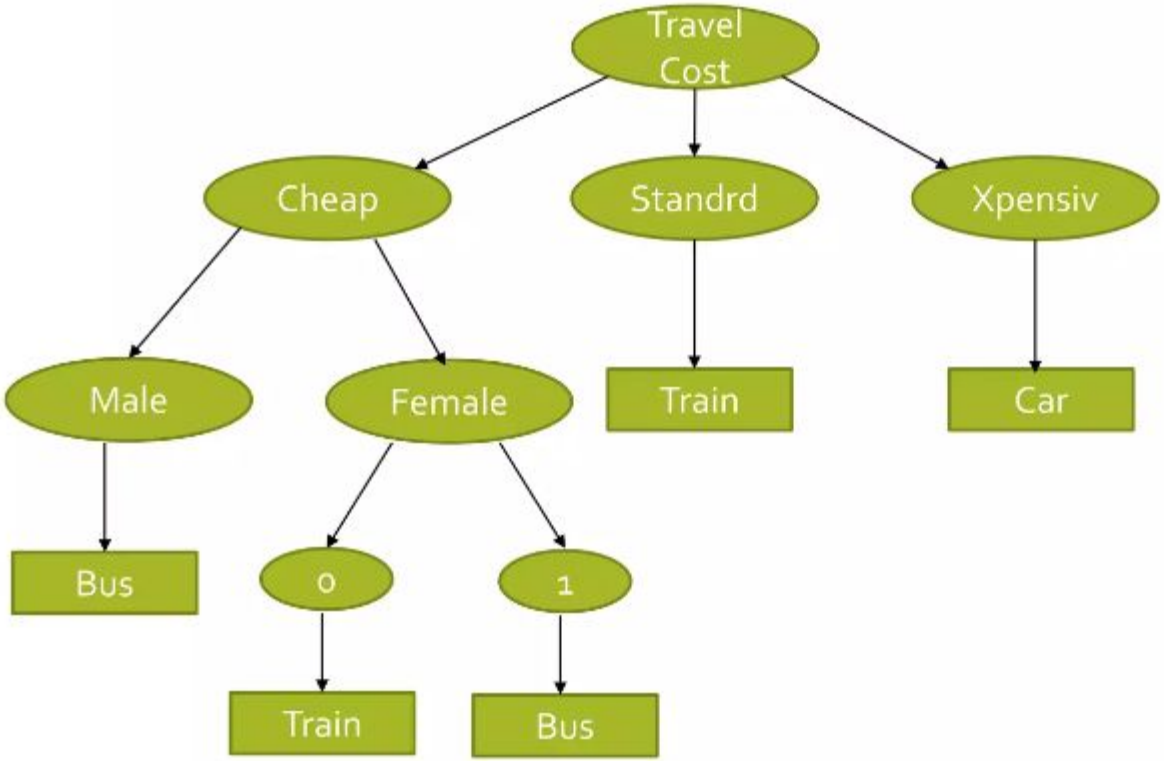
IG = 0.171

| Attributes | Information Gain |
|---|---|
| Gender | 0.322 |
| Car Ownership | 0.171 |
| Income Level | 0.171 |

| Attributes | | | Classes |
|---|---|---|---|
| Gender | Car Ownership | Income Level | Transportation |
| Male | 0 | Low | Bus |
| Male | 1 | Medium | Bus |
| Male | 1 | Medium | Bus |

| Attributes | | | Classes |
|---|---|---|---|
| Gender | Car Ownership | Income Level | Transportation |
| Female | 1 | Medium | Train |
| Female | 0 | Low | Bus |

# Diagram : Decision Tree



# Our Problem :

| Name | Gender | Car Ownership | Travel Cost | Income Level | Transportation |
|---|---|---|---|---|---|
| Abhi | Male | 1 | Standard | High | ? |
| Pavi | Male | 0 | Cheap | Medium | ? |
| Ammu | Female | 1 | Cheap | High | ? |

# Solution to Our Problem :

| Name | Gender | Car Ownership | Travel Cost | Income Level | Transportation |
|---|---|---|---|---|---|
| Abhi | Male | 1 | Standard | High | Train |
| Pavi | Male | 0 | Cheap | Medium | Bus |
| Ammu | Female | 1 | Cheap | High | Bus |

# Drawbacks of ID3

- Data may be over-fitted or over-classified, if a small sample is tested.
- Only one attribute at a time is tested for making a decision.
- Classifying continuous data may be computationally expensive.
- Does not handle numeric attributes and missing values.

# C4.5 Algorithm

- This algorithm is also invented by J.Ross Quinlan., for generating decision tree.
- It can be treated as an extension of ID3.
- C4.5 is mainly used for classification of data.
- It is called statistical classifier because of its potential for handling noisy data.

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

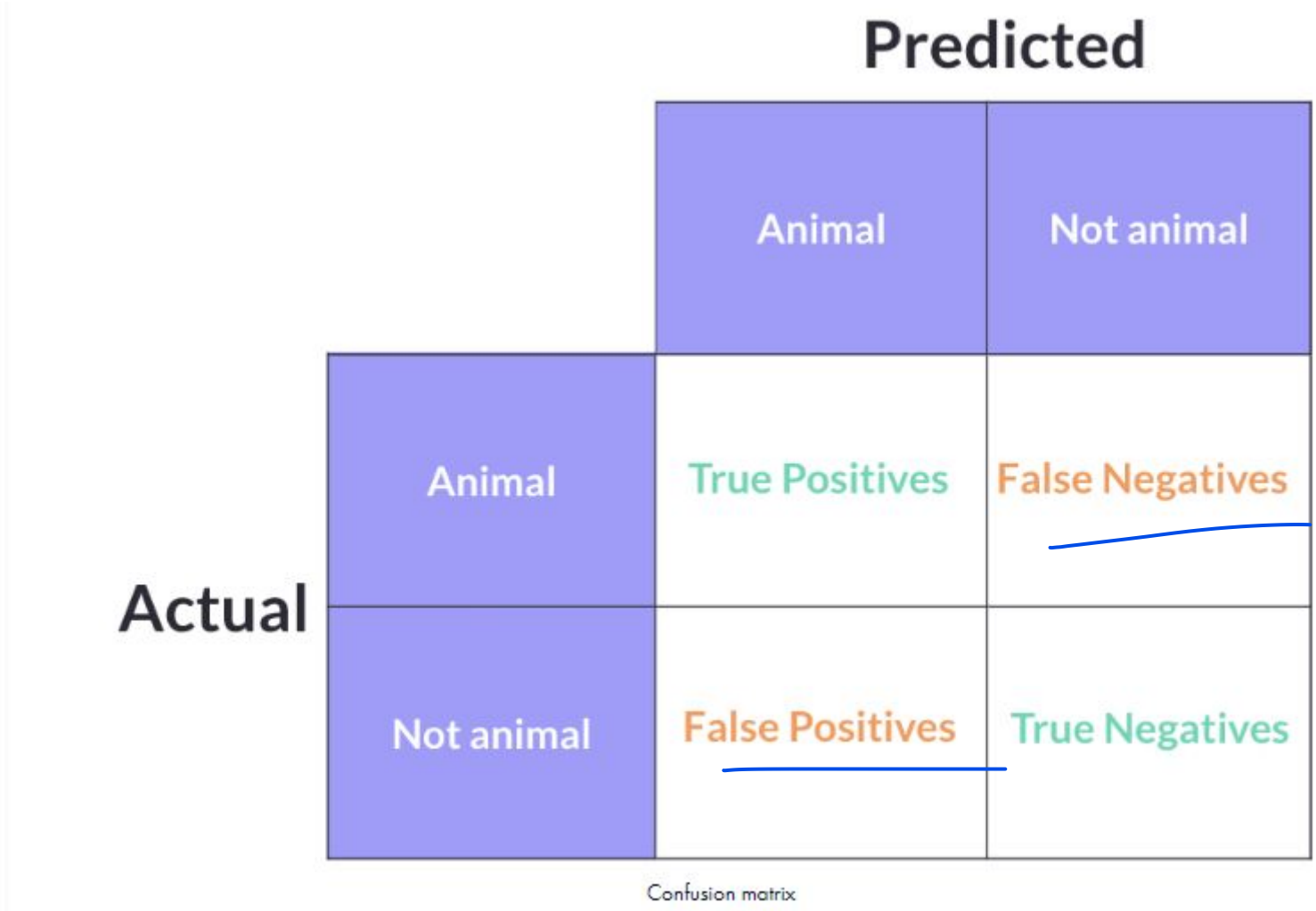
# Evaluation metrics

If we talk about classification problems, the most common metrics used are:

- Accuracy
- Precision (P)
- Recall (R)
- F1 score (F1)
- Area under the ROC (Receiver Operating Characteristic) curve or simply
AUC (AUC),- Log loss
- Precision at k (P@k) - Average precision at k (AP@k)
- Mean average precision at k (MAP@k)

| | |
|---|---|
| Animal |  Barn owl<br>  Chihuahua<br>  Sheep dog |
| Not animal |  Apple<br>  Muffin<br>  Mop |

Animals and things that look like animals

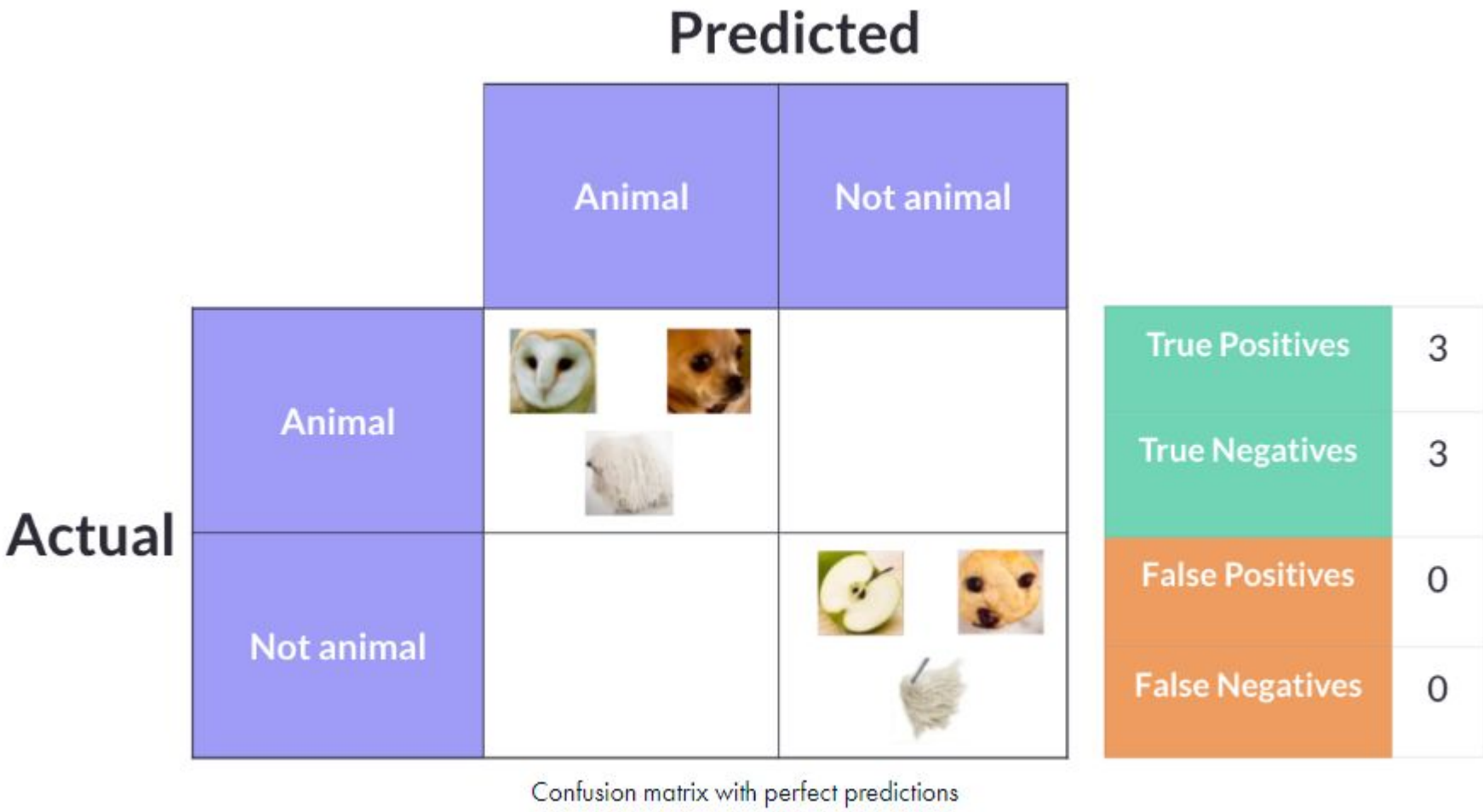



Confusion matrix

# TP / FP / FN / TN

The counts of **true positive** (TP), **false positive** (FP), **false negative** (FN), and **true negative** (TN) ground truths and inferences are essential for summarizing model performance. These metrics are the building blocks of many other metrics, including accuracy, precision, and recall.

| Metric | | Description |
|---|---|---|
| True Positive | TP | An instance for which both predicted and actual values are positive |
| False Positive | FP | An instance for which predicted value is positive but actual value is negative |
| False Negative | FN | An instance for which predicted value is negative but actual value is positive |
| True Negative | TN | An instance for which both predicted and actual values are negative |




Confusion matrix with perfect predictions

| | | Predicted: Animal | Predicted: Not animal |
|---|---|---|---|
| Actual | Animal |   |  |
| | Not animal |  |   |

| | |
|---|---|
| True Positives | 2 |
| True Negatives | 2 |
| False Positives | 1 |
| False Negatives | 1 |

Confusion matrix with imperfect predictions

**Accuracy**

Accuracy is the most common metric to be used in everyday talk. Accuracy answers the question **"Out of all the predictions we made, how many were true?"**

$$accuracy = \frac{true\ positives + true\ negatives}{true\ positives + true\ negatives + false\ negatives + false\ positives}$$

As we will see later, accuracy is a blunt measure and can sometimes be misleading.

**Precision**

Precision is a metric that gives you the proportion of true positives to the amount of total positives that the model predicts. It answers the question **"Out of all the positive predictions we made, how many were true?"**

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

**Recall**

Recall focuses on how good the model is at finding all the positives. Recall is also called true positive rate and answers the question "**Out of all the data points that should be predicted as true, how many did we correctly predict as true?**"

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

As you can see from the definitions of precision and recall they are tightly connected.

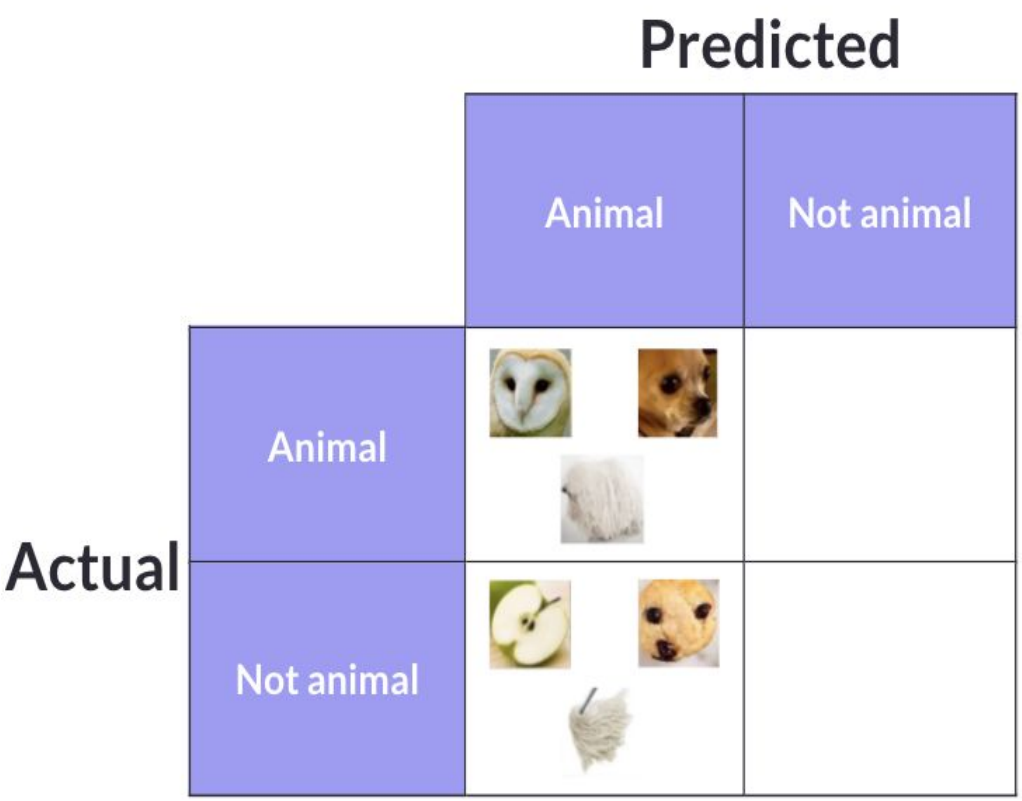
**F1 Score**

F1 Score is a measure that combines recall and precision. As we have seen there is a trade-off between precision and recall, F1 can therefore be used to measure how effectively our models make that trade-off.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

One important feature of the F1 score is that the result is zero if any of the components (precision or recall) fall to zero. Thereby it penalizes extreme negative values of either component.

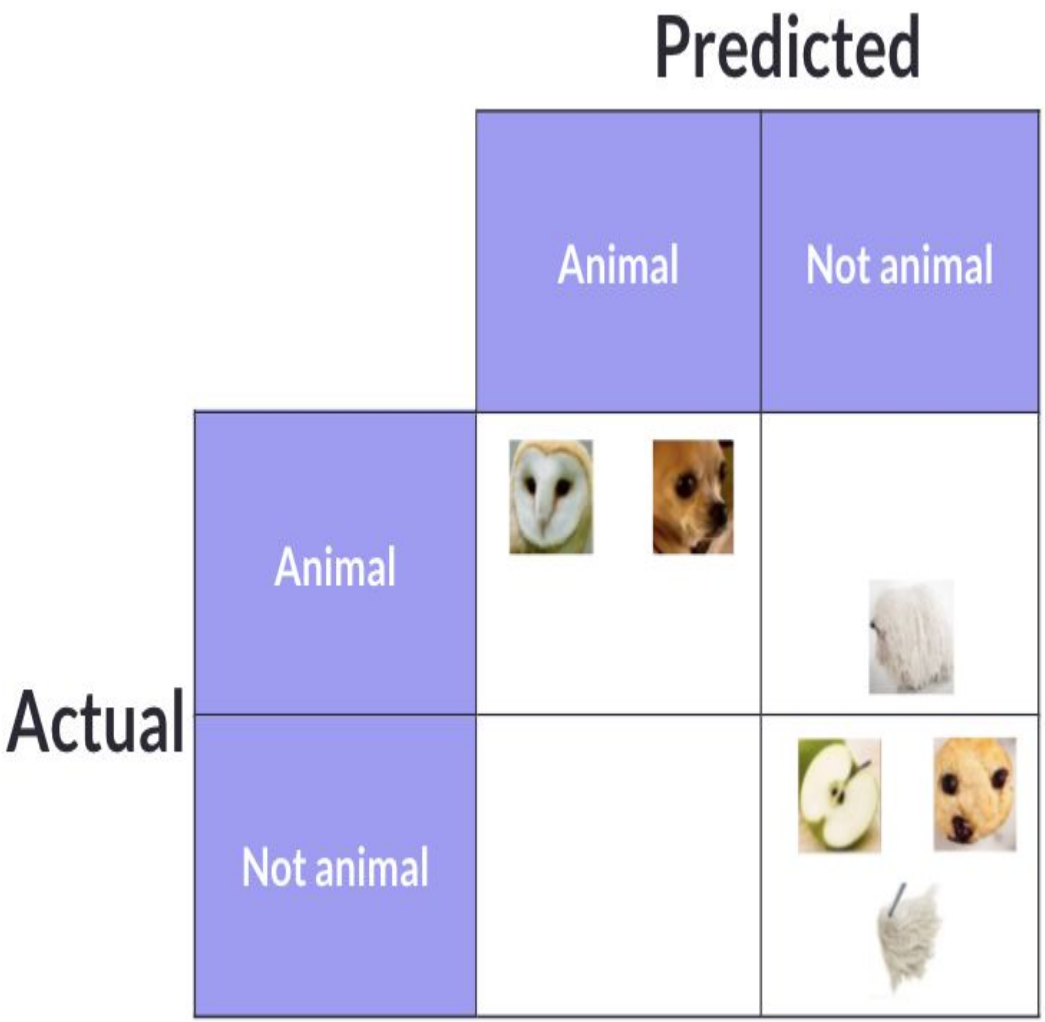
## Model 1 (Classifies all images as animal)




| | |
|---|---|
| True Positives | 3 |
| True Negatives | 0 |
| False Positives | 3 |
| False Negatives | 0 |

| | | |
|---|---|---|
| Accuracy | 50% | $\frac{3+0}{3+0+0+3}$ |
| Precision | 50% | $\frac{3}{3+3}$ |
| Recall | 100% | $\frac{3}{3+0}$ |
| F1 score | 67% | $2 . \frac{0.5 \cdot 1}{0.5+1}$ |

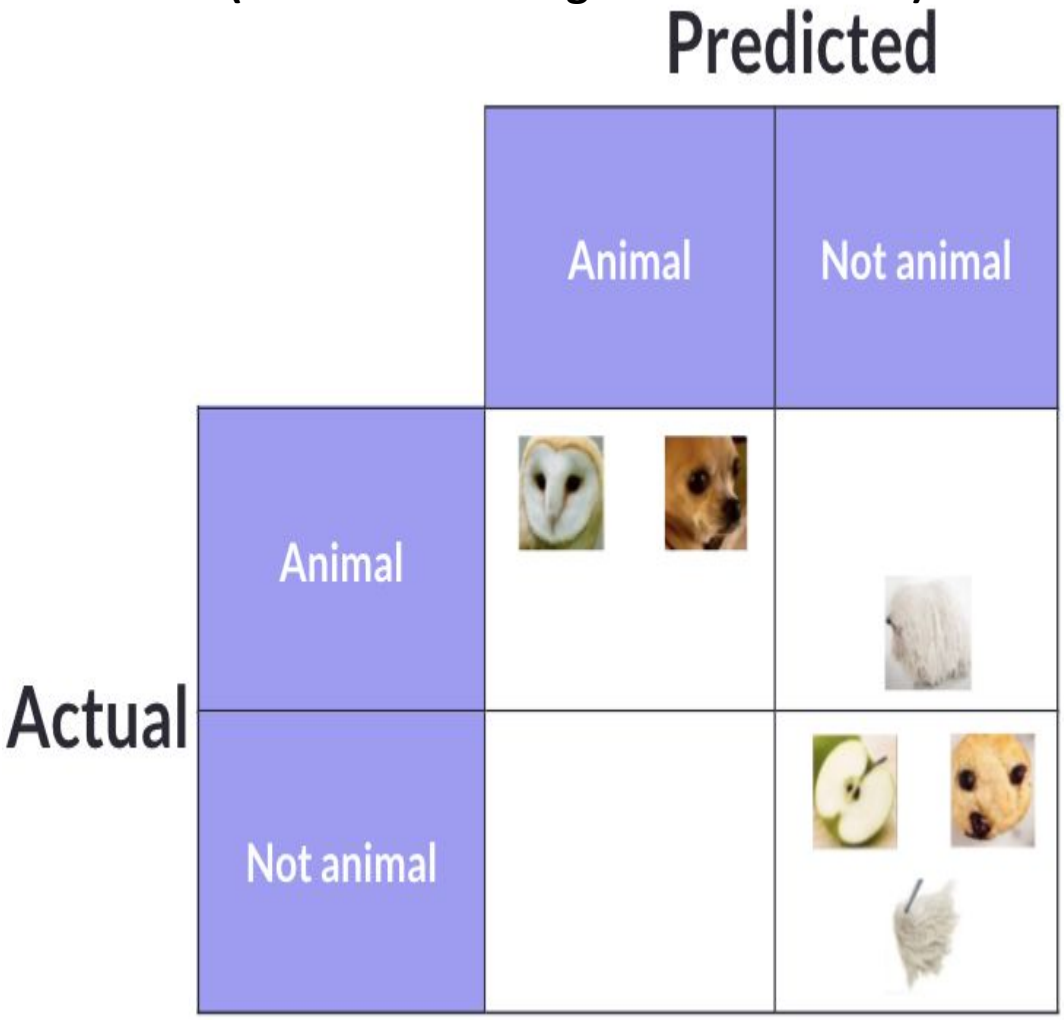
**Model 2 (Classifies all images as not animal)**




| | |
|---|---|
| True Positives | 2 |
| True Negatives | 3 |
| False Positives | 0 |
| False Negatives | 1 |

| | | |
|---|---|---|
| Accuracy | 83% | $\frac{2+3}{2+3+1+0}$ |
| Precision | 100% | $\frac{2}{2+0}$ |
| Recall | 67% | $\frac{2}{2+1}$ |
| F1 score | 80% | $2 \cdot \frac{1 \cdot 0.67}{1+0.67}$ |

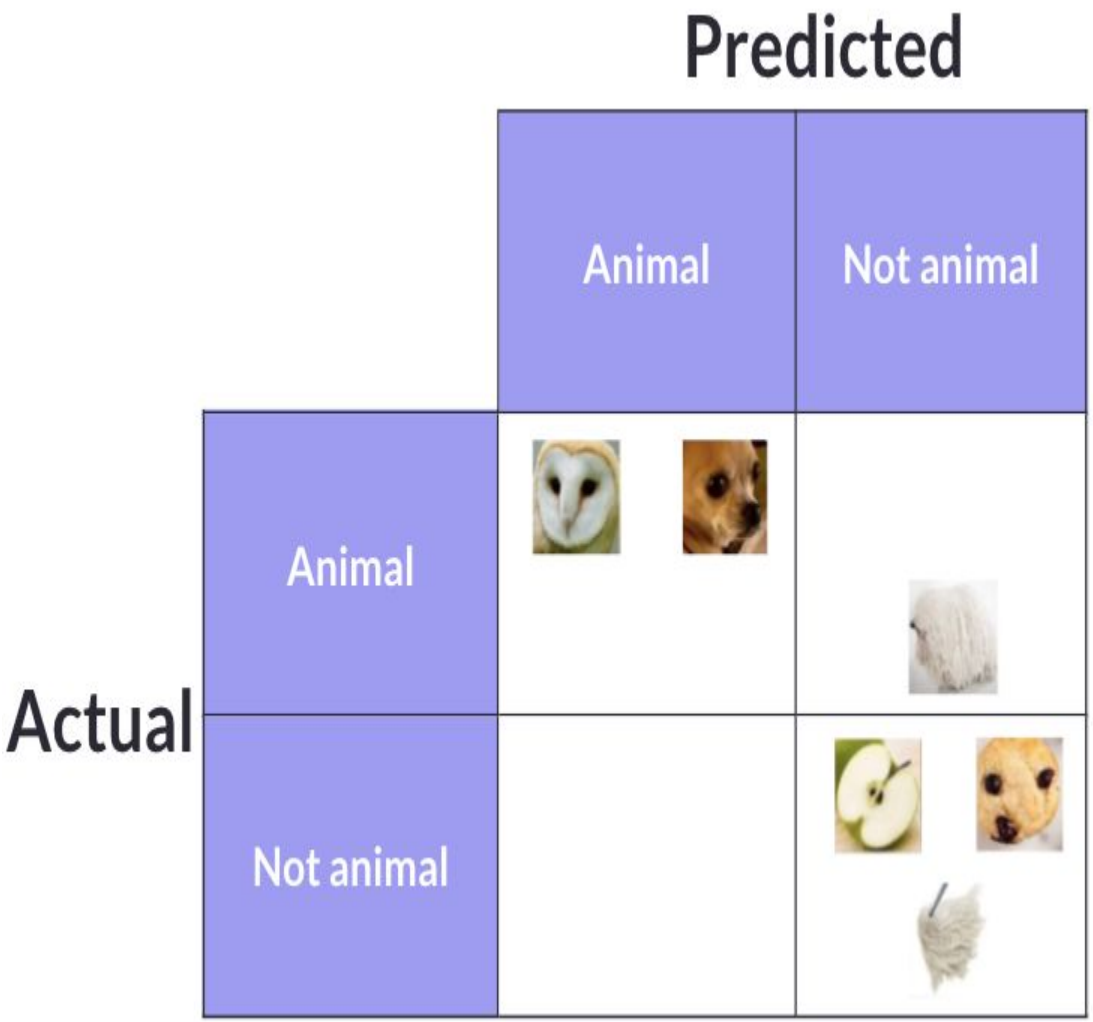
**Model 3 (Classifies all images as not animal)**




| | |
|---|---|
| True Positives | 2 |
| True Negatives | 3 |
| False Positives | 0 |
| False Negatives | 1 |

| | | |
|---|---|---|
| Accuracy | 83% | $\frac{2+3}{2+3+1+0}$ |
| Precision | 100% | $\frac{2}{2+0}$ |
| Recall | 67% | $\frac{2}{2+1}$ |
| F1 score | 80% | $2 \cdot \frac{1 \cdot 0.67}{1+0.67}$ |

**Model 4 (Classifies all images as not animal)**




| | |
|---|---|
| True Positives | 2 |
| True Negatives | 3 |
| False Positives | 0 |
| False Negatives | 1 |

| | | |
|---|---|---|
| Accuracy | 83% | $\frac{2+3}{2+3+1+0}$ |
| Precision | 100% | $\frac{2}{2+0}$ |
| Recall | 67% | $\frac{2}{2+1}$ |
| F1 score | 80% | $2 \cdot \frac{1 \cdot 0.67}{1+0.67}$ |

# APPROPRIATE PROBLEMS FOR DECISION TREE LEARNING

- ***Instances are represented by attribute-value pairs.***
  - attributes (e.g., ***Temperature)*** and their values (e.g., ***Hot)***

- ***The target function has discrete output values***
  (e.g., ***yes*** or ***no)***

- ***Disjunctive descriptions may be required.***

Can be rewritten as a set of rules, i.e. disjunctive normal form (DNF).

- red $\wedge$ circle $\rightarrow$ pos
- red $\wedge$ circle $\rightarrow$ A

blue $\rightarrow$ B; red $\wedge$ square $\rightarrow$ B

green $\rightarrow$ C; red $\wedge$ triangle $\rightarrow$ C

- ***The training data may contain errors.***

Errors in classifications of the training examples & errors in the attribute values

***The training data may contain missing attribute values.***

***values*** known for only some of the training examples

# Properties of Decision Tree Learning

- Continuous (real-valued) features can be handled by allowing nodes to split a real valued feature into two ranges based on a threshold (e.g. length < 3 and length ≥3)
- Classification trees have discrete class labels at the leaves, ***regression trees*** allow real-valued outputs at the leaves.
- Algorithms for finding consistent trees are efficient for processing large amounts of training data for data mining tasks.
- Methods developed for handling noisy training data (both class and feature noise).
- Methods developed for handling missing feature values.

# ISSUES IN DECISION TREE LEARNING

## What makes a good tree?

- Not too small – need to include enough attributes to handle possibly subtle distinctions in data
- Not too big

  - computational efficiency (avoid redundant, spurious attributes)

  - avoid over-fitting training examples (noisy, scarce data)

Occam's Razor: find simplest hypothesis (tree) that is consistent with all observations

inductive bias – small trees, with informative nodes near root

# Inductive Bias in Decision Tree Learning

## Occam's Razor

Why prefer short hypotheses?

Argument in favor:

- Fewer short hyps. than long hyps.

→ a short hyp that fits data unlikely to be coincidence

→ a long hyp that fits data might be coincidence

Argument opposed:

- There are many ways to define small sets of hyps
- e.g., all trees with a prime number of nodes that use attributes beginning with "Z"
- What's so special about small sets based on *size* of hypothesis??

# C4.5

- ID3 favors attributes with large number of divisions
  - Lead to overfitting
- Improved version of ID3:
  - Missing Data
  - Continuous Data
  - Pruning
    - **Subtree replacement by leaf node**
    - **Subtree raising**
  - Automated rule generation
  - GainRatio: take into account the cardinality of each division

# CART Split Selection Method

Motivation: We need a way to choose quantitatively between different splitting predicates

- Idea: Quantify the *impurity* of a node
- Method: Select splitting predicate that generates children nodes with minimum impurity from a space of possible splitting predicates

# CART

- If a data set $D$ contains examples from $n$ classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where $p_j$ is the relative frequency of class $j$ in $D$

- If a data set $D$ is split on A into two subsets $D_1$ and $D_2$, the *gini* index $gini(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node

# Measure of Impurity: GINI

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

- Maximum (0.5) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

| C1 | 0 |
|---|---|
| C2 | 6 |
| Gini=0.000 | |

| C1 | 1 |
|---|---|
| C2 | 5 |
| Gini=0.278 | |

| C1 | 2 |
|---|---|
| C2 | 4 |
| Gini=0.444 | |

| C1 | 3 |
|---|---|
| C2 | 3 |
| Gini=0.500 | |

# Examples for computing GINI

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

| | |
|---|---|
| C1 | **0** |
| C2 | **6** |

**P(C1) = 0/6 = 0  P(C2) = 6/6 = 1**

**Gini = 1 – P(C1)² – P(C2)² = 1 – 0 – 1 = 0**

| | |
|---|---|
| C1 | **1** |
| C2 | **5** |

**P(C1) = 1/6  P(C2) = 5/6**

**Gini = 1 – (1/6)² – (5/6)² = 0.278**

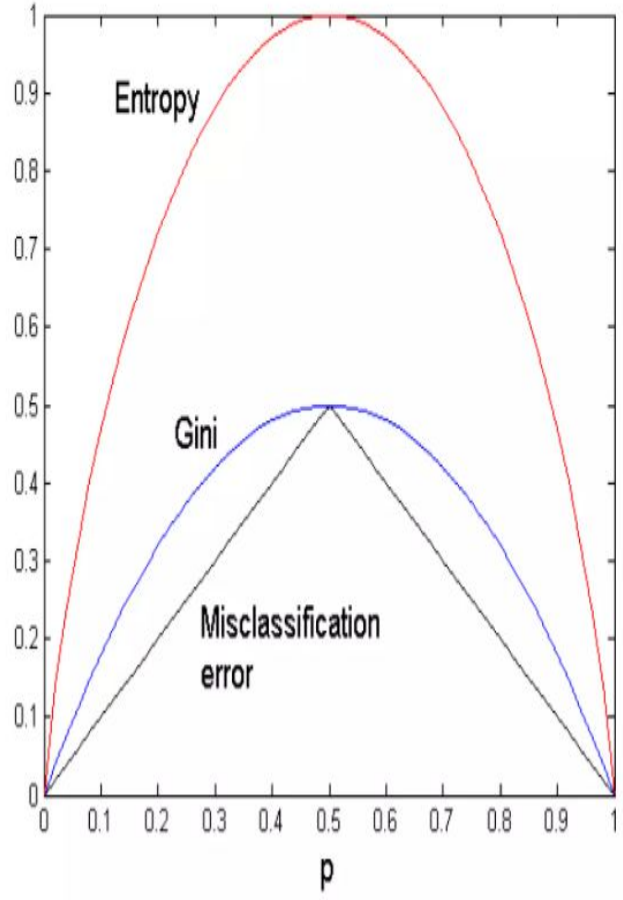
| | |
|---|---|
| C1 | **2** |
| C2 | **4** |

**P(C1) = 2/6  P(C2) = 4/6**

**Gini = 1 – (2/6)² – (4/6)² = 0.444**

# Comparison among Splitting Criteria

**For a 2-class problem:**



# CART

- Ex. D has 9 tuples in buys_computer = “yes” and 5 in “no”

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in $D_1$: {low, medium} and 4 in $D_2$

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_1)$$

$$= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right)$$

$$= 0.450$$

$$= Gini_{income \in \{high\}}(D)$$

but $gini_{\{medium,high\}}$ is 0.30 and thus the best since it is the lowest

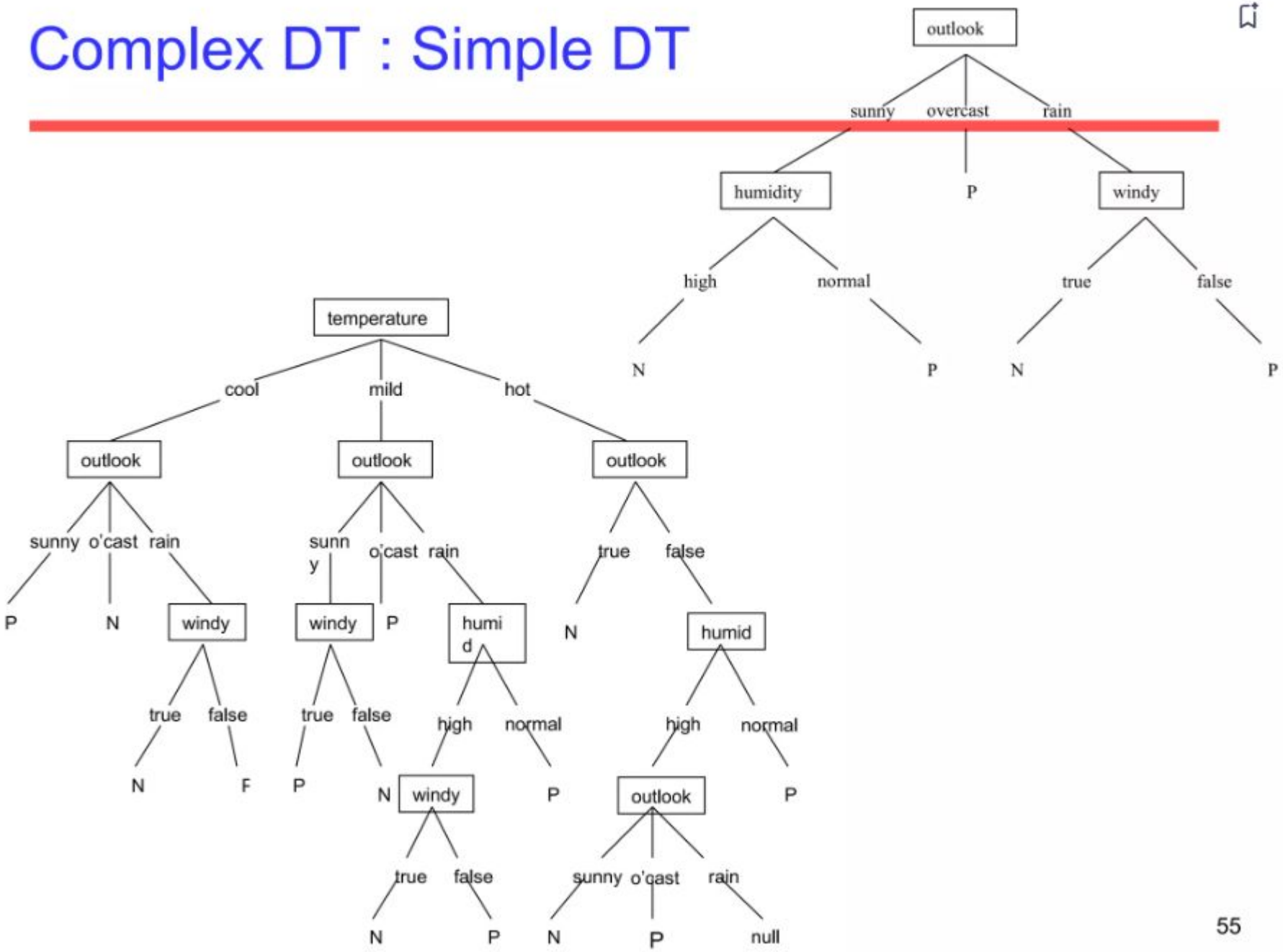
- All attributes are assumed continuous-valued

- Can be modified for categorical attributes

# Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - Information gain:
    - biased towards multivalued attributes
  - Gain ratio:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - Gini index:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

# Complex DT : Simple DT



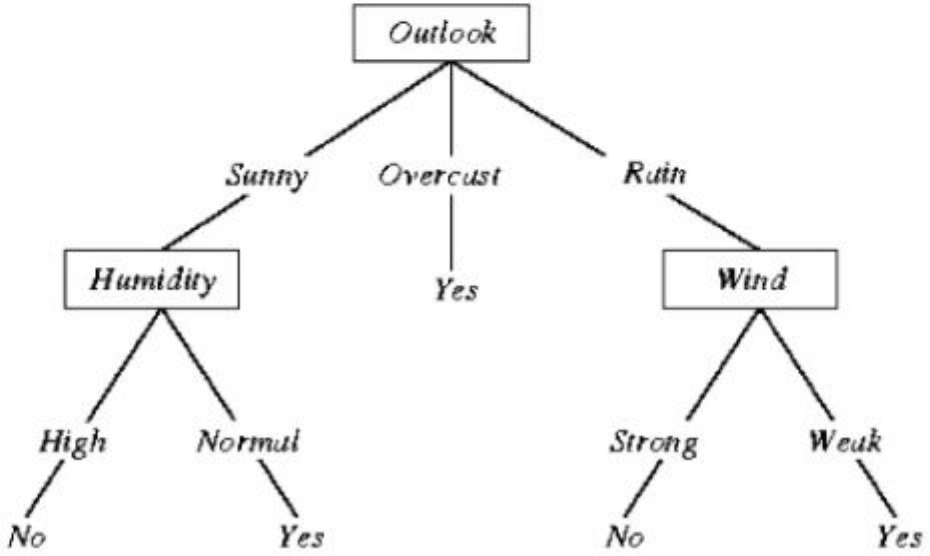
# Issues in Decision Tree Learning

## Overfitting in Decision Trees

Consider adding noisy training example #15:

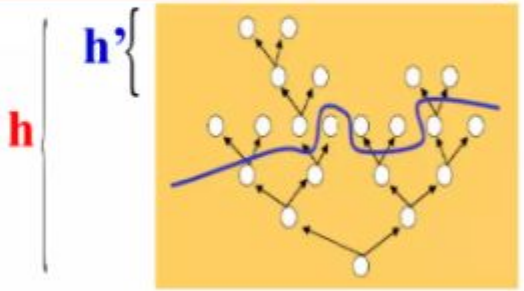
*Sunny, Hot, Normal, Strong, PlayTennis = No*

What effect on earlier tree?



# Issues in Decision Tree Learning

## Overfitting in Decision Trees




Consider error of hypothesis $h$ over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

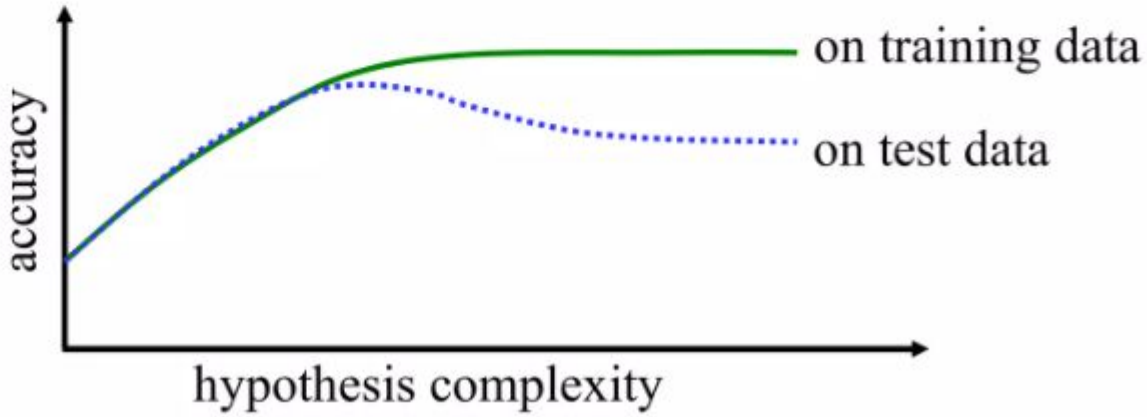
$$error_{train}(h) < error_{train}(h')$$

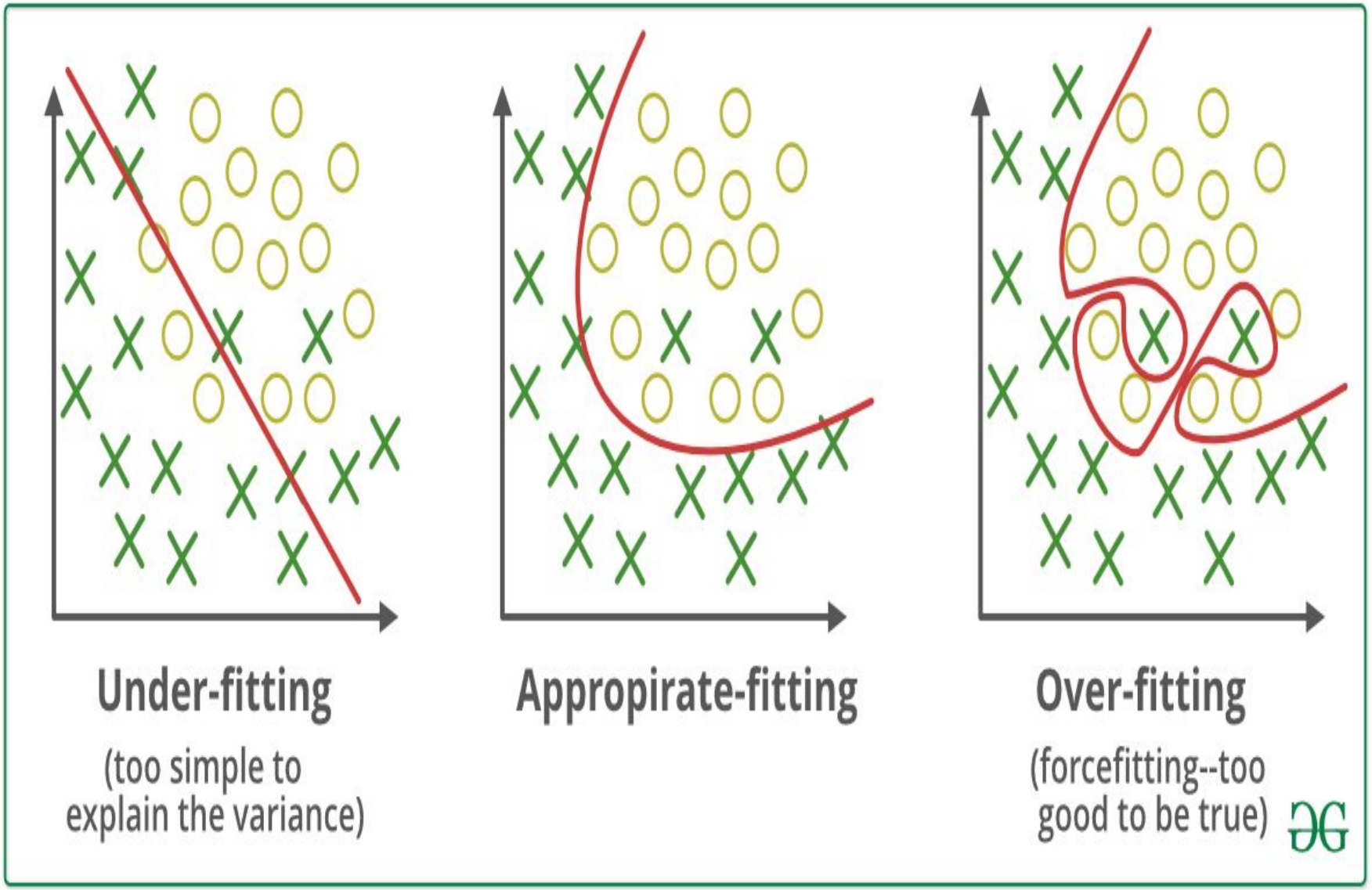
and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

# Overfitting

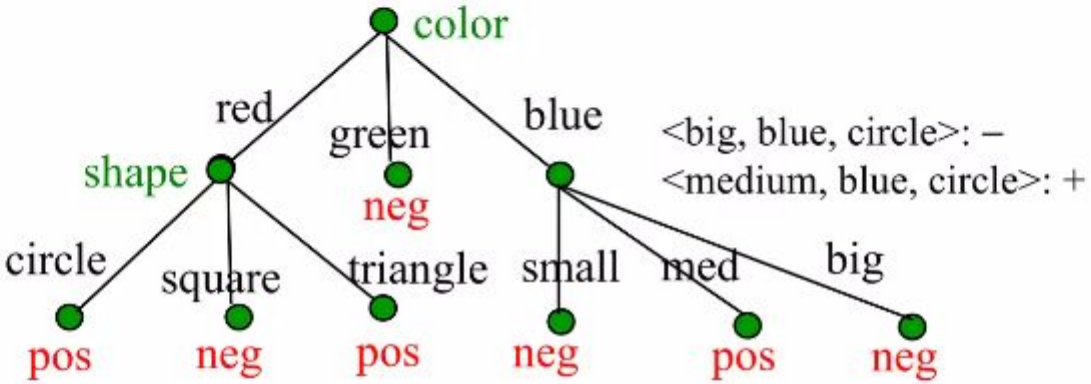
- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization to unseen data.
  - There may be noise in the training data that the tree is erroneously fitting.
  - The algorithm may be making poor decisions towards the leaves of the tree that are based on very little data and may not reflect reliable trends.




Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

# Overfitting Noise in Decision Trees

- Category or feature noise can easily cause overfitting.
  - Add noisy instance <medium, blue, circle>: pos (but really neg)



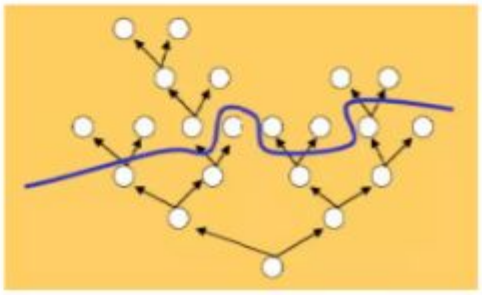

- Noise can also cause different instances of the same feature vector to have different classes. Impossible to fit this data and must label leaf with the majority class.
  - <big, red, circle>: neg (but really pos)
- Conflicting examples can also arise if the features are incomplete and inadequate to determine the class or if the target concept is non-deterministic.

# Overfitting Prevention (Pruning) Methods

- Two basic approaches for decision trees
  - Prepruning: Stop growing tree as some point during top-down construction when there is no longer sufficient data to make reliable decisions.
  - Postpruning: Grow the full tree, then remove subtrees that do not have sufficient evidence.
- Label leaf resulting from pruning with the majority class of the remaining data, or a class probability distribution.
- Method for determining which subtrees to prune:
  - Cross-validation: Reserve some training data as a hold-out set (***validation set***) to evaluate utility of subtrees.
  - Statistical test: Use a statistical test on the training data to determine if any observed regularity can be dismisses as likely due to random chance.
  - Minimum description length (MDL): Determine if the additional complexity of the hypothesis is less complex than just explicitly remembering any exceptions resulting from pruning.

# Pruning

- Goal: Prevent overfitting to noise in the data
- Two strategies for "pruning" the decision tree:
  - (Stop earlier / Forward pruning): Stop growing the tree earlier – extra stopping conditions, e.g.
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
    - Stop if number of instances < some user-specified threshold
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or Gain).
  - (Post-pruning): Allow overfit and then post-prune the tree.
    - Estimation of errors and tree size to decide which subtree should be pruned.
- Postpruning preferred in practice—prepruning can "stop too early"



# Early stopping

- Pre-pruning may stop the growth process prematurely: *early stopping*
- But: XOR-type problems rare in practice
- And: pre-pruning faster than post-pruning

# Issues in Decision Tree Learning

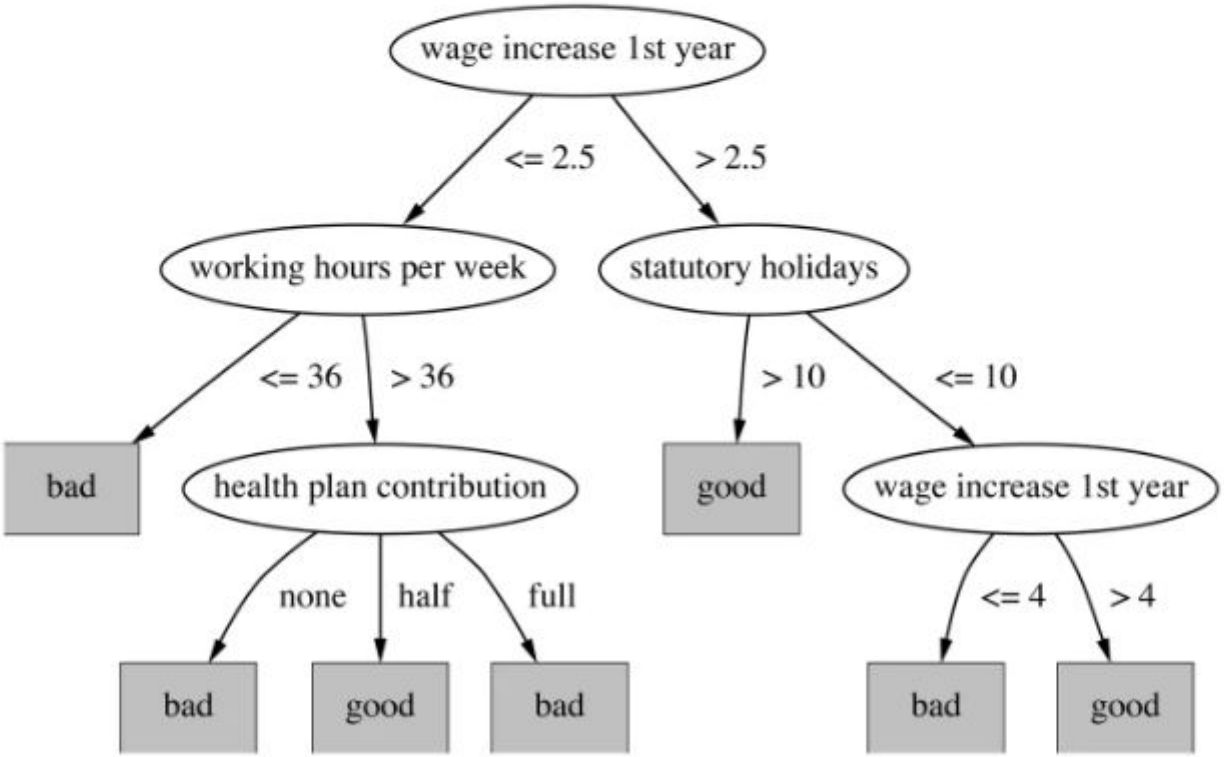
- Weakness of decision trees
  - Not Always sufficient to learn complex concepts (e.g., weighted evaluation function)
  - Can be hard to understand. Real problems can produce deep trees with a large branching factor
  - Some problems with continuously-valued attributes or classes may not be easily discretized
  - Methods for handling missing attribute values are somewhat clumsy

# Post-pruning

- First, build full tree
- Then, prune it
  - Fully-grown tree shows all attribute interactions
- Problem: some subtrees might be due to chance effects
- Two pruning operations:
  - *Subtree replacement*
  - *Subtree raising*
- Possible strategies:
  - error estimation
  - significance testing
  - MDL principle

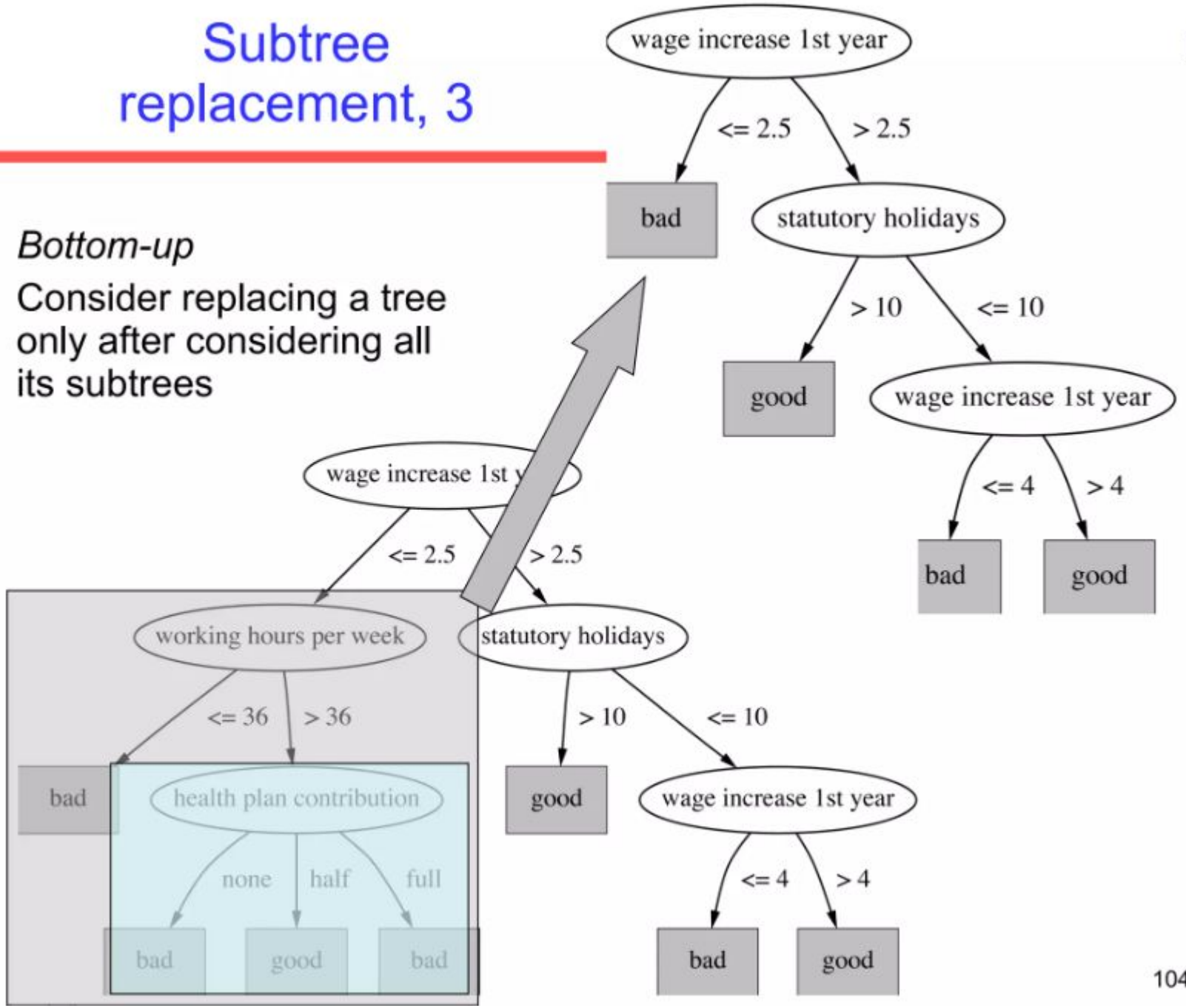
# Post-pruning: Subtree replacement

What subtree can we replace?



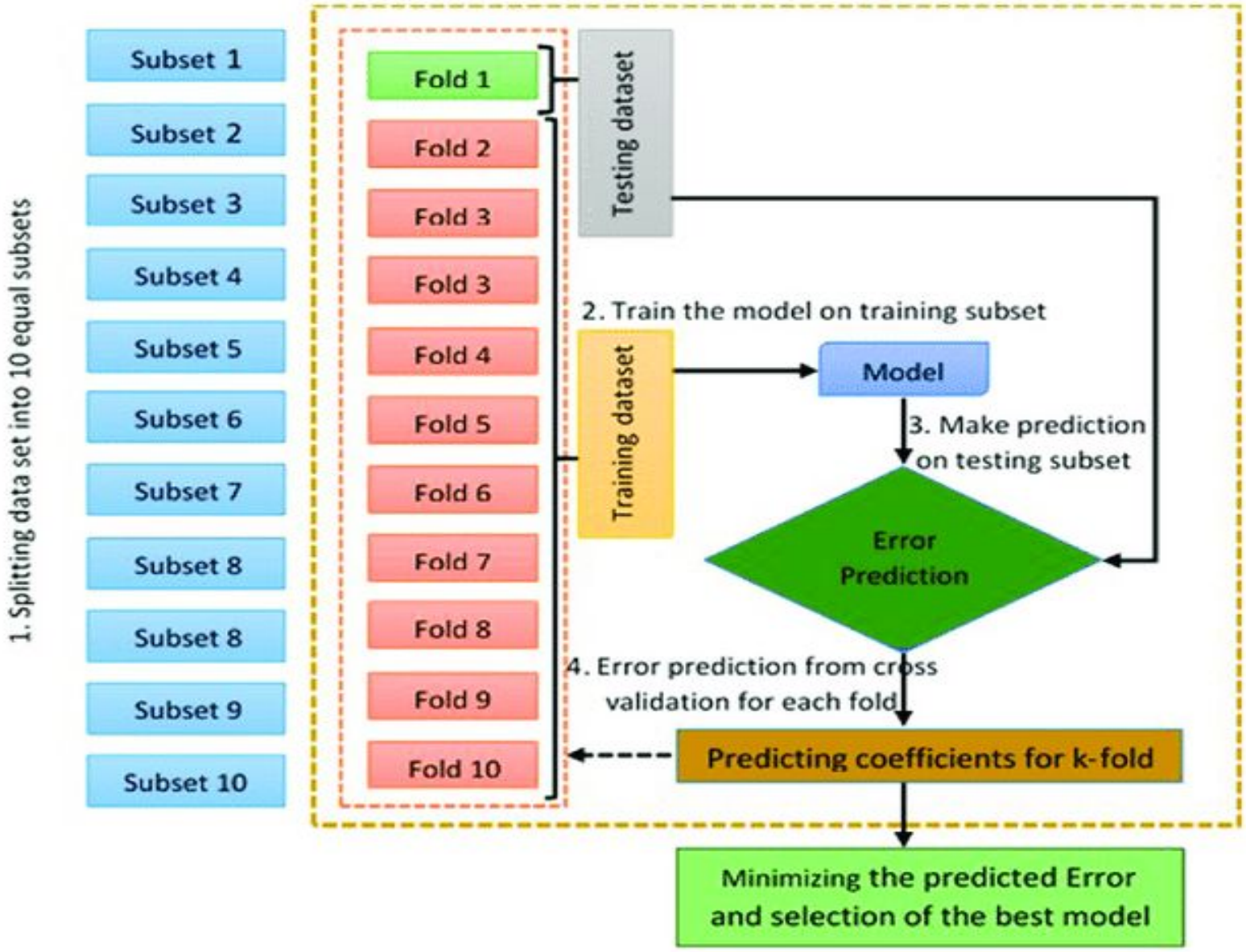
# Subtree replacement, 3

- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees



# Cross Validation in Machine Learning

- Cross validation is the **use of various techniques to evaluate a machine learning model's ability** to generalize when processing new and unseen datasets.
- Cross validation requires the **partitioning of a dataset into training and validation** datasets.
- It is a technique **mostly used with predictive machine learning models**, which use an understanding of input and output data to predict trends with new and unseen data.
- Cross validation in machine learning **compares a model's accuracy with training data against testing or validation datasets**.
- The training dataset is used to train the machine learning model, then the testing dataset is used to evaluate the effectiveness of the model with unseen data.


1. Splitting data set into 10 equal subsets
Subset 1
Subset 2
Subset 3
Subset 4
Subset 5
Subset 6
Subset 7
Subset 8
Subset 8
Subset 9
Subset 10
Fold 1
Fold 2
Fold 3
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10
Testing dataset
Training dataset
2. Train the model on training subset
Model
3. Make prediction on testing subset
Error Prediction
4. Error prediction from cross validation for each fold
Predicting coefficients for k-fold
Minimizing the predicted Error and selection of the best model

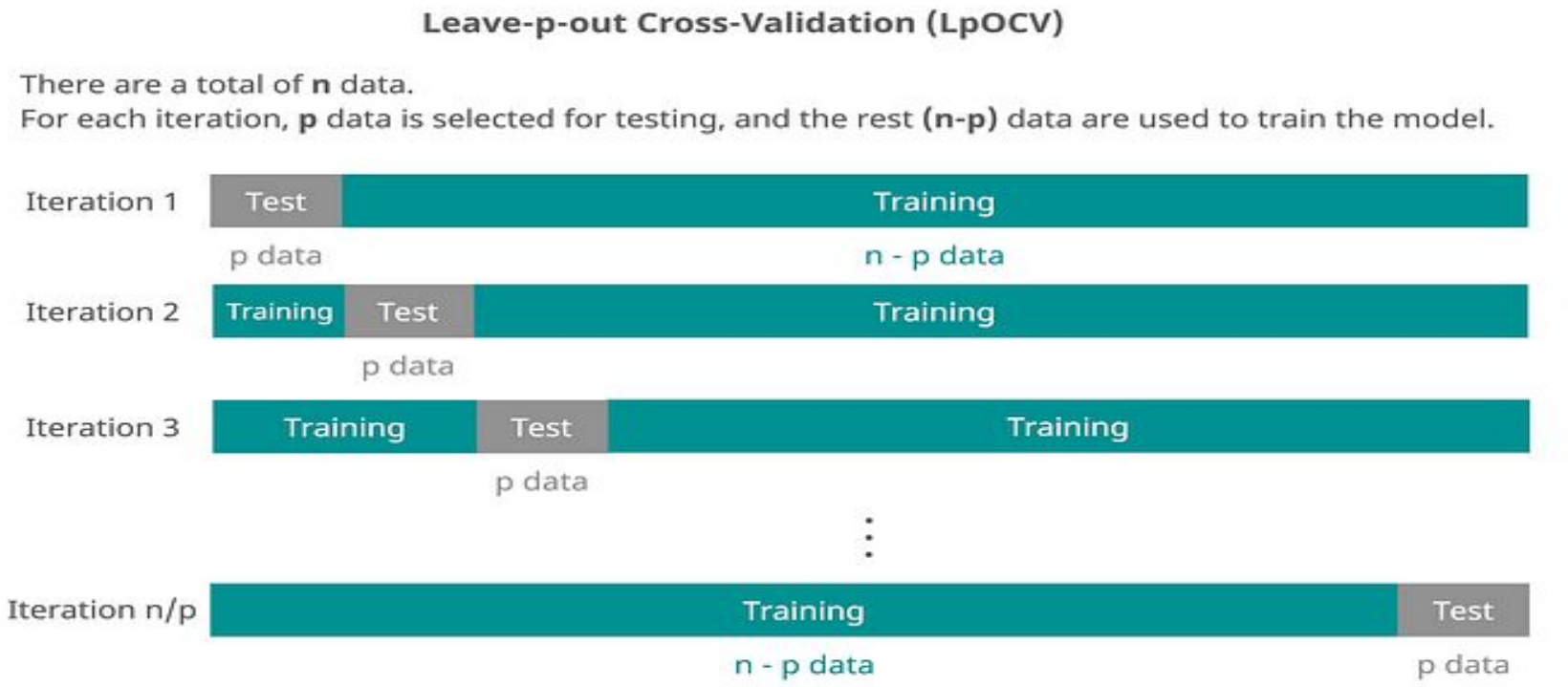
# Methods used for Cross-Validation

## ❑Validation Set Approach

- training set (50%) and test or validation set (50%) in the validation set approach. It also tends to give the under fitted model.

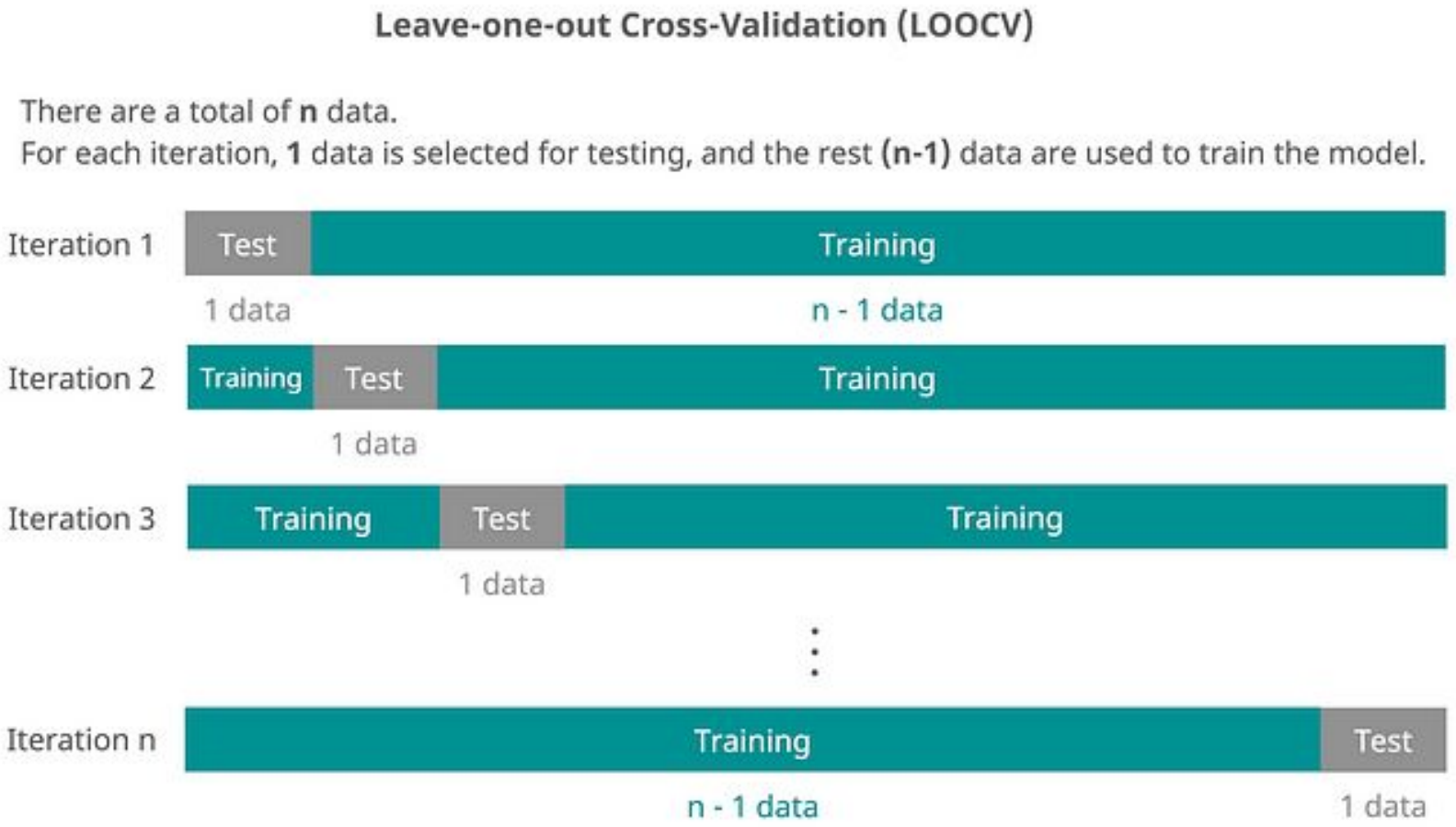
## ❑Leave-P-out cross-validation

- total n data points in the original input dataset, then n-p data points will be used as the training dataset and the p data points as the validation set.



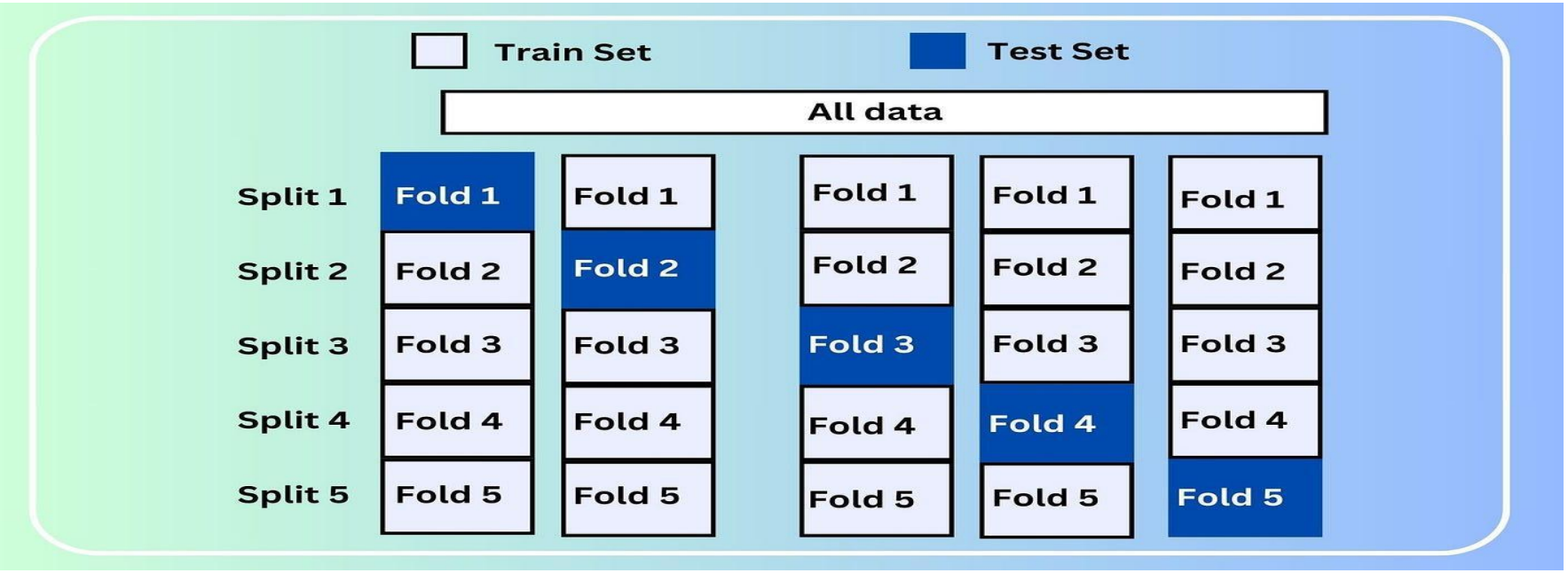
## Leave one out cross-validation

instead of p, we need to take 1 dataset out of training



## ❑K-fold cross-validation

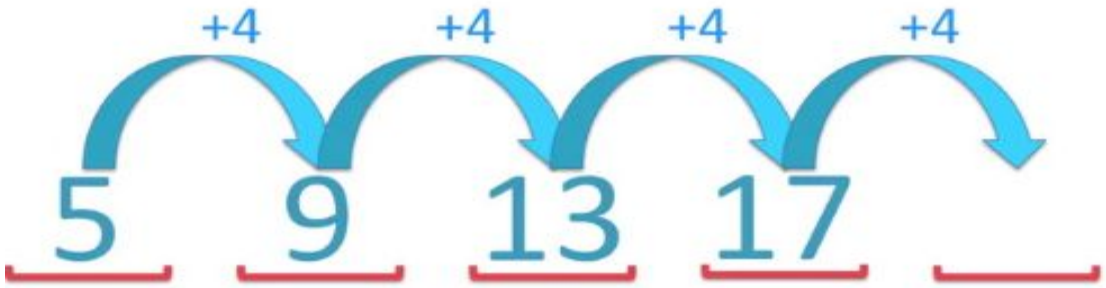
- K-fold cross-validation approach divides the input dataset into K groups of samples of equal sizes. These samples are called **folds**. k-1 folds are training, and the rest of the folds are used for the test set.




## ❑Stratified k-fold cross-validation

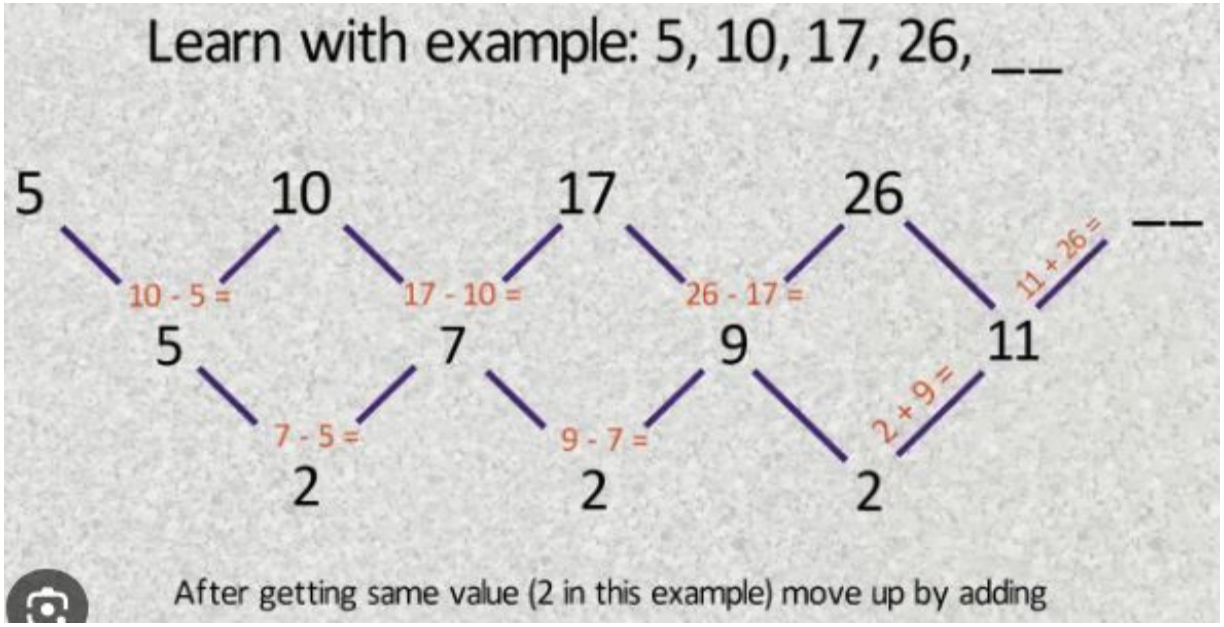
- To ensure that each fold of the cross-validation process maintains the same class distribution as the entire dataset.

**"TRAINING SET AND TEST SET ON A RATIO OF 70:30, 80:20"**




What is the next term in the sequence?

?



# Linear Regression

**Linear regression** is also a type of machine-learning algorithm more specifically a **supervised machine-learning algorithm** that learns from the labeled datasets and maps the data points to the most optimized linear functions.

which can be used for prediction on new datasets.

**Regression**: It predicts the continuous output variables based on the independent input variable.

Ex: the prediction of house prices based on different parameters like house age, distance from the main road, location, area, etc.

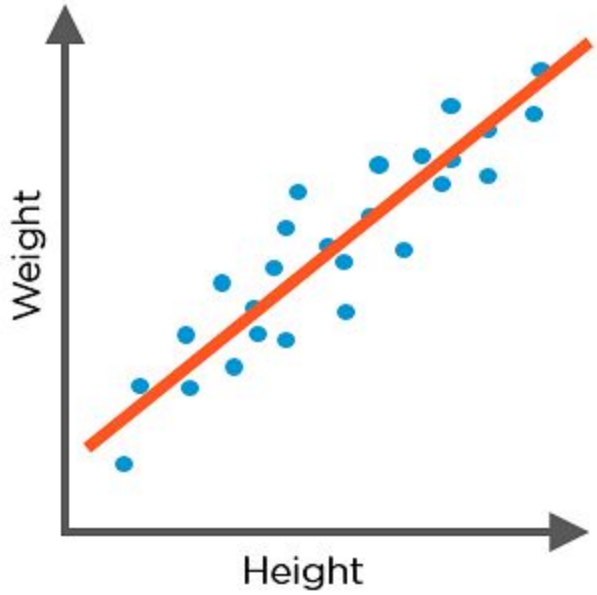
Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation to observed data.

# What is Linear Regression?

Linear regression is a form of statistical analysis that shows the relationship between two or more continuous variables.

It creates a predictive model using relevant data to show trends. Analysts typically use the “least square method” to create the model.

There are other methods, but the least square method is the most commonly used.



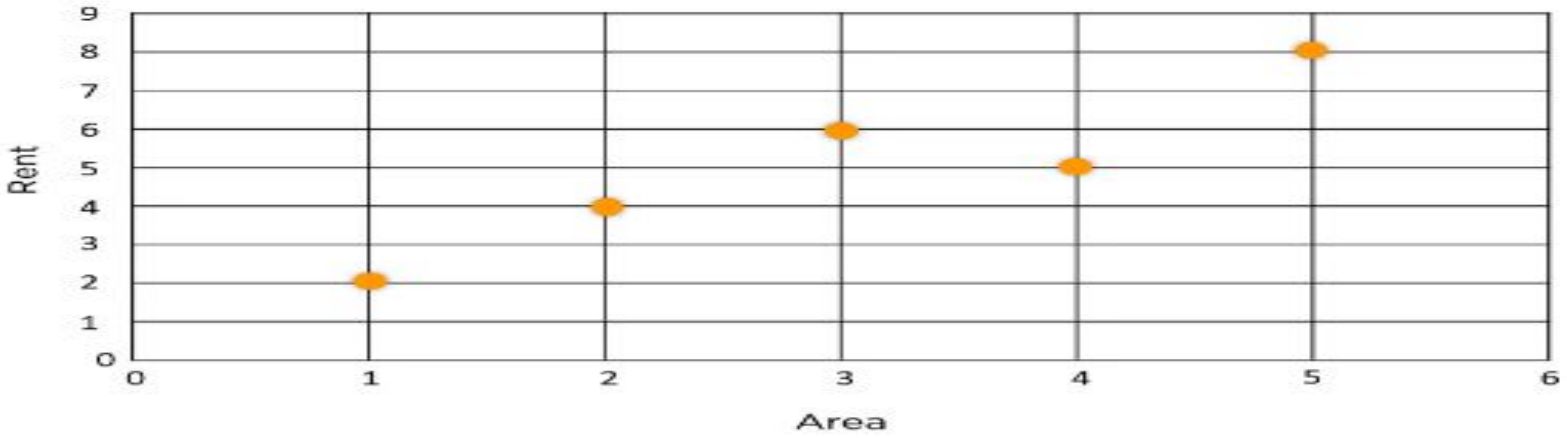

The red line is the linear regression that shows the height of a person is positively related to its weight.

# Working of Linear Regression

The dataset is contains two fields, Area and Rent, and is used to predict the house's rent based on the area

| Area | Rent |
|---|---|
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 4 | 5 |
| 5 | 8 |

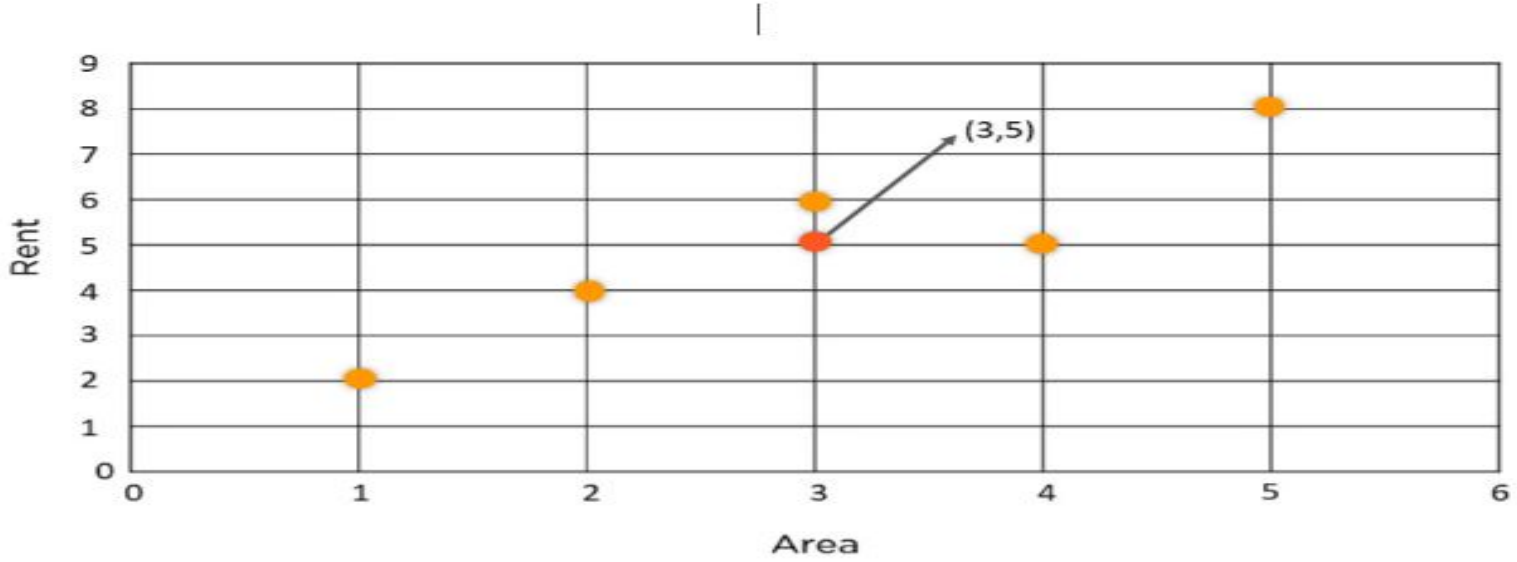
1. With the available data, we plot a graph with Area in the X-axis and Rent on Y-axis.

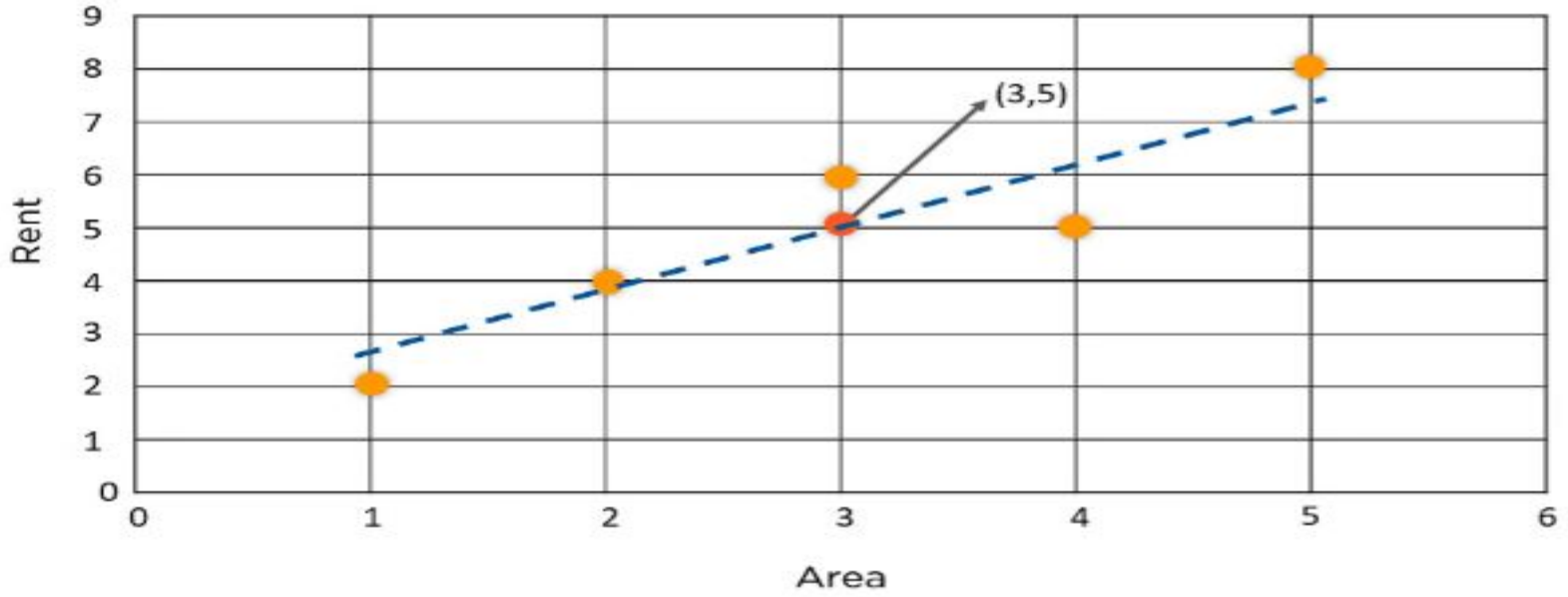



2. Next, we find the mean of Area and Rent.

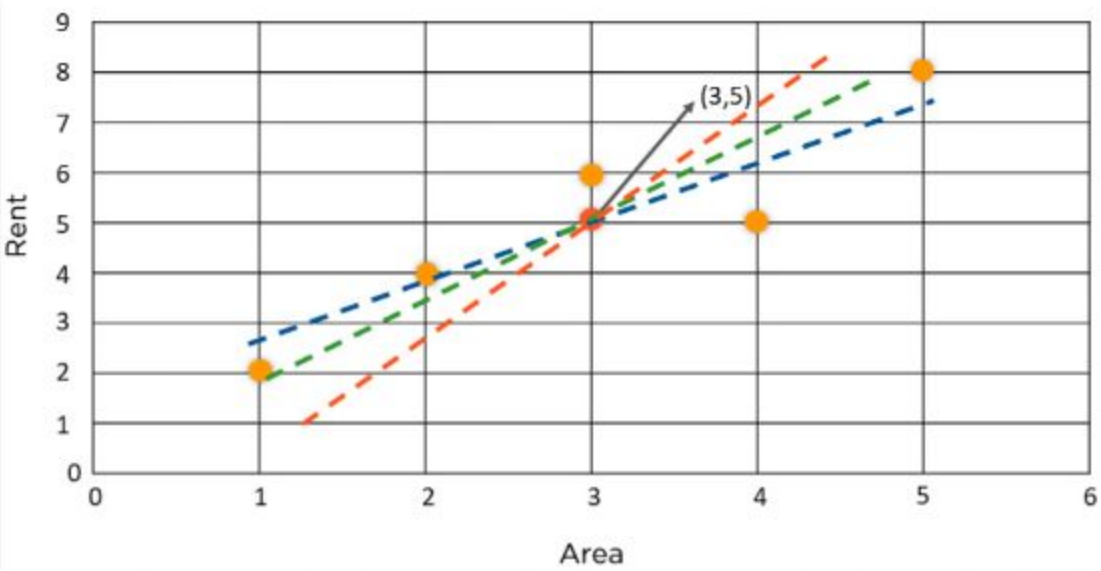
| | Area | Rent |
|---|---|---|
| | 1 | 2 |
| | 2 | 4 |
| | 3 | 6 |
| | 4 | 5 |
| | 5 | 8 |
| Mean | 3 | 5 |

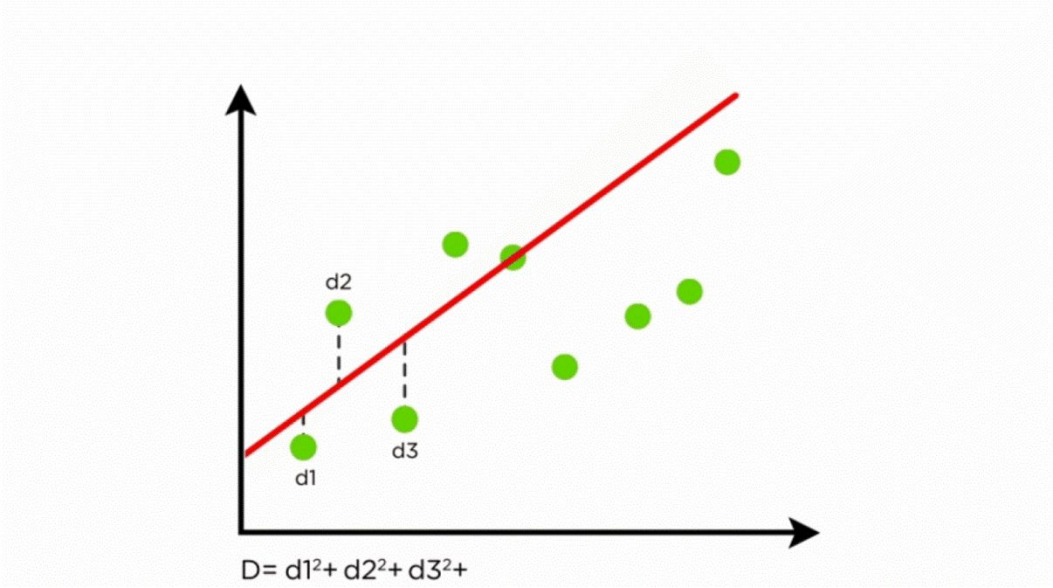
**3. We then plot the mean on the graph and draw a line of best fit that passes through the mean..**



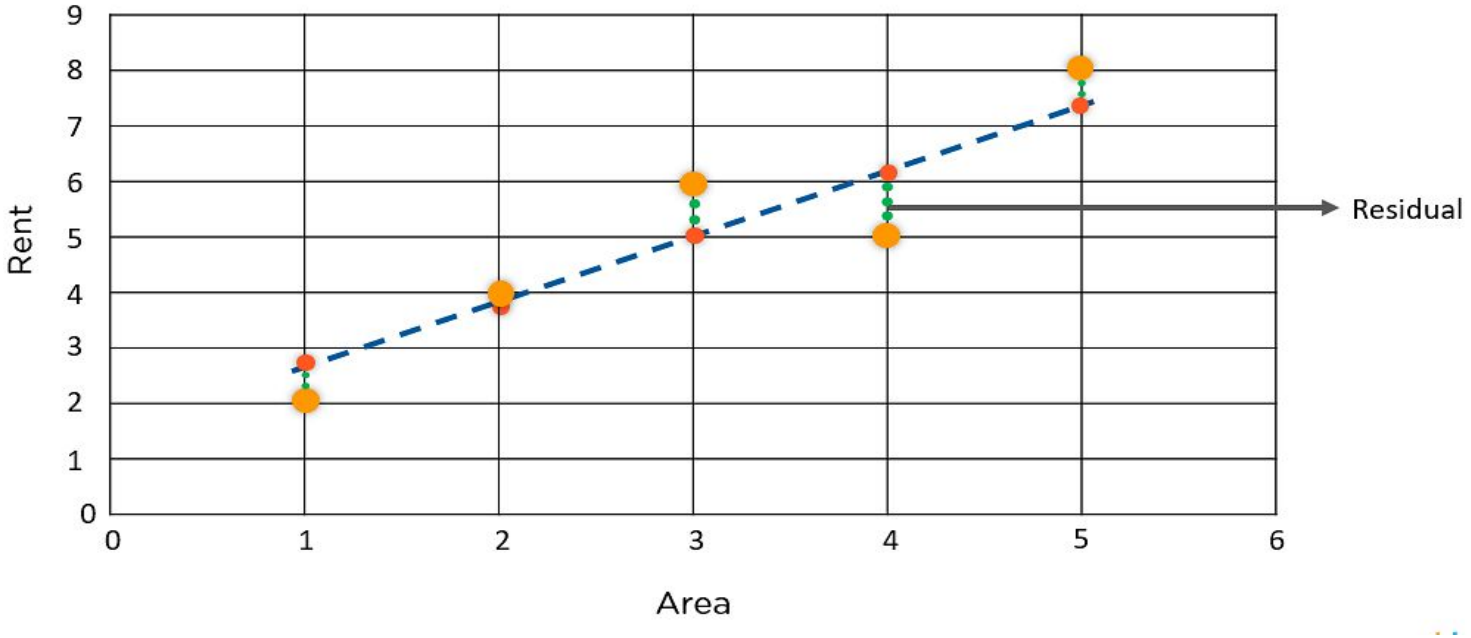



But we encounter a problem. As you can see below, multiple lines can be drawn through the mean:

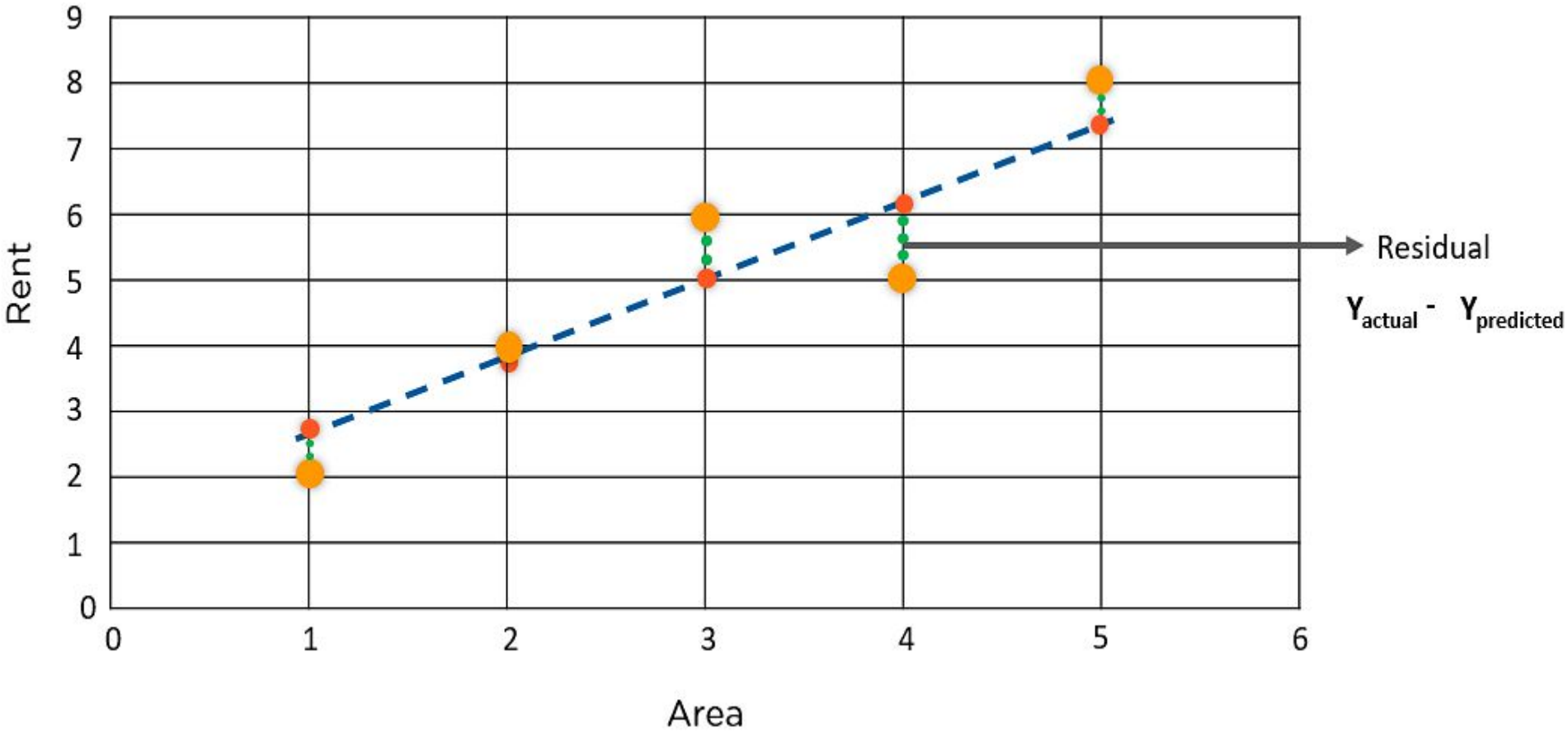


The least-square distance is found by adding the square of the residuals



**We now arrive at the relation that, Residual is the distance between Y-actual and Y-pred.**



**The value of m & c for the best fit line, y = mx+ c can be calculated using these formulas:**

$$c = \frac{((\Sigma(y)*\Sigma(x^2))-(\Sigma(x)*\Sigma(x*y)}{((n*\Sigma(x^2))-(\Sigma(x)^2)}$$

$$m = \frac{((n*\Sigma\,(x\ *y))\ -(\Sigma(x)*\Sigma(y))}{((n*\ \Sigma(x^2))-(\Sigma(x)^2)}$$

This helps us find the corresponding values:

| Rent | Area | | | |
|---|---|---|---|---|
| $x$ | $y$ | $x^2$ | $y^2$ | $x * y$ |
| 1 | 2 | 1 | 4 | 2 |
| 2 | 4 | 4 | 16 | 8 |
| 3 | 6 | 9 | 36 | 18 |
| 4 | 5 | 16 | 25 | 20 |
| 5 | 8 | 25 | 64 | 40 |
| **15** | **25** | **55** | **145** | **88** |

11. With that, we can obtain the values of m & c.

$$m = \frac{((n^{*}\Sigma\,(x\ *y))\ -(\Sigma(x)^{*}\Sigma(y))}{((n^{*}\ \Sigma(x^{2}))-(\Sigma(x)^{2})} = 1.3$$

$$c = \frac{((\Sigma(y)^{*}\Sigma(x^{2}))\text{-}(\Sigma(x)^{*}\Sigma(x^{*}y)}{((n^{*}\Sigma(x^{2}))\text{-}(\Sigma(x)^{2})} = 1.1$$

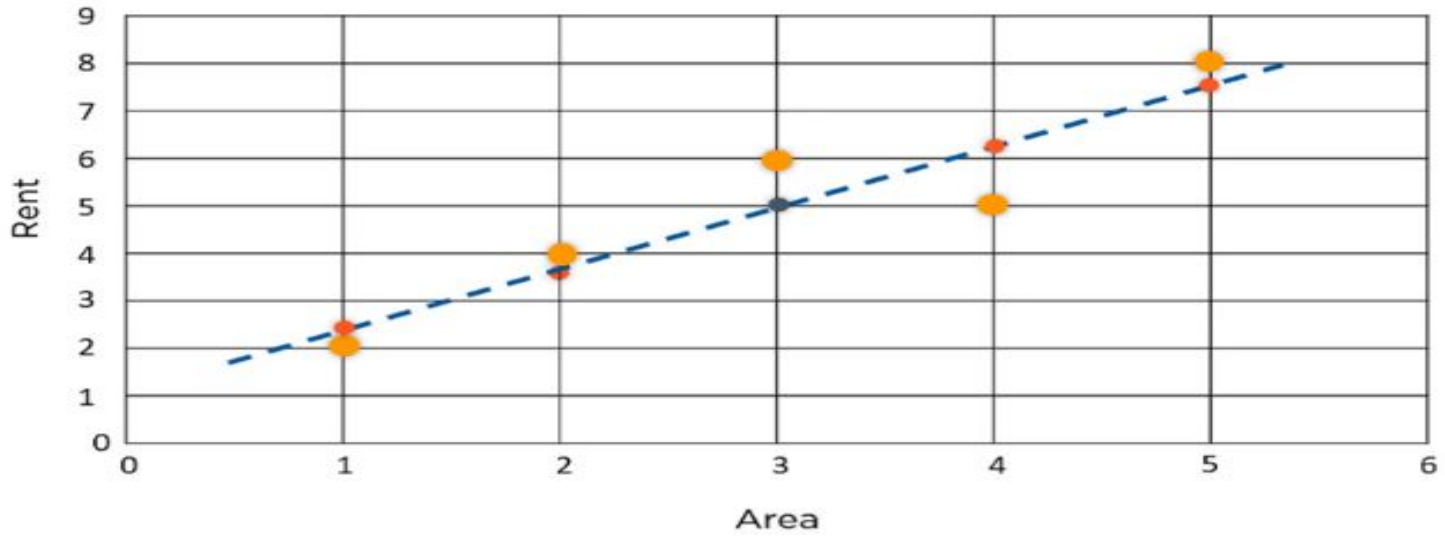
12. Now, we can find the value of Y-pred.

| x | $Y_{predicted} = m_i x_i + c$ |
|---|---|
| 1 | 1.3 * 1 + 1.1 = 2.3 |
| 2 | 1.3 * 2 + 1.1 = 3.7 |
| 3 | 1.3 * 3 + 1.1 = 5 |
| 4 | 1.3 * 4 + 1.1 = 6.3 |
| 5 | 1.3 * 5 + 1.1 = 7.6 |

13. After calculating, we find that the least square value for the below line is 3.02.

| Rent | Area | | | |
|---|---|---|---|---|
| x | y | $y_{pred}$ | $y - y_{pred}$ | $(y - y_{pred})^2$ |
| 1 | 2 | 2.3 | -0.3 | 0.09 |
| 2 | 4 | 3.71 | 0.29 | 0.0841 |
| 3 | 6 | 5 | 1 | 1 |
| 4 | 5 | 6.3 | -1.3 | 1.69 |
| 5 | 8 | 7.6 | 0.4 | 0.16 |
| **15** | **25** | **24.91** | **0.09** | **3.02** |

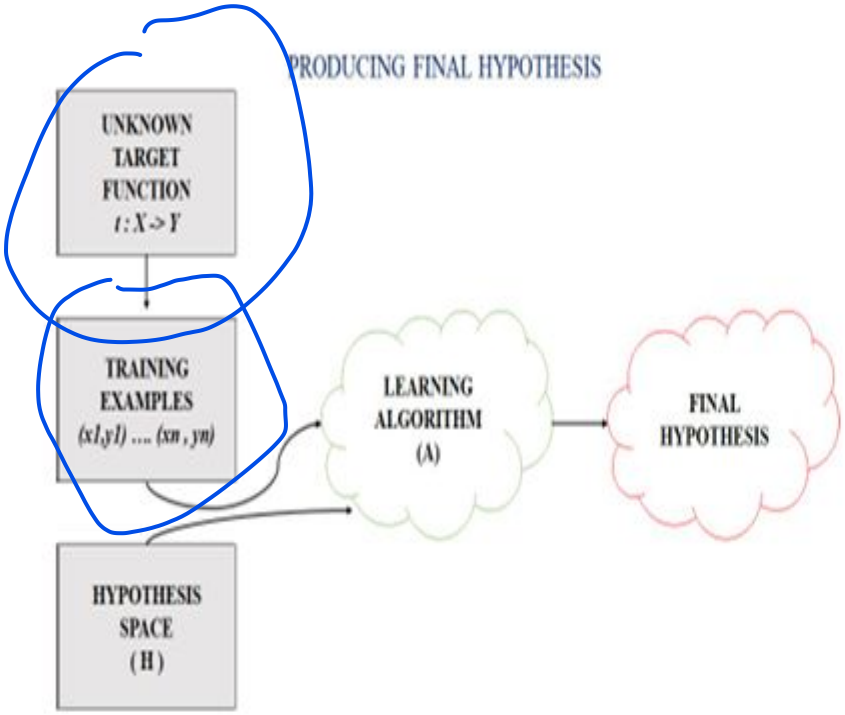
14. Finally, we are able to plot the Y-pred and this is found out to be the best fit line.



**Multiple Regression :** https://www.statology.org/multiple-linear-regression-by-hand/

Tool : https://www.statology.org/how-to-load-the-analysis-toolpak-in-excel/

# HYPOTHESIS




A hypothesis is a function that best describes the target in supervised machine learning.

The hypothesis that an algorithm would come up depends upon the data and also depends upon the restrictions and bias that we have imposed on the data.

# Hypothesis (h)

- In the realm of machine learning, a hypothesis serves as an **initial assumption when attempting to address a problem.**

- Hypothesis represents an assumption, while a model is a mathematical representation employed to test that hypothesis.

**Representing Hypotheses :**

Here, $h$ is conjunction of constraints on attributes

Each constraint can be

- a specific value (e.g., $Water = Warm$)
- don't care (e.g., "$Water$ =?")
- no value allowed (e.g.,"Water=$\emptyset$")

For example,

| Sky | AirTemp | Humid | Wind | Water | Forecst |
|---|---|---|---|---|---|
| $\langle Sunny$ | ? | ? | $Strong$ | ? | $Same\rangle$ |

# Training Examples for EnjoySport

| Sky | Temp | Humid | Wind | Water | Forecst | EnjoySpt |
|---|---|---|---|---|---|---|
| Sunny | Warm | Normal | Strong | Warm | Same | Yes |
| Sunny | Warm | High | Strong | Warm | Same | Yes |
| Rainy | Cold | High | Strong | Warm | Change | No |
| Sunny | Warm | High | Strong | Cool | Change | Yes |

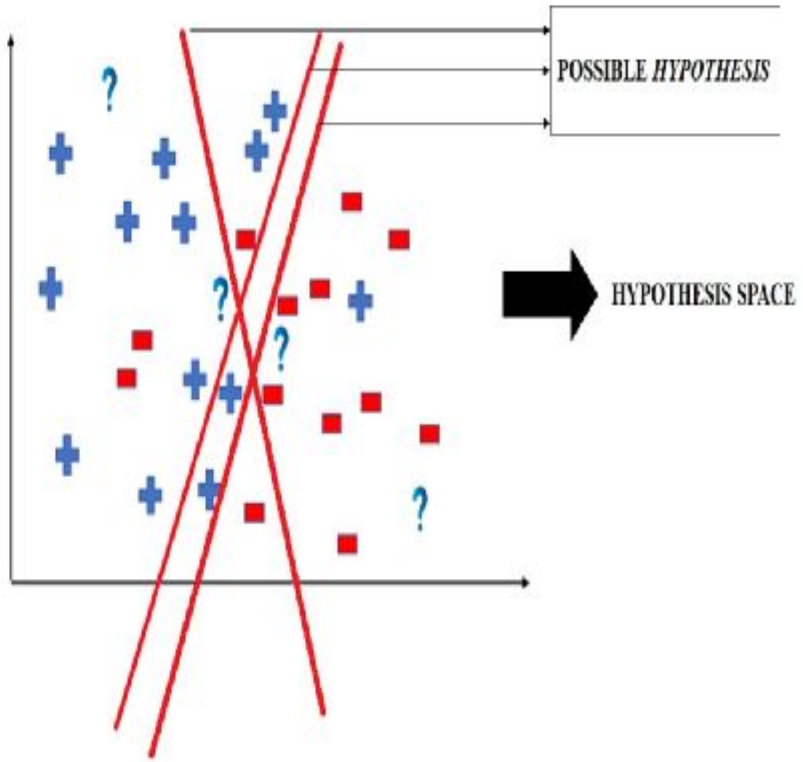
What is the general concept?

# Hypothesis Space

The way in which the coordinate would be divided depends on the data, algorithm and constraints.

All these legal possible ways in which we can divide the coordinate plane to predict the outcome of the test data composes of the Hypothesis Space (H).

Each individual possible way is known as the hypothesis (h).



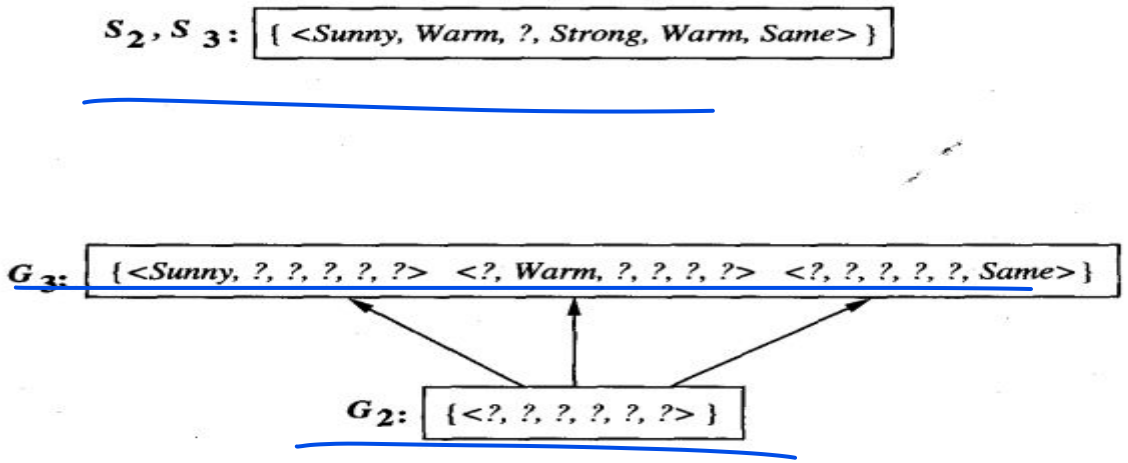
# HYPOTHESIS SPACE

Inductive bias = a +bx

Infinite Infinite

# Tasks in Hypothesis

- It is General to Specific




Training Example:

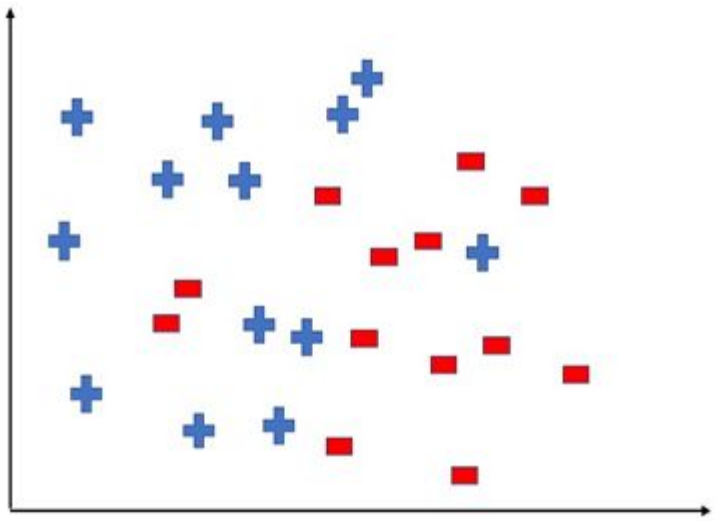
3. <Rainy, Cold, High, Strong, Warm, Change>, EnjoySport=No

**Methods:**

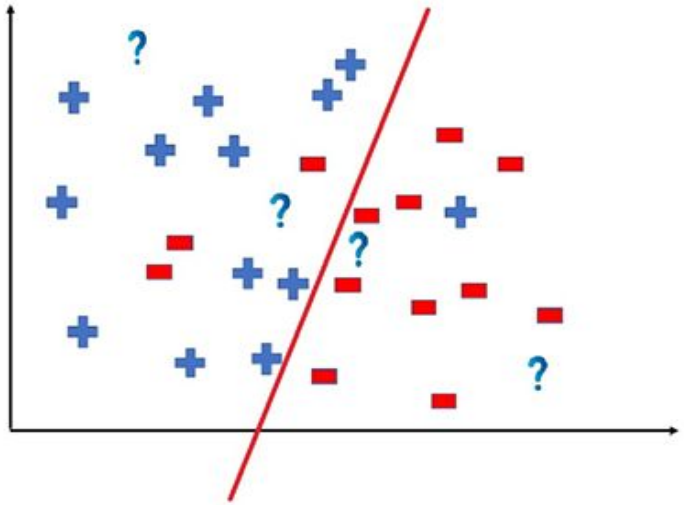
- List then eliminate
- Candidate eliminate

# CLASSIFICATION EXAMPLE

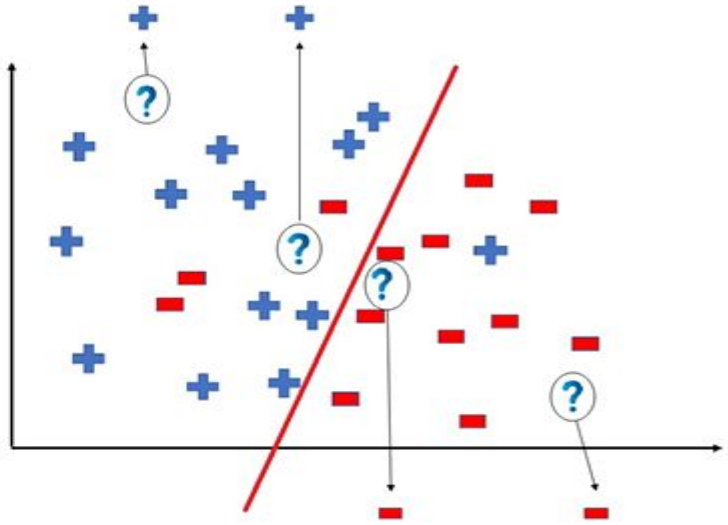
To better understand the Hypothesis Space and Hypothesis consider the following coordinate that shows the distribution of some data:



We can predict the outcomes by dividing the coordinate as shown below




So the test data would yield the following result:



# Inductive Bias

The **inductive bias** (also known as learning **bias**) of a learning algorithm is the set of assumptions that the learner uses to predict outputs.

In machine learning, one aim to construct algorithms that are able to learn to predict a certain target output.

Inductive Bias = Y=a+bx (Linear Model)

**That assumptions must be observed ?**

1. How does the size of the hypothesis space influence the number of training examples ?

2. ***Any*** hypothesis consistent with the training data. ?

**3 Consideration about the number of Samples?**

1. **A Biased Hypothesis Space** -

The obvious solution to ensuring that the target concept is included in the hypothesis space H is to make sure it can represent every possible subset of instances X

2. **An Unbiased Learner** -

It is ensuring fair and accurate results based on the data.
(If a model that makes predictions )

**3. The Futility of Bias-Free Learning –**

In machine learning, completely eliminating bias is often impossible because all models are influenced by the data they are trained on, which may inherently contain biases.

# Inductive Bias

Consider

- concept learning algorithm $L$
- instances $X$, target concept $c$
- training examples $D_c = \{\langle x, c(x)\rangle\}$
- let $L(x_i, D_c)$ denote the classification assigned to the instance $x_i$ by $L$ after training on data $D_c$.

**Definition:**

The **inductive bias** of $L$ is any minimal set of assertions $B$ such that for any target concept $c$ and corresponding training examples $D_c$

$$(\forall x_i \in X)[(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$$

where $A \vdash B$ means $A$ logically entails $B$

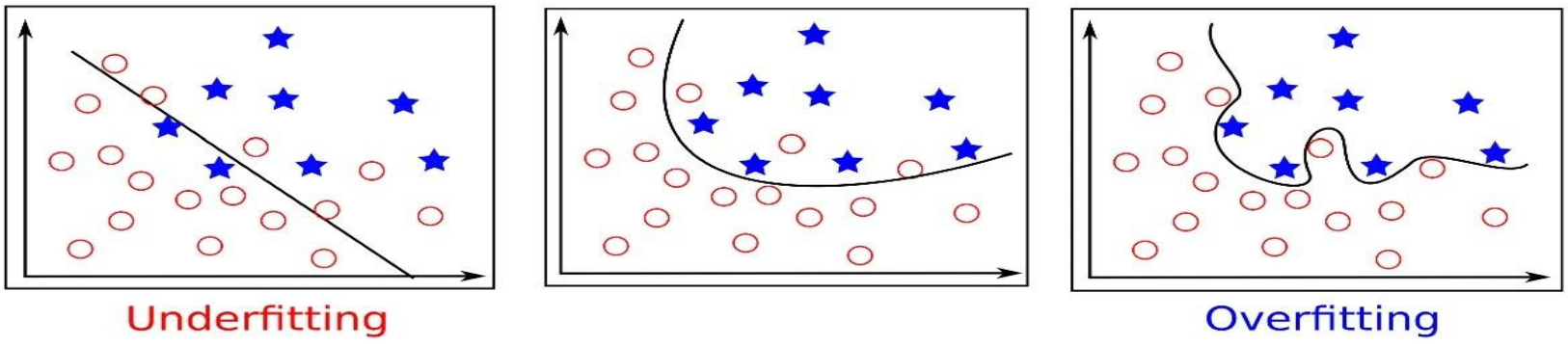
# VARIANCE

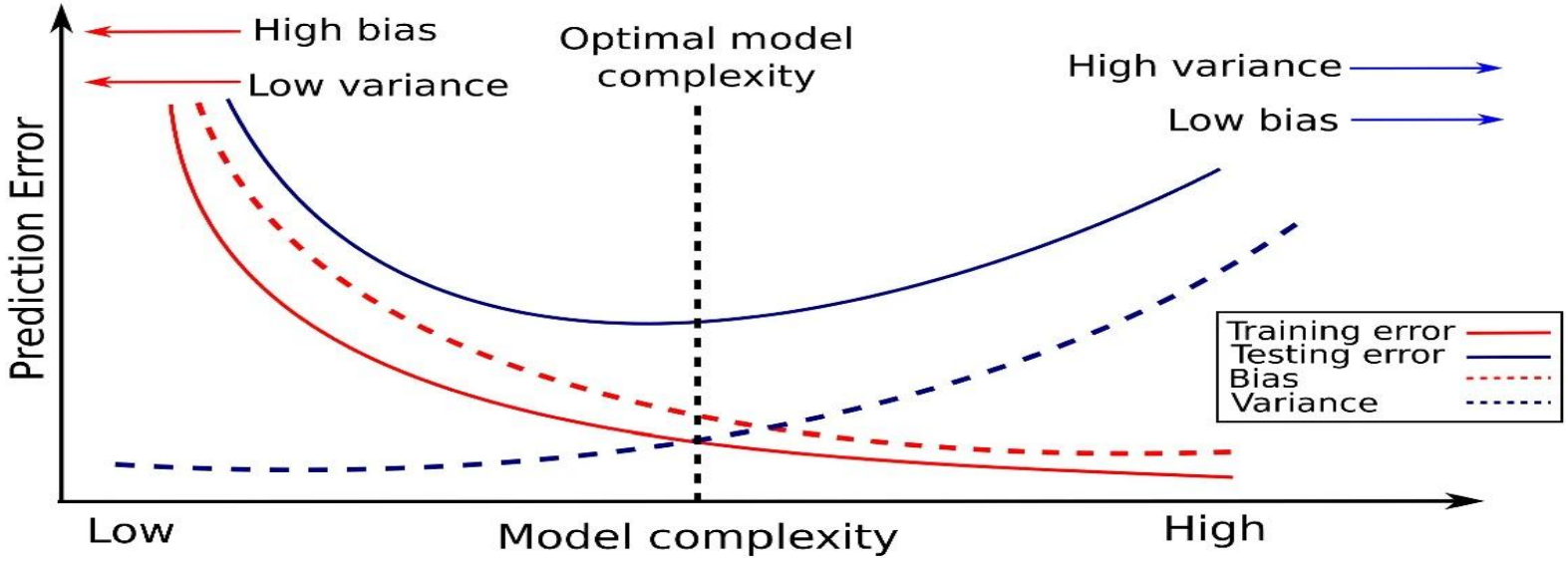
- When a model does not perform as well as it does with the trained data set, there is a possibility that the model has a variance.
- It basically tells how scattered the predicted values are from the actual values.

# Bias –Variance Trade off

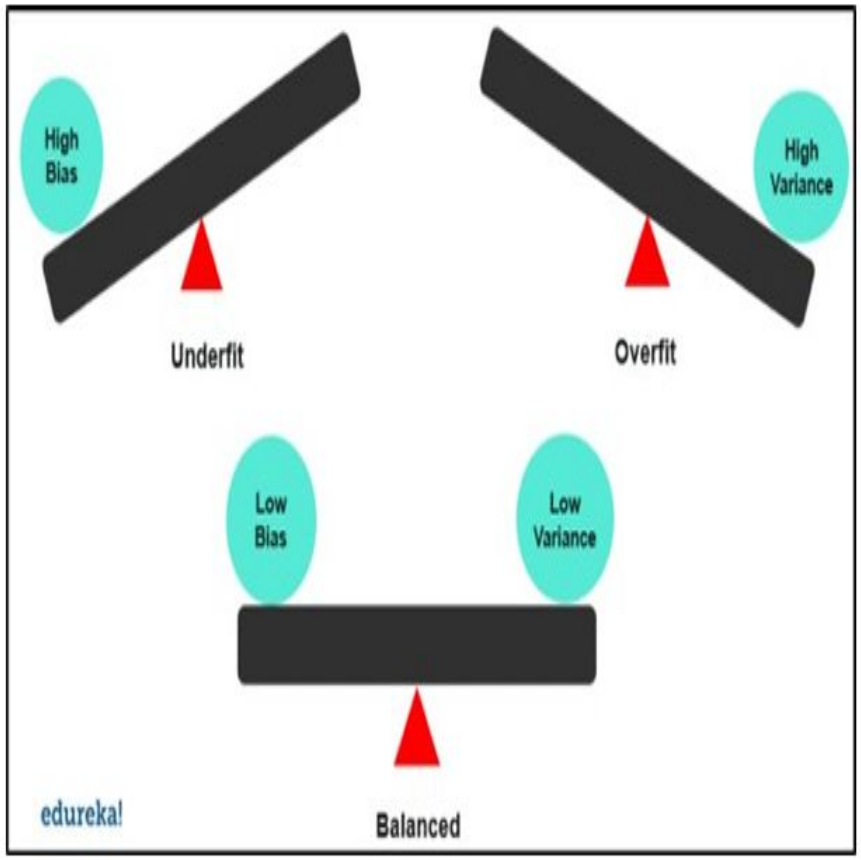
**Bias - Error in Training data**

**Variance - Error in Test data**



# IDEAL SCENARIO




**UNDERFITTING-**

**Training Data Accuracy Is low**
**and**
**Test Data Accuracy is low**

**OVERFITTING-**

**Training Data Accuracy Is high**
**and**
**Test Data Accuracy is low**

# Feature Reduction

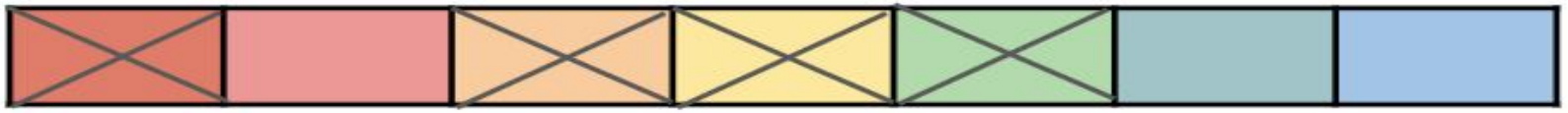
- Techniques aimed at reducing the number of features (variables, columns) in a dataset while preserving the most relevant information.
- The primary goal is to simplify the model, improve computational efficiency, and sometimes enhance model performance by removing redundant or irrelevant features.
- Feature reduction techniques include methods like
  feature selection (choosing a subset of the original features) and
  feature extraction (creating new features that capture the essence of the original ones)

# feature selection

Feature selection is based on omitting those features from the available measurements which do not contribute to class separability. In other words, redundant and irrelevant features are ignored.

All Features

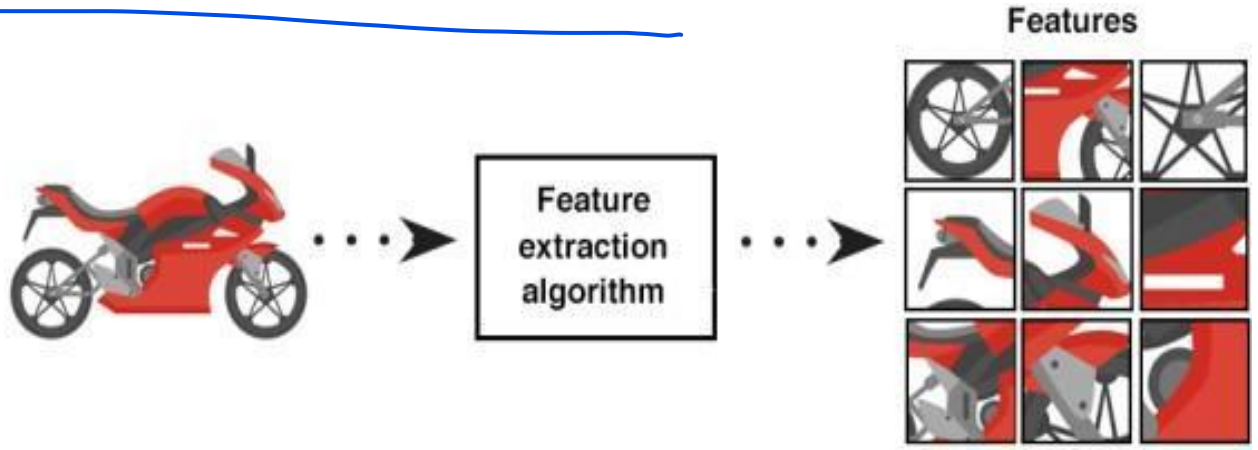
Feature Selection

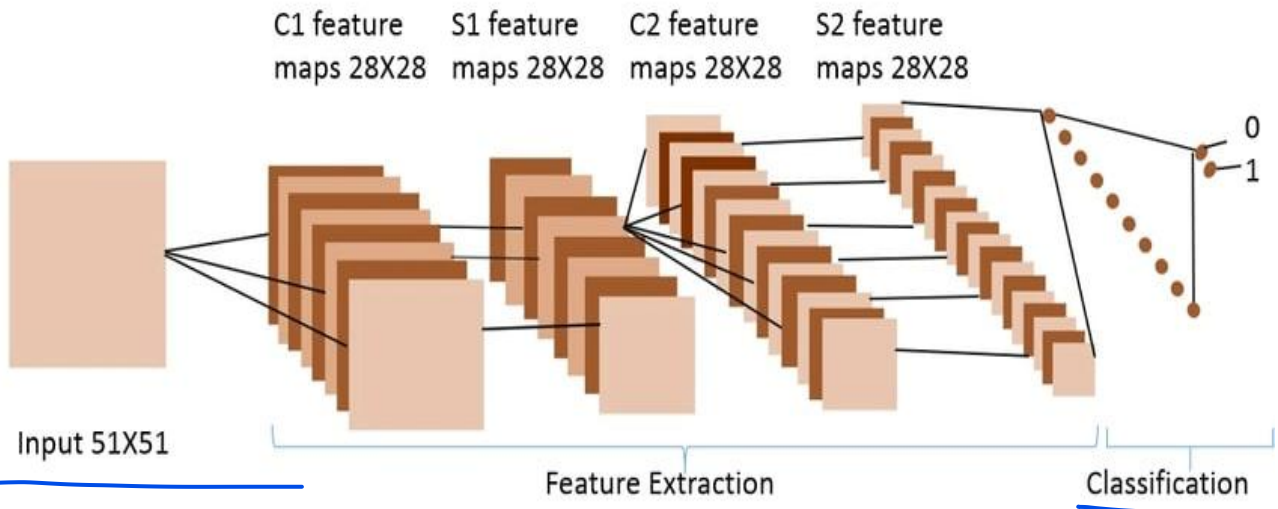


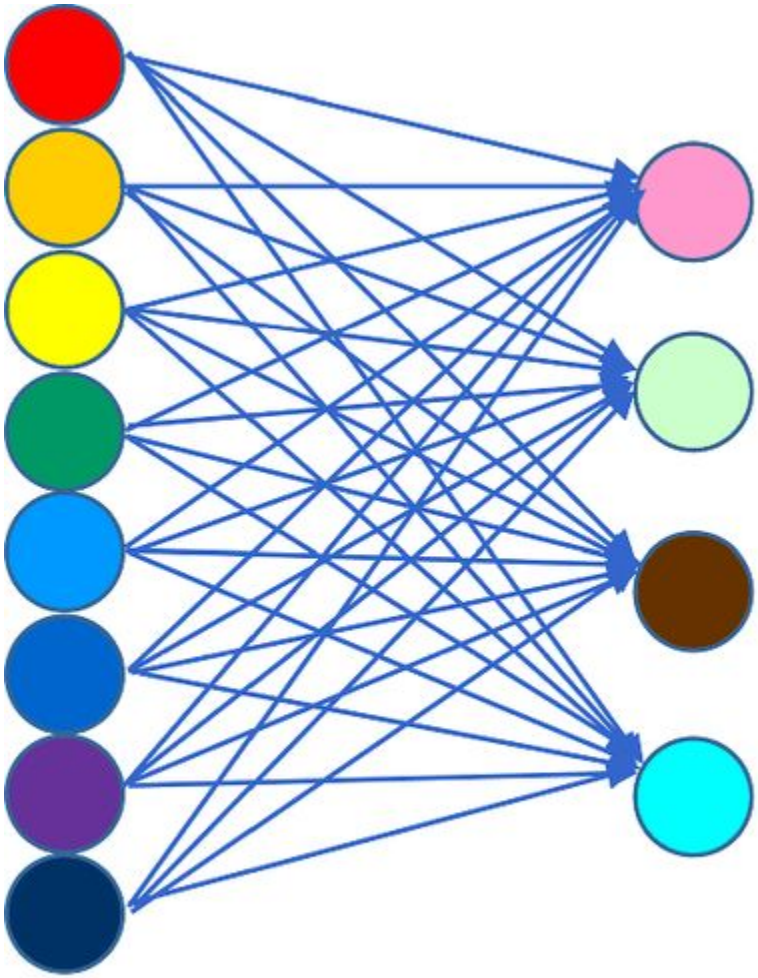
Final Features

# feature extraction

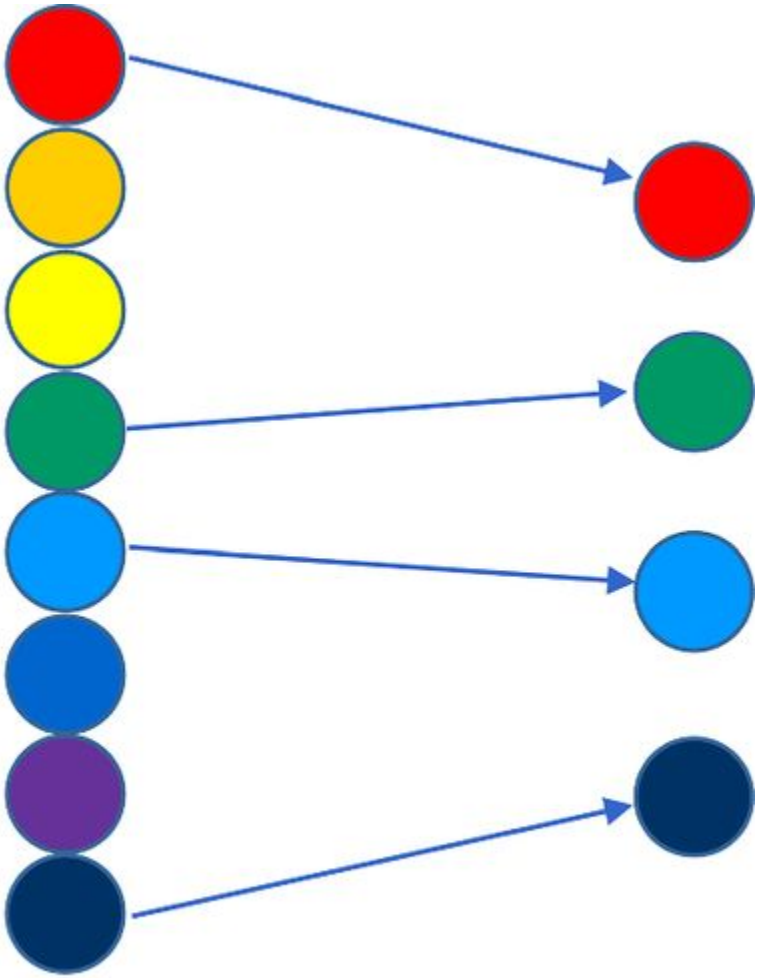
Feature extraction is the process of converting the raw data into some other data type, with which the algorithm works is called Feature Extraction. Feature extraction creates a new, smaller set of features that captures most of the useful information in the data.





feature extraction



feature selection

# Advanced Feature Selection Techniques

## 1.Supervised Techniques

### 1.1 Filter-based Approach

- Information gain
- Chi-square Test
- Fisher's Score
- Missing Value Ratio


### 1.2 Wrapper-based Approach

- Forward Selection
- Backward Selection
- Exhaustive Feature Selection
- Recursive Feature Elimination

### 1.3 Embedded Approach

- Regularization
- Random Forest Importance

## 2.Unsupervised Techniques

### 2.1 PCA

### 2.2 ICA


### 2.3 NMF

### 2.4 t-SNE

### 2.5 Autoencoder