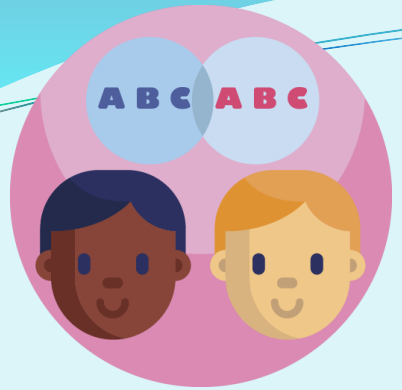


SUPERVISED LEARNING

ONLINE STUDENT TRAINING

FOR

"ARTIFICIAL INTELLIGENCE & MACHINE LEARNING"

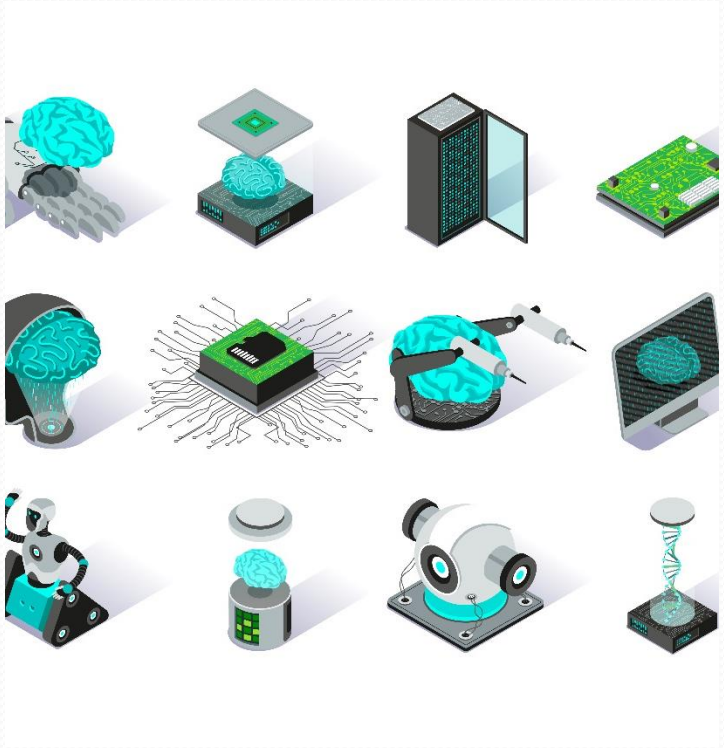
(4TH FEB, 2021 – 17TH MAR, 2021)

FACULTY TRAINER

NAW VARSHA PIPADA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ENGINEERING COLLEGE BIKANER



Contents

Supervised Learning

Classification & Regression

Basic Concepts

Model Evaluation Techniques

Model Evaluation Metrics

Supervised Learning

- Supervised machine learning algorithms are designed to learn by example. The name “supervised” learning originates from the idea that training this type of algorithm is like having a teacher supervise the whole process.
- When training a supervised learning algorithm, the training data will consist of inputs paired with the correct outputs.
- During training, the algorithm will search for patterns in the data that correlate with the desired outputs.

Categories

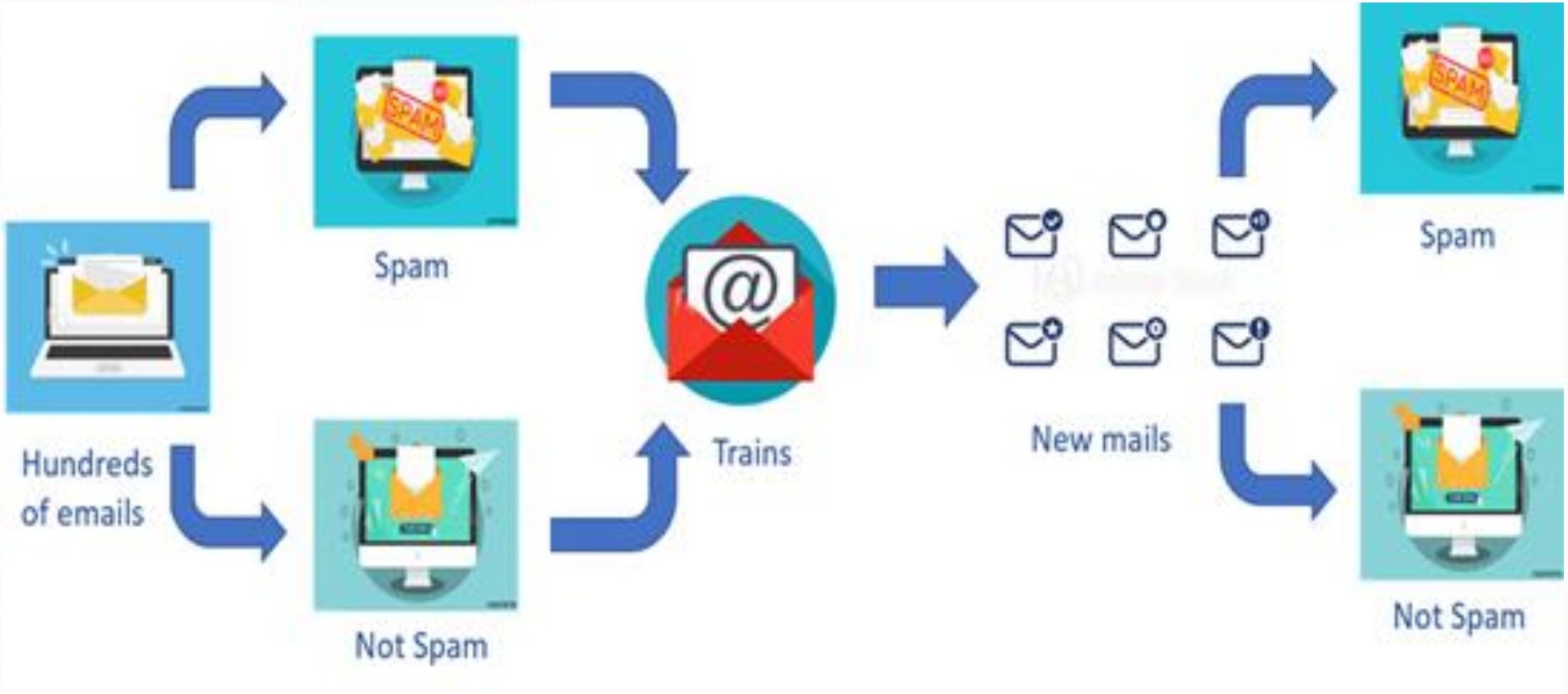
Classification

- It is used for problems where the output variable can be categorized, such as “Yes” or “No”, or “Pass” or “Fail.”
- Classification Models are used to predict the category of the data.
- Examples - spam detection, sentiment analysis, scorecard prediction of exams, etc.

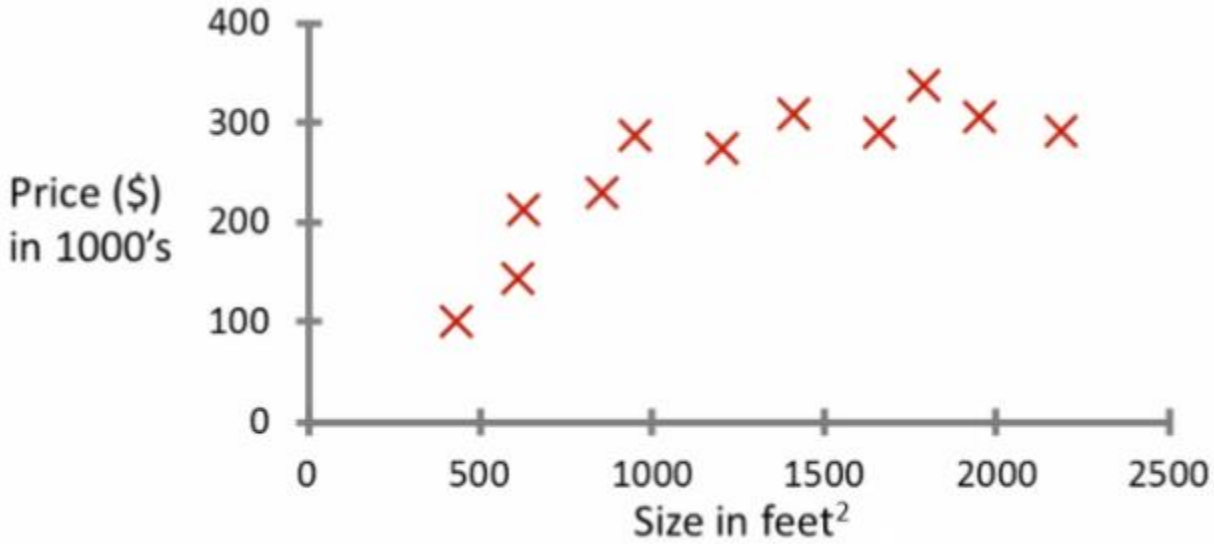
Regression

- Regression models are used for problems where the output variable is a real value such as a unique number, dollars, salary, weight or pressure, for example
- Some of the more familiar regression algorithms include linear regression, logistic regression, polynomial regression, and ridge regression

Classification



Regression


Price ($)
in 1000's
400
300
200
100
0
0
500
1000
1500
2000
2500
Size in feet²

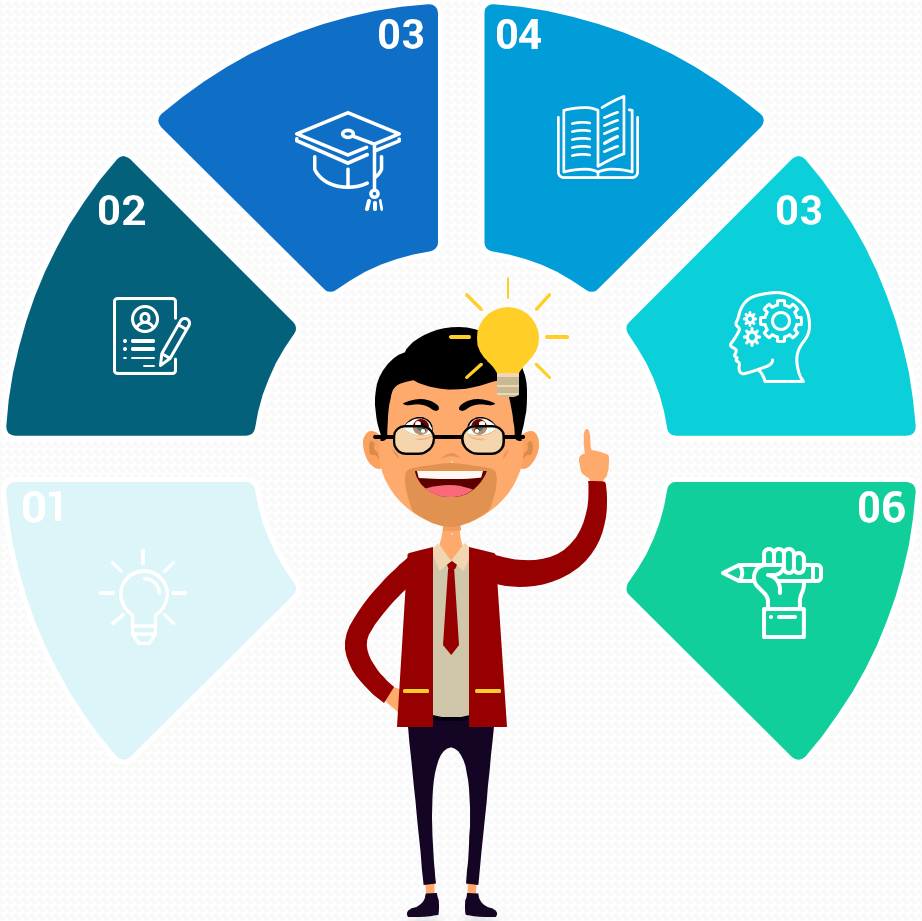
Machine Learning Process

Some Basic Concepts

Features & Target

Training Data

Testing Data



Bias-Variance Tradeoff

Model Evaluation Techniques

Model Evaluation Metrics

Terminology : Features & Its Types

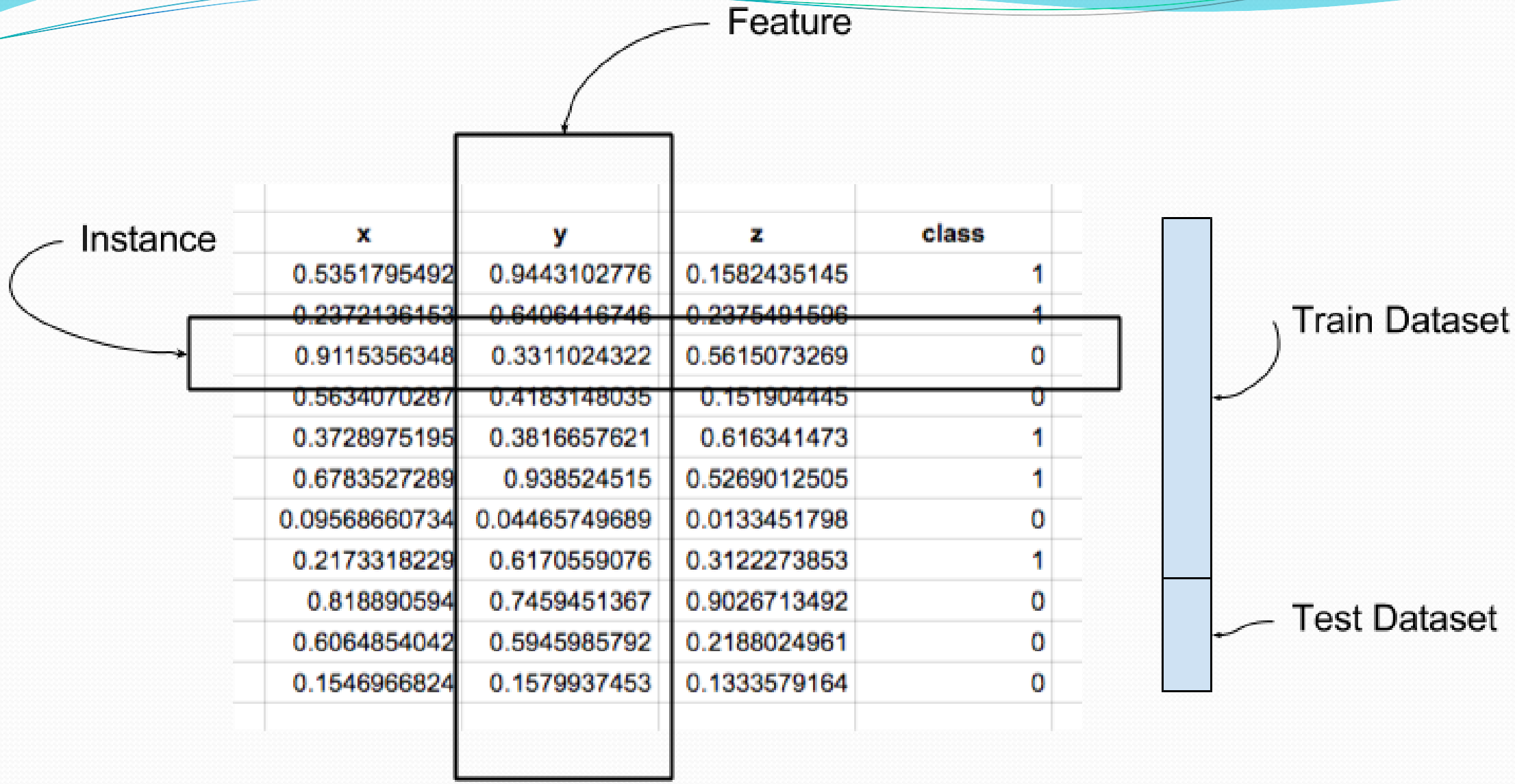
- Features are individual independent variables that act like a input to the system.
- Features are individual observations when analyzed into a set of quantifiable properties. It can be interpreted as column in dataset.
- Features are building blocks of datasets as the quality of the features in your dataset has major impact on the quality of the insights you will get while using the dataset for machine learning.
- Types
  - Categorical (e.g., Blood Group)
  - Ordinal (e.g., Shirt Size)
  - Integer (e.g., No. of words in text)
  - Real Values (e.g., Height)

Terminology : Training Data

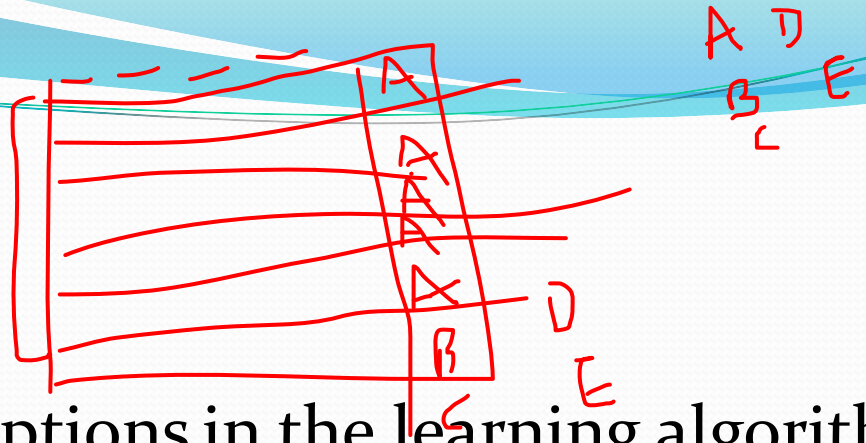
- Target
  - The **Target** is the information the machine learns to predict. In mathematical formulas, the target is usually called y or y_i for a single instance.
- Training Data
  - The observations in the training set form the experience that the algorithm uses to learn.
  - In supervised learning problems, each observation consists of an observed output variable and one or more observed input variables.

Terminology : Test Data

- Test Data
  - The test set is a set of observations used to evaluate the performance of the model using some performance metric.
  - It is important that no observations from the training set are included in the test set.
  - If the test set does contain examples from the training set, it will be difficult to assess whether the algorithm has learned to generalize from the training set or has simply memorized it.


Feature
Instance
x
y
z
class
0.5351795492
0.9443102776
0.1582435145
1
0.2372136153
0.6406416746
0.2375491596
1
0.9115356348
0.3311024322
0.5615073269
0
0.5634070287
0.4183148035
0.151904445
0
0.3728975195
0.3816657621
0.616341473
1
0.6783527289
0.938524515
0.5269012505
1
0.09568660734
0.04465749689
0.0133451798
0
0.2173318229
0.6170559076
0.3122273853
1
0.818890594
0.7459451367
0.9026713492
0
0.6064854042
0.5945985792
0.2188024961
0
0.1546966824
0.1579937453
0.1333579164
0
Train Dataset
Test Dataset

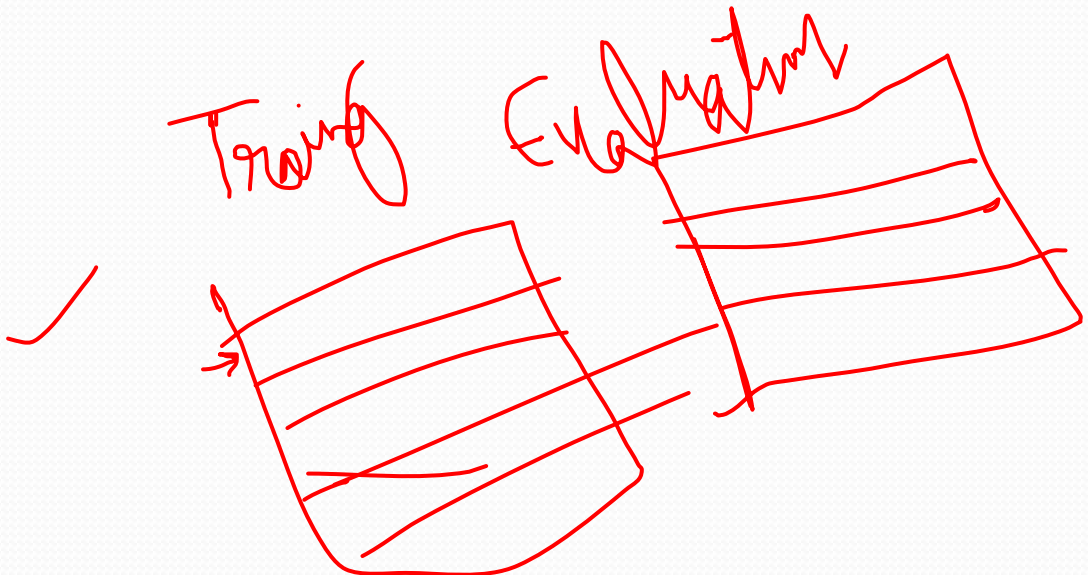
Terminology: Bias



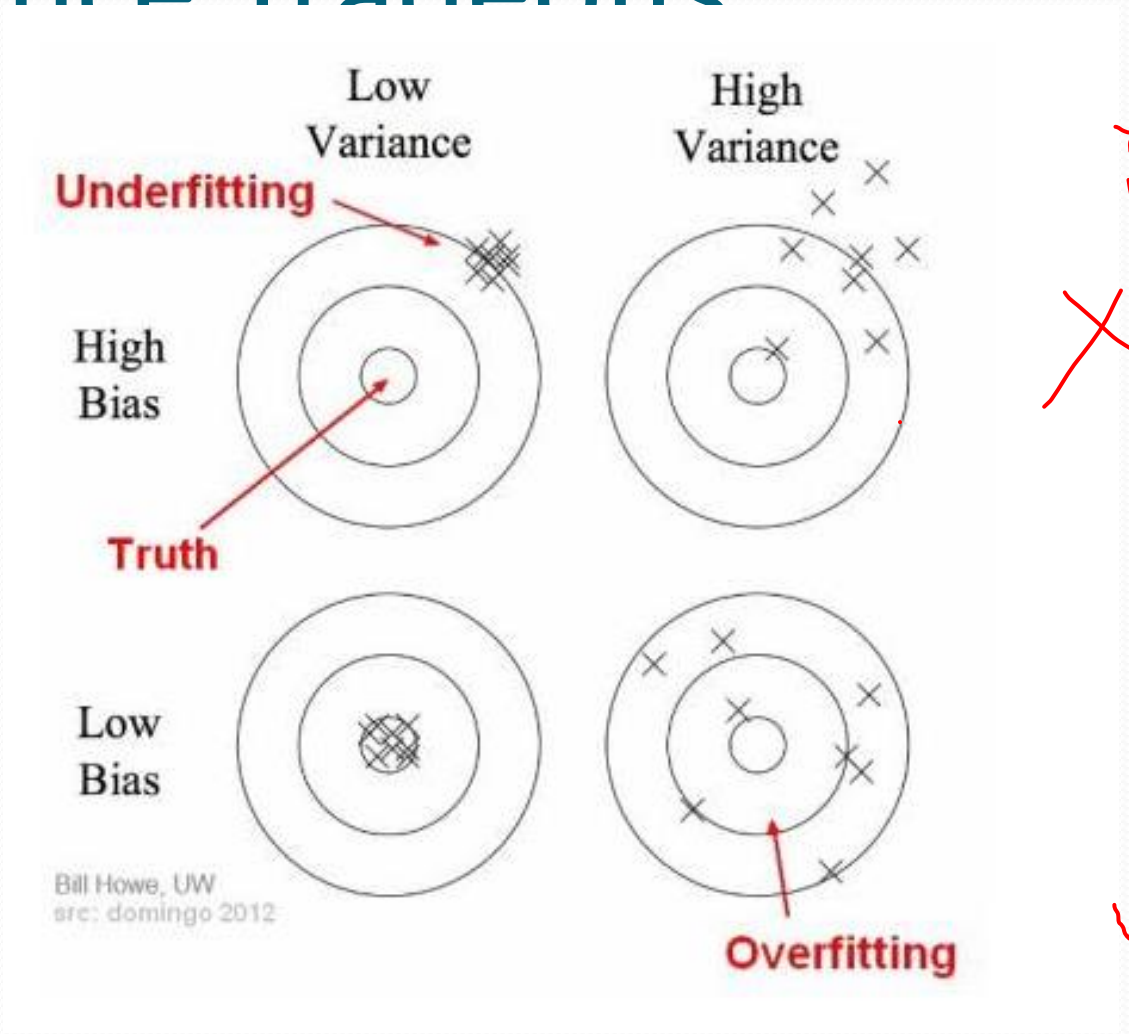
- Bias is an error from erroneous assumptions in the learning algorithm.
- High bias can cause an algorithm to miss the relevant relations between features and target outputs
- Bias is the algorithm's tendency to consistently learn the wrong thing by not taking into account all the information in the data
- Example
  - If you have a simple model, you might conclude that every "Alex" are amazing people. This presents a High Bias and Low Variance problem. Your dataset is 'biased' towards people with the name Alex. Thus, most predictions will be similar, since you believe people with 'Alex' act a certain way.

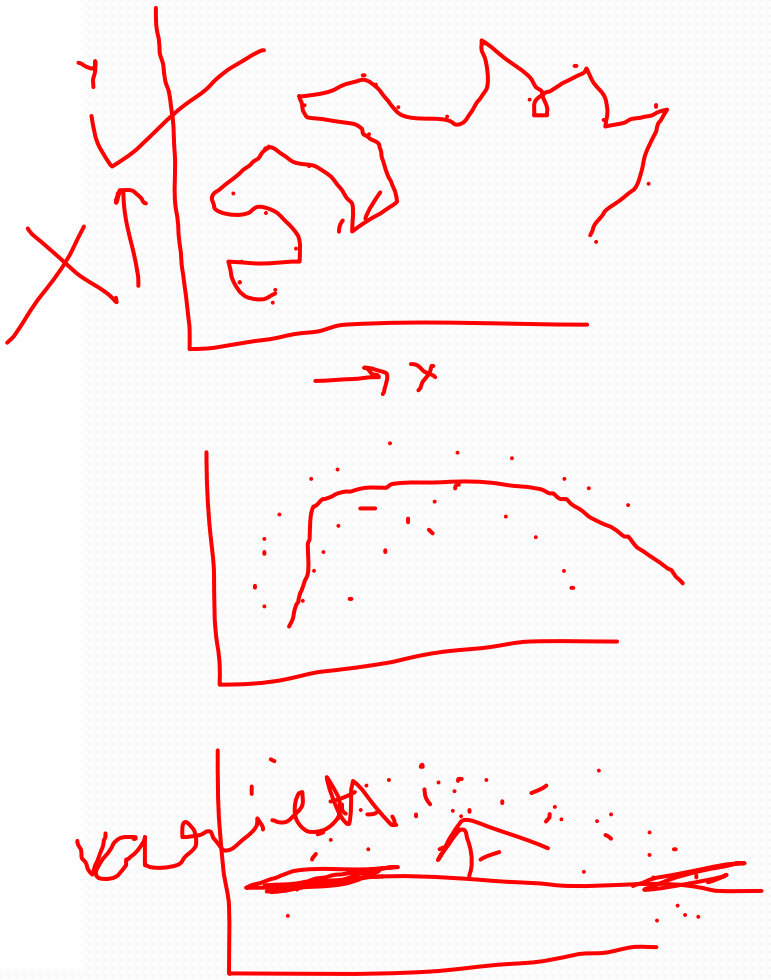
Terminology: Variance

- Variance is the variability of model prediction for a given data point or a value which tells us spread of our data.
- Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before.
- As a result, such models perform very well on training data but has high error rates on test data.



Bias-Variance Tradeoffs





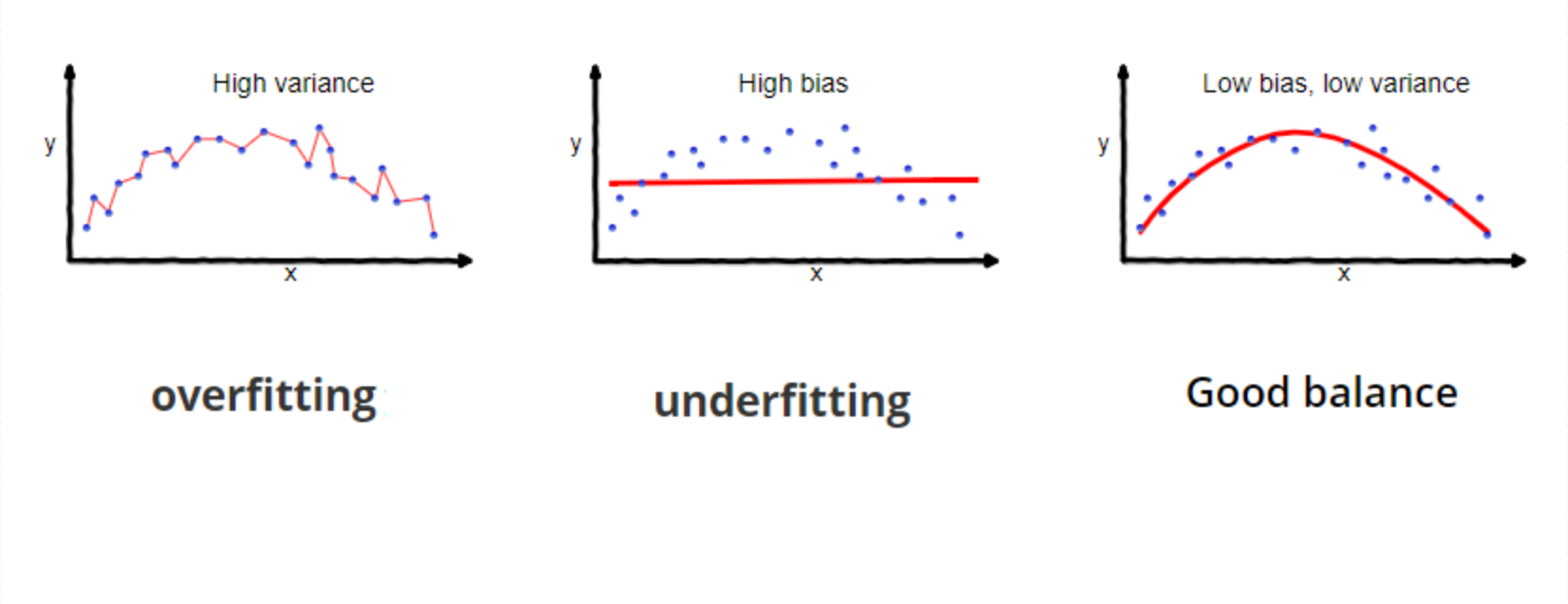
Overfitting

- Overfitting refers to a model that models the training data too well.
- Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.
- This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model.
- The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize.

Underfitting

- Underfitting refers to a model that can neither model the training data nor generalize to new data.
- An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data.
- Underfitting is often not discussed as it is easy to detect given a good performance metric.

Overfitting Vs. Underfitting



Model Evaluation

- How the model generalizes on unseen data ?
- Whether it actually works and, consequently, if we can trust its predictions.
- Could the model be merely memorizing the data it is fed with, and therefore unable to make good predictions on future samples, or samples that it hasn't seen before?
- Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data.

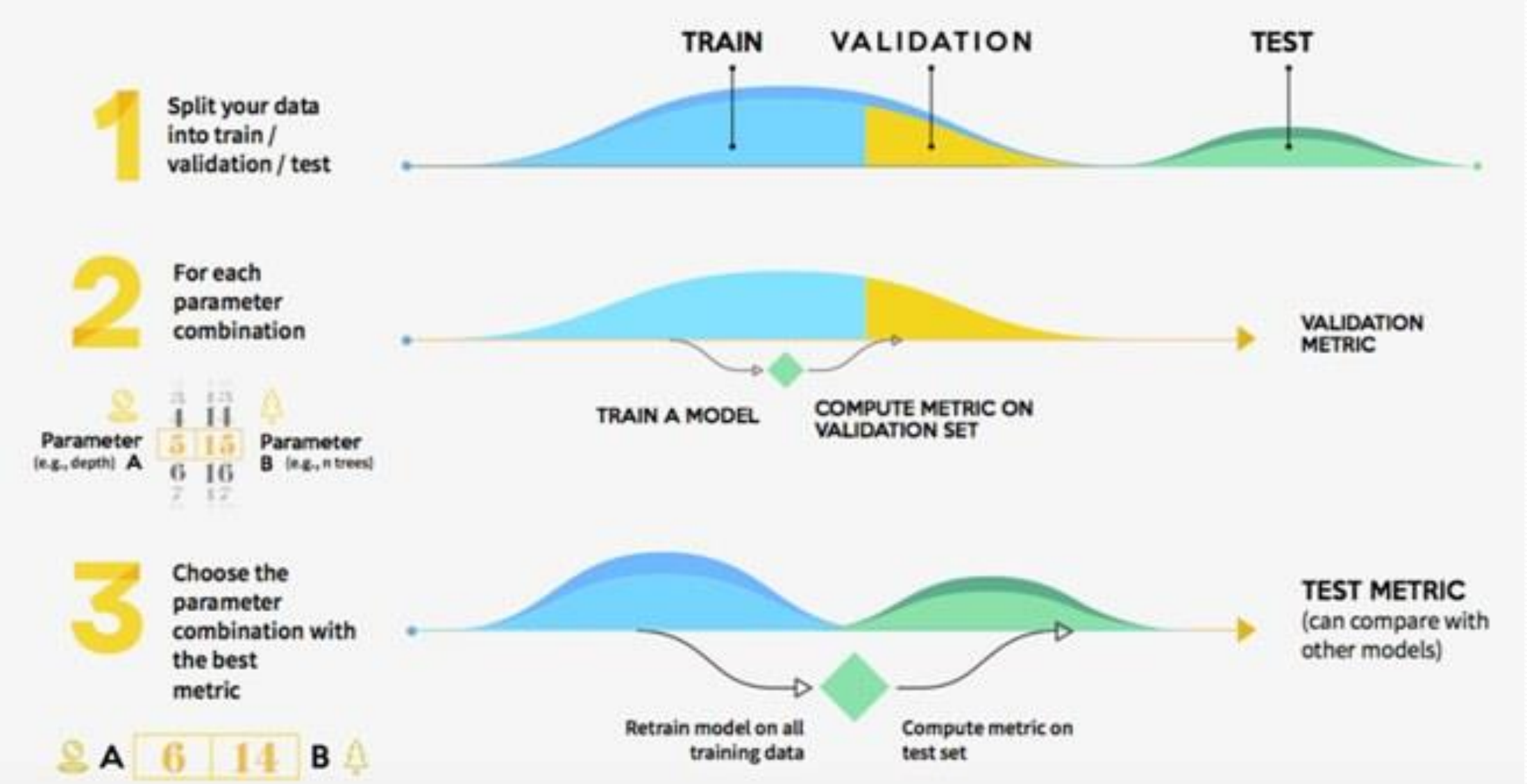
Model Evaluation Techniques

- Most common methods
  - Holdout
  - Cross-validation
- Both methods use a test set (i.e data not seen by the model) to evaluate model performance.
- It's not recommended to use the data we used to build the model to evaluate it.
- This is because our model will simply remember the whole training set, and will therefore always predict the correct label for any point in the training set. That means overfitting.

Model Evaluation Techniques : Holdout

- The purpose is to test a model on different data than it was trained on and hence to get an unbiased estimate of learning performance.
- In this method, the dataset is *randomly* divided into three subsets:
  - **Training set** is a subset of the dataset used to build predictive models.
  - **Validation set** is a subset of the dataset used to assess the performance of the model built in the training phase. It provides a test platform for fine-tuning a model's parameters and selecting the best performing model. Not all modeling algorithms need a validation set.
  - **Test set**, or unseen data, is a subset of the dataset used to assess the likely future performance of a model. If a model fits to the training set much better than it fits the test set, overfitting is probably the cause.

Hold-Out Method



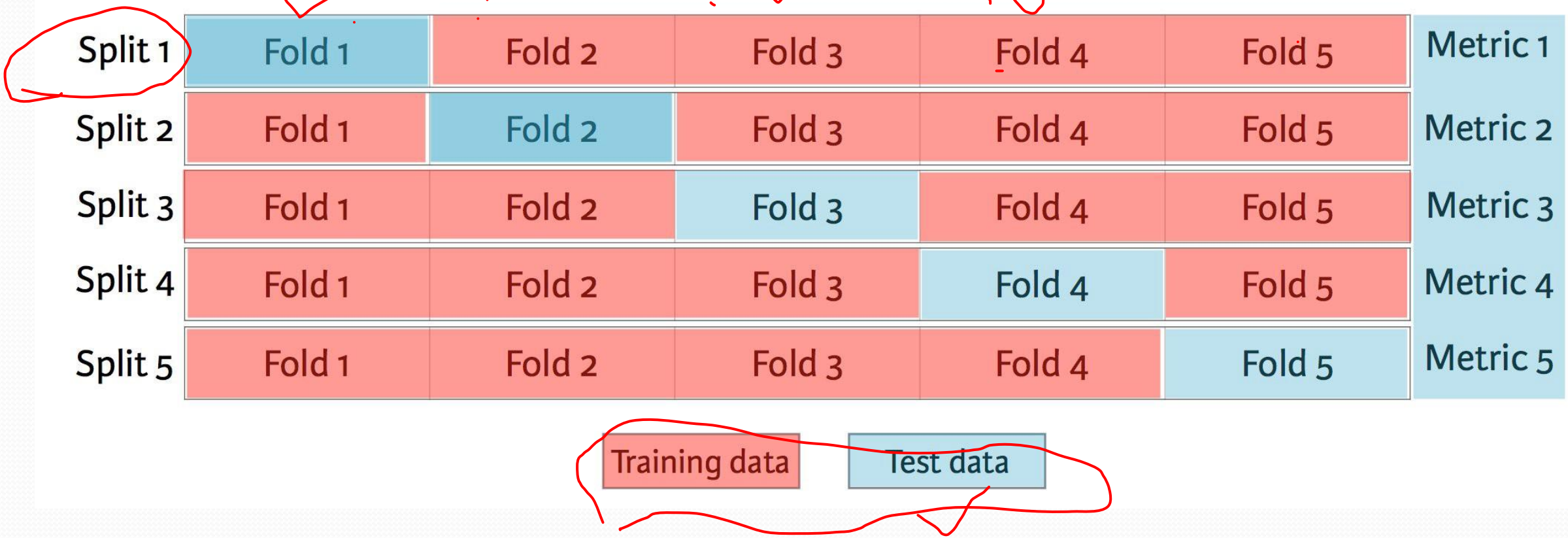
Model Evaluation Techniques : Cross-Validation

- The most common cross-validation technique is **k-fold cross-validation**, where the original dataset is partitioned into k equal size subsamples, called folds.
- The k is a user-specified number, usually with 5 or 10 as its preferred value.
- This is repeated k times, such that each time, one of the k subsets is used as the test set/validation set and the other k-1 subsets are put together to form a training set.
- The error estimation is averaged over all k trials to get the total effectiveness of our model.

Model Evaluation Techniques : Cross-Validation

- As can be seen, every data point gets to be in a test set exactly once and gets to be in a training set k-1 times.
- This significantly reduces bias, as we're using most of the data for fitting, and it also significantly reduces variance, as most of the data is also being used in the test set.
- Interchanging the training and test sets also adds to the effectiveness of this method.
- Cross-validation techniques can also be used to compare the performance of different machine learning models on the same data set and can also be helpful in selecting the values for a model's parameters that maximize the accuracy of the model—also known as parameter tuning.

Model Evaluation Techniques : Cross-Validation



Model Evaluation Metrics

- Model evaluation metrics are required to quantify model performance.
- The choice of evaluation metrics depends on a given machine learning task (such as classification, regression, ranking, clustering, topic modeling, among others).

Model Evaluation Metrics

- Classification Accuracy
- Confusion matrix
- Logarithmic Loss
- Area under curve (AUC)
- F-Measure
- Mean Absolute Error (or MAE)
- Root Mean Squared Error (RMSE)

Thank You

For your precious time

Online Student Training for "Artificial Intelligence & Machine Learning"

(4th Feb, 2021 – 17th Mar, 2021)

LINEAR REGRESSION

Faculty Trainer

Nawvarsha Pipada

Department of Computer Science & E

Engineering College Bikane

Regression

Regression is a method of modelling a target value based on independent predictors.

This method is mostly used for forecasting and finding out cau and effect relationship between variables.

Regression techniques mostly differ based on the number of independent variables and the type of relationship between the independent and dependent variables.

Regression types

Based on type of relationship between the independent and dependent variables

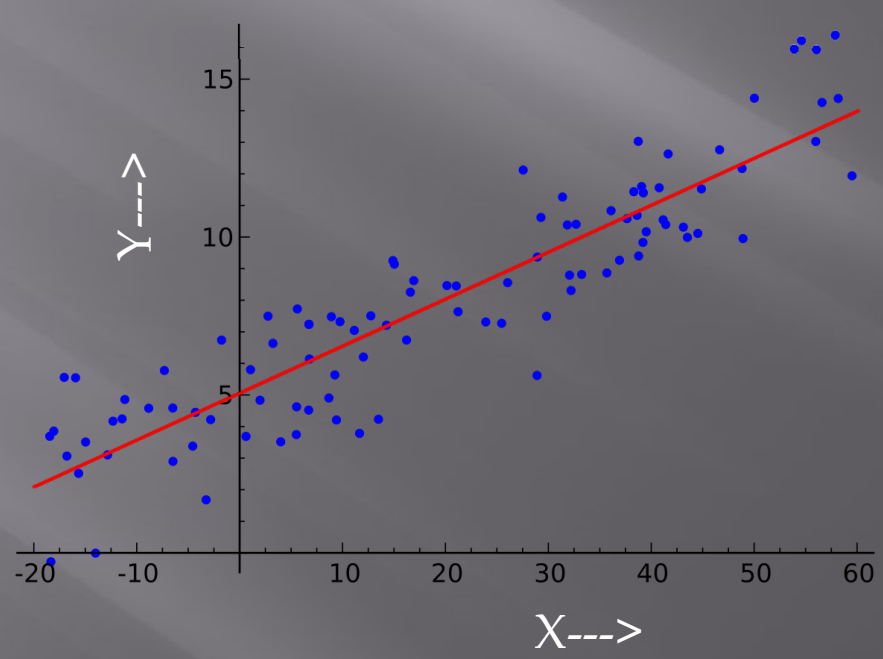
- Linear
- Non-Linear

Based on the number of independent variables (as dependent variable will be only one)

- Simple
- Multiple

Simple Linear Regression

Simple linear regression is a type of regression analysis where the number of independent variables is one and there is a linear relationship between the independent(x) and dependent(y) variable.



Datapoints
Regression Line

Simple Linear Regression

The regression line is referred to as the best fit straight line.

Based on the given data points, we try to plot a line that models the points the best.

The line can be modelled based on the linear equation shown below.

$$y = a_0 + a_1 * x \qquad \text{## Linear Equation}$$

Two important Concepts

- Cost Function
  - A cost function is a measure of how wrong the model is in terms of its ability to estimate the relationship between X and y.
  - This is typically expressed as a difference or distance between the predicted value a the actual value
  - In our case, the cost function is actually minimization problem where we would lik minimize the error between the predicted value and the actual value. It is also know ad mean squared er

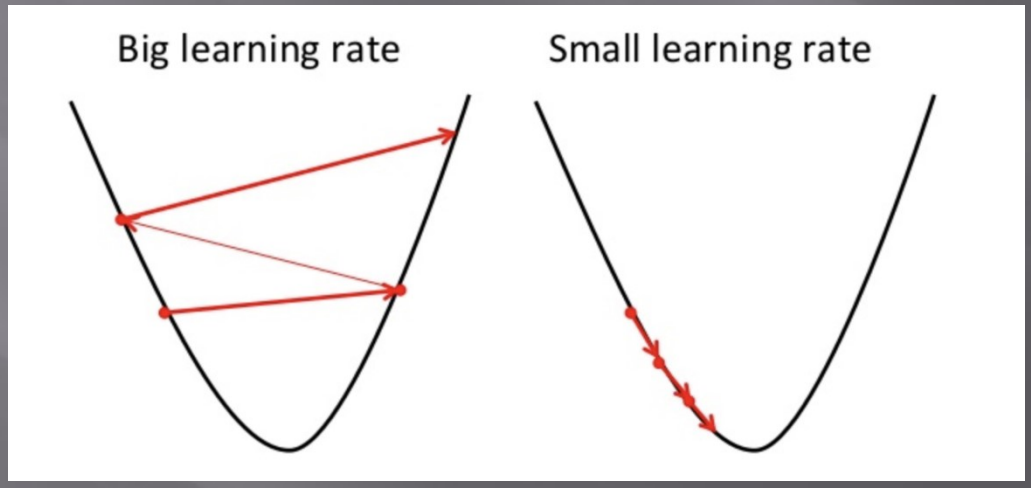
$$minimize \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

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

Two important Concepts

Gradient Descent

- Gradient descent is a method of updating a_0 and a_1 to reduce the cost function(MS
- The idea is that we start with some values for a_0 and a_1 and then we change these values iteratively to reduce the cost.
- Gradient descent helps us on how to change the values.



To find the gradients

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

$$J = \frac{1}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i)^2$$

$$\frac{\partial J}{\partial a_0} = \frac{2}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i) \implies \frac{\partial J}{\partial a_0} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$

$$\frac{\partial J}{\partial a_1} = \frac{2}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i) \cdot x_i \implies \frac{\partial J}{\partial a_1} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i) \cdot x_i$$

Grad

Updating a_0 and a_1

Alpha is the learning rate which is a hyperparameter that you must speci

A smaller learning rate could get you closer to the minima but takes more time to reach the minima, a larger learning rate converges sooner but ther is a chance that you could overshoot the minima.

$$a_0 = a_0 - \alpha \cdot \frac{2}{n} \sum_{i=1}^{n} (pred_i - y_i)$$

$$a_1 = a_1 - \alpha \cdot \frac{2}{n} \sum_{i=1}^{n} (pred_i - y_i) \cdot x_i$$

Goodness-of-fit

R-squared is a statistical measure of how close the data are to th fitted regression line.

It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

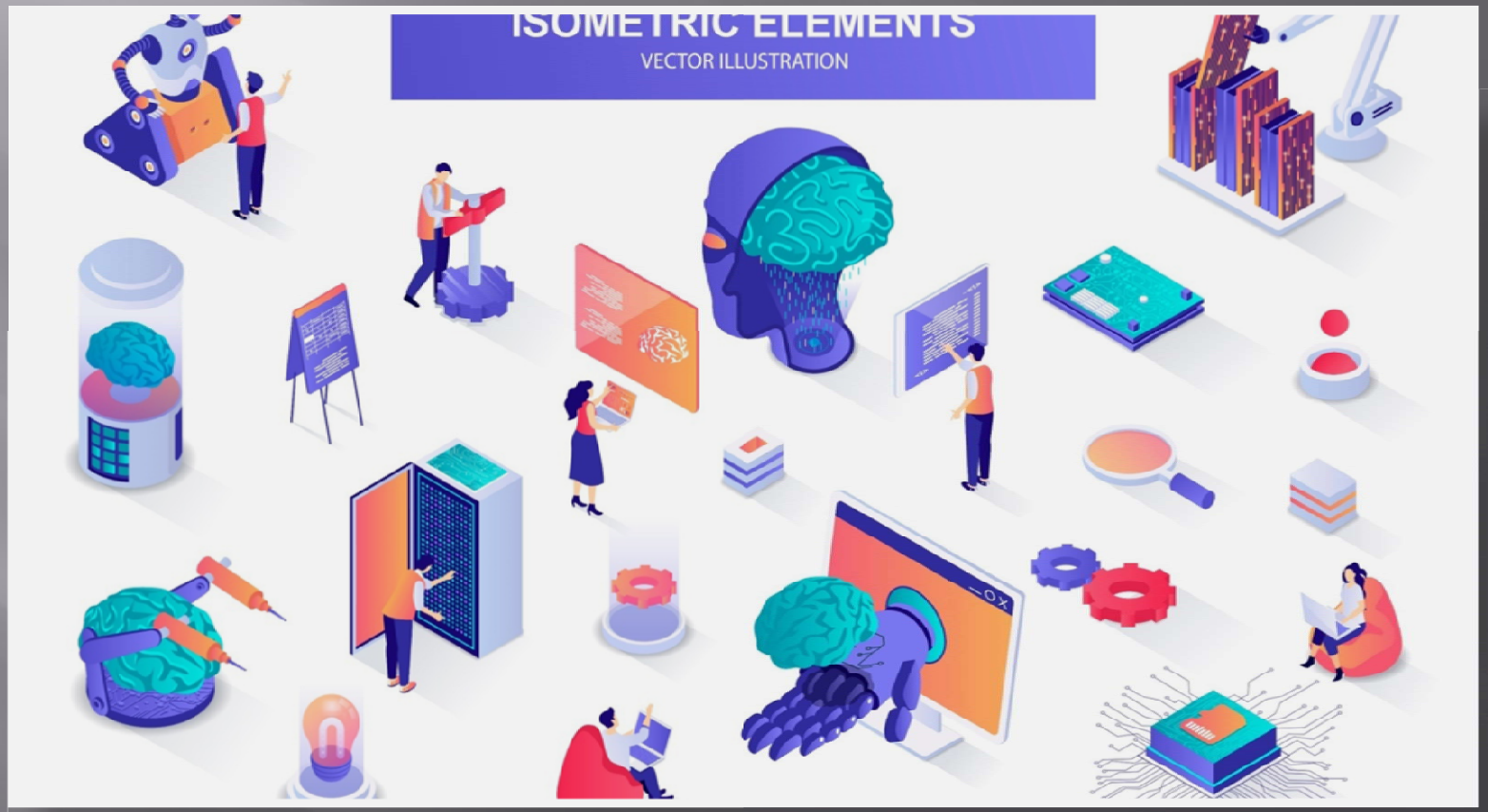
In general, the higher the R-squared, the better the model fits your data

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

Steps for Implementing Linear Regression

1. Check if there exist linear relationship between independent and dependent variable.
2. Load training and test data from dataset files.
3. Initialize the values of learning rate, slope, intercept number of runs (update slope and intercept) .
4. Predict the value of target for training data and calculate the residual error.
5. Update the values of slope and intercept using gradients to minimize residual errors.
6. After getting, final values of slope and intercept, predict the targe values for testing data.
7. Evaluate the fitness of model using R2-Score.

THANK YOU!!!!!



Online Student Training for "Artificial Intelligence & Machine Learning"

(4th Feb, 2021 –17th Mar, 2021)

Decision Tree Classifier

Faculty Trainer

Naw Varsha Pipada

Department of Computer Science & Engineering

Engineering College Bikaner

Contents

01 **Decision Tree**

Introduction to Decision Tree.

02 **How it works**

Understand the steps and process

03 **Generation of Decision Tree**

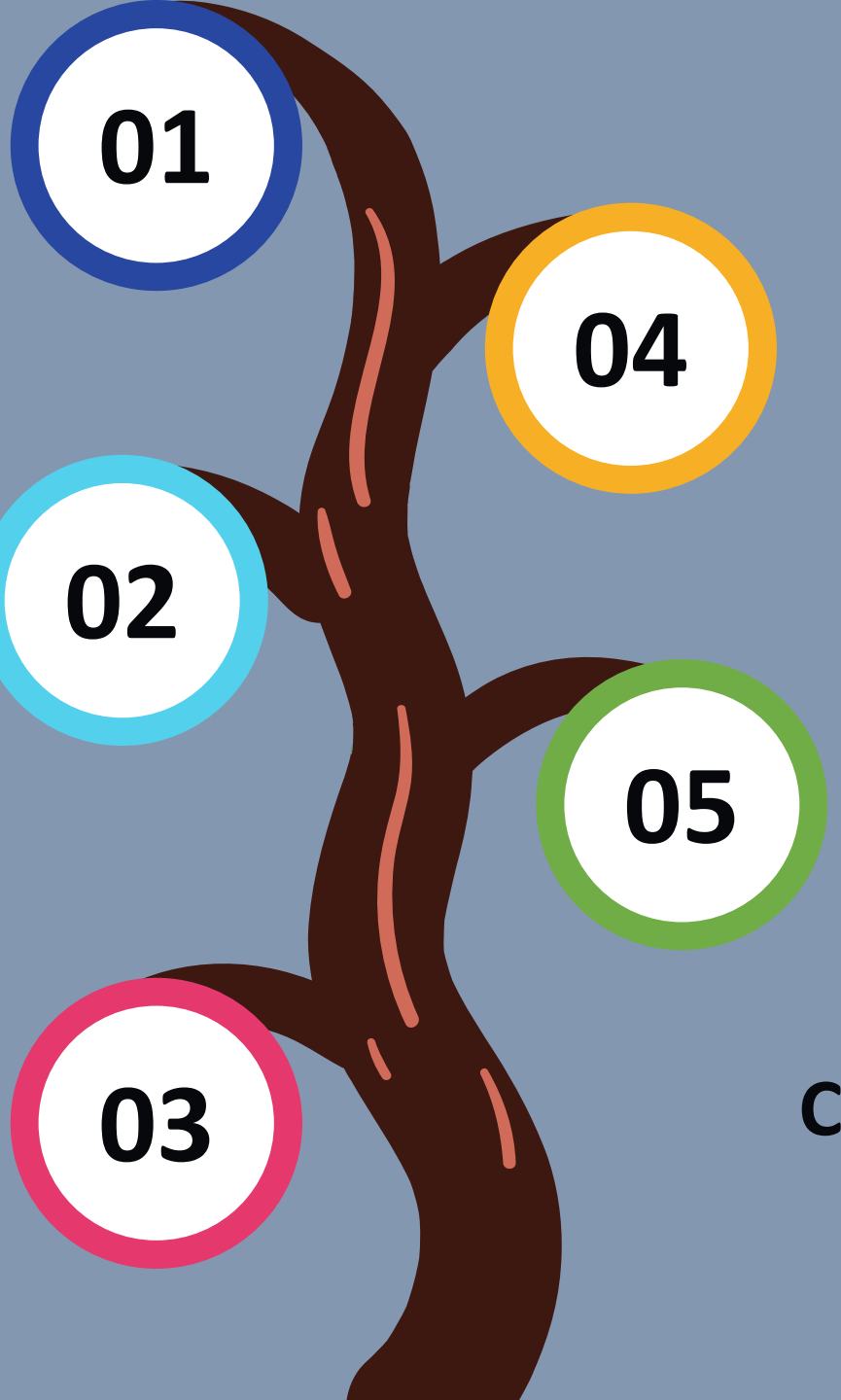
Entropy & Information Gain

04 **Examples & Limitations**

To get a clear understanding

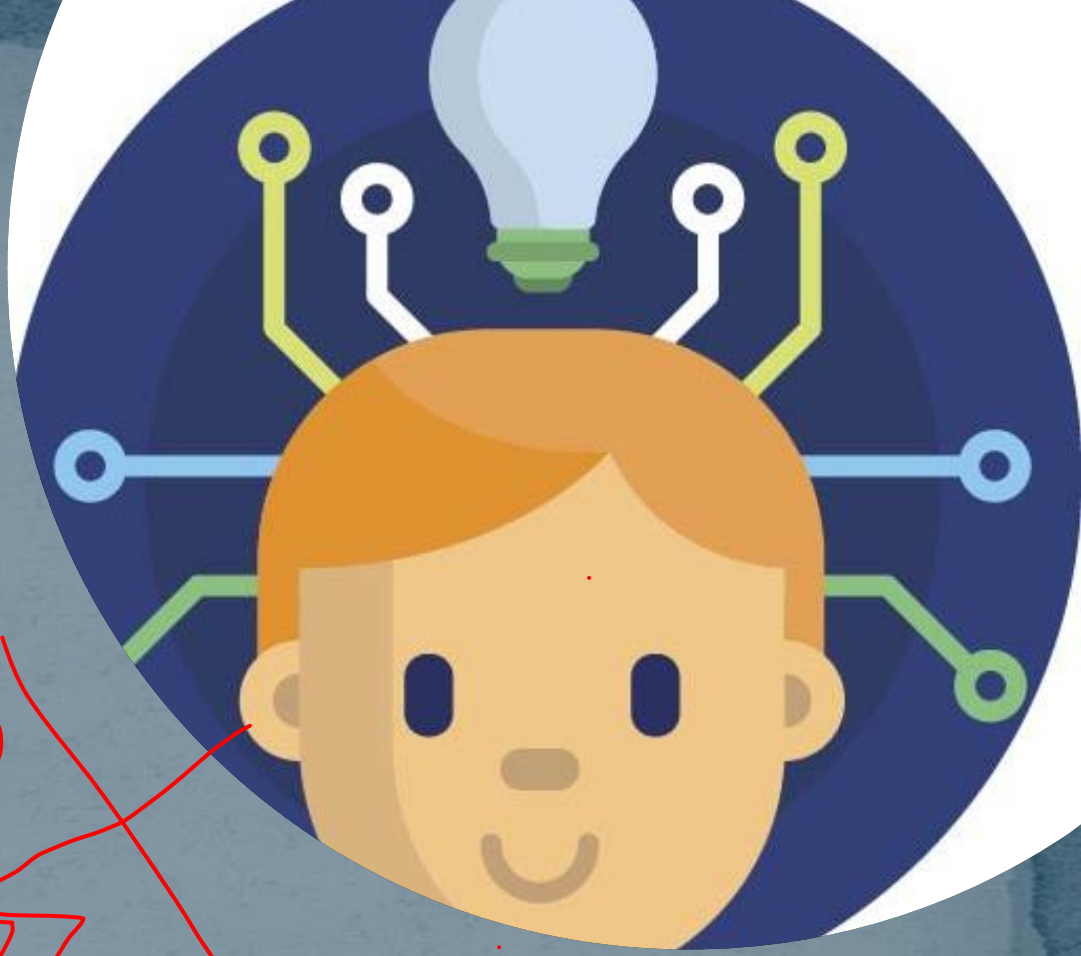
05 **Metrics for Classifiers**

How to evaluate the classifier you created



Guess the type of problem

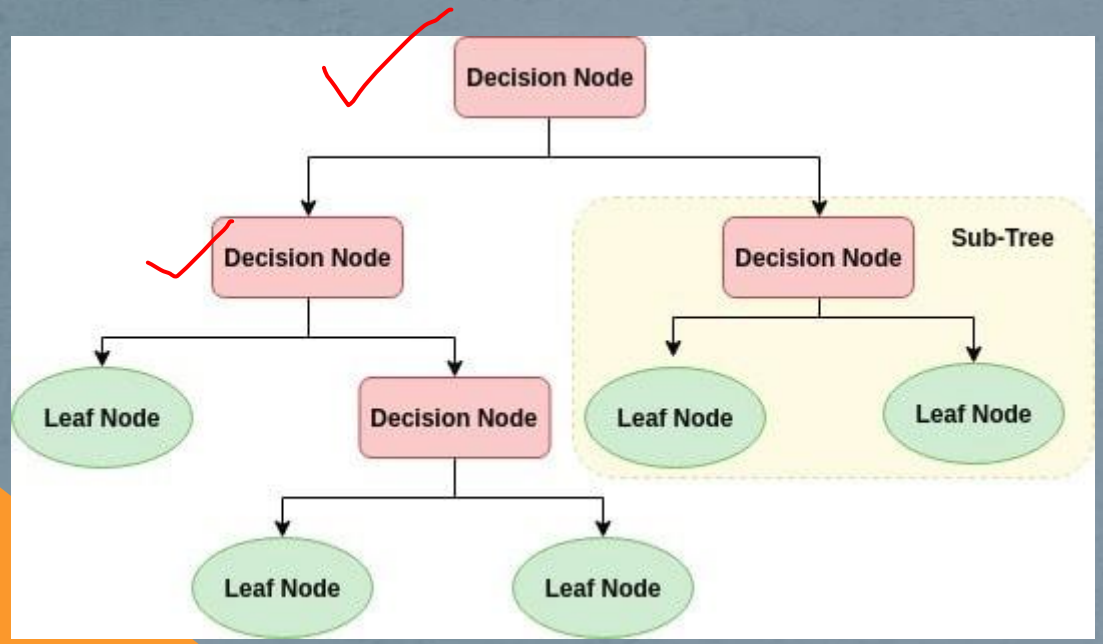
- Approve or reject the loan ?
- Whether the car will give high mileage or low mileage?
- Should I play Tennis today or not?



Decision Tree

- Decision tree is one of the method to find solution to classification problem.
- A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.
- The topmost node in a decision tree is known as the root node.
- It learns to partition on the basis of the attribute value using recursive partitioning. This flowchart-like structure helps you in decision making

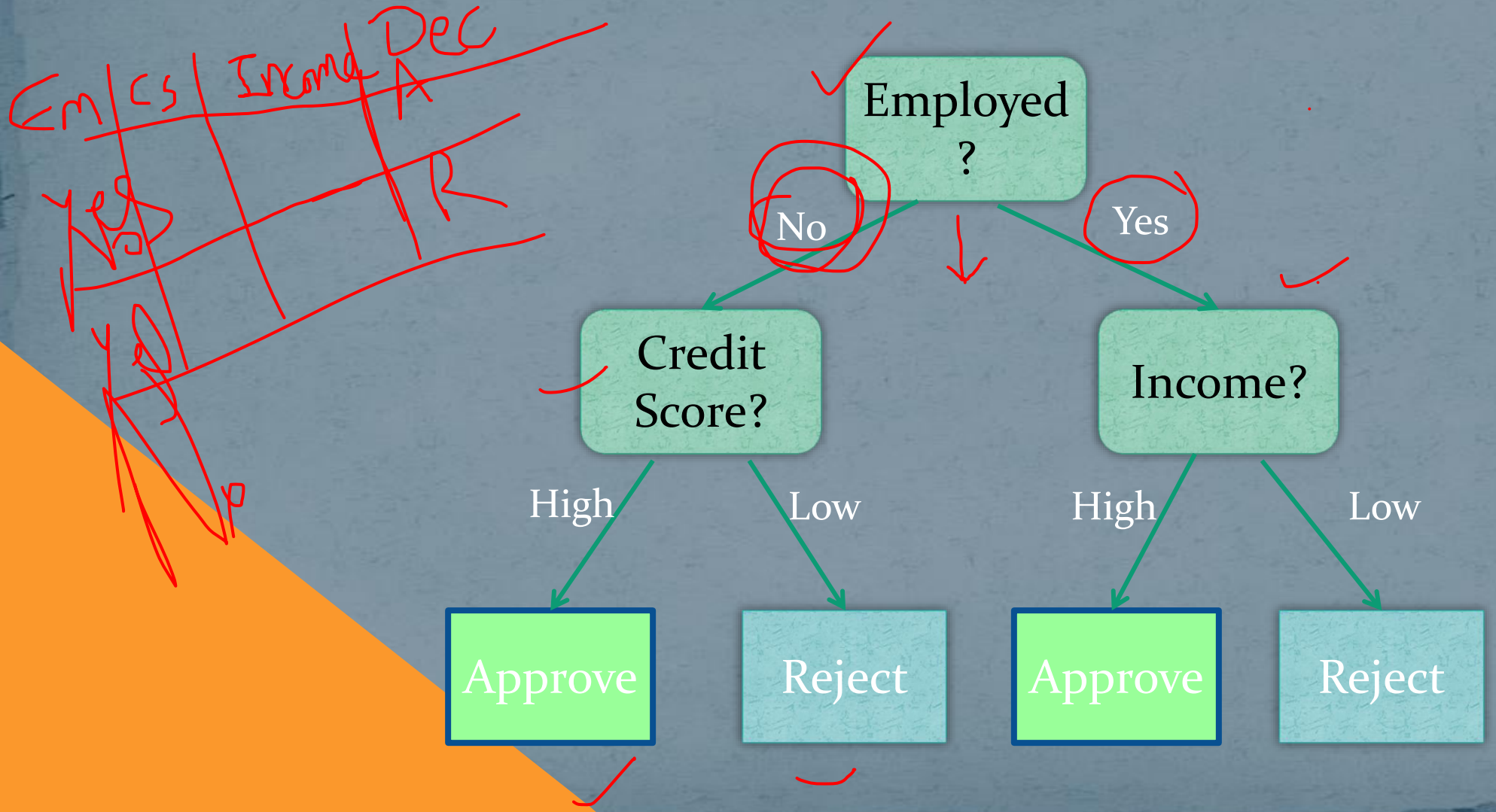
Decision Tree



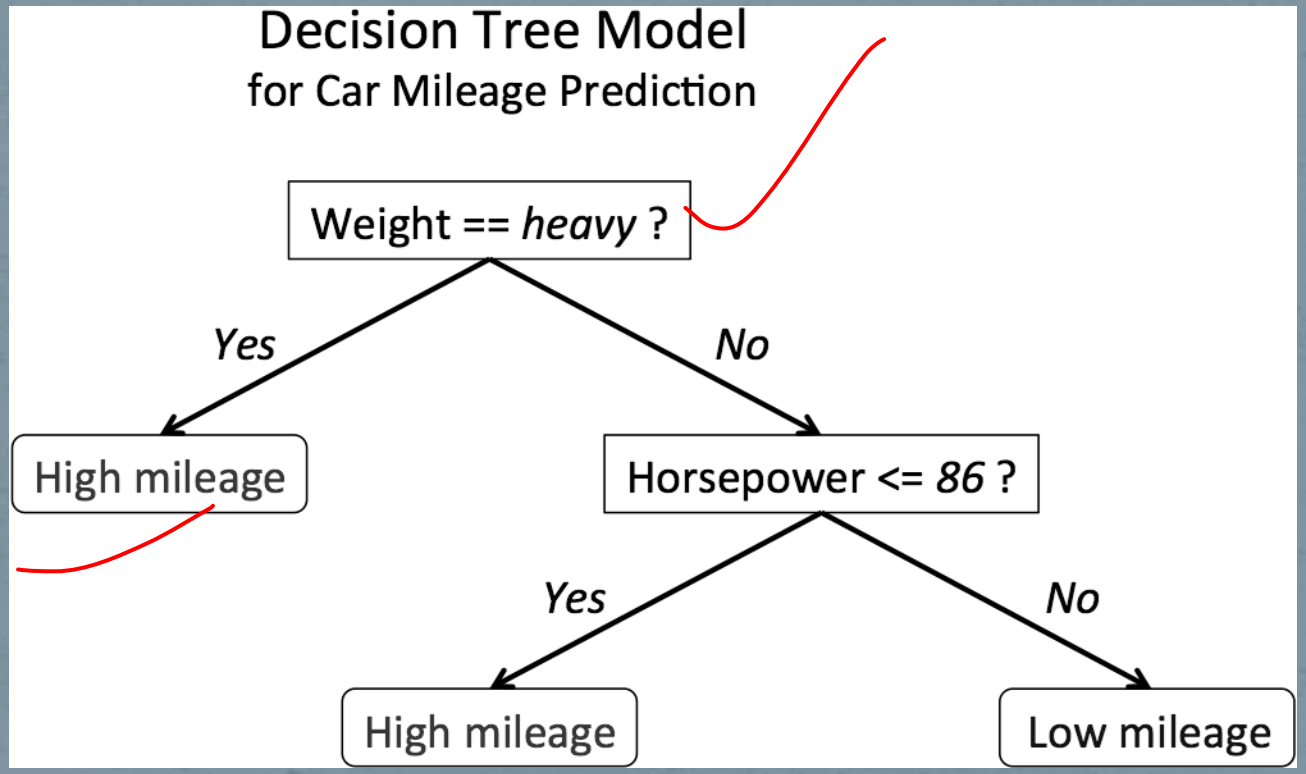
Decision node: Specifies a choice or test of some attribute, with one branch for each outcome

Leaf node: Indicates classification of an example

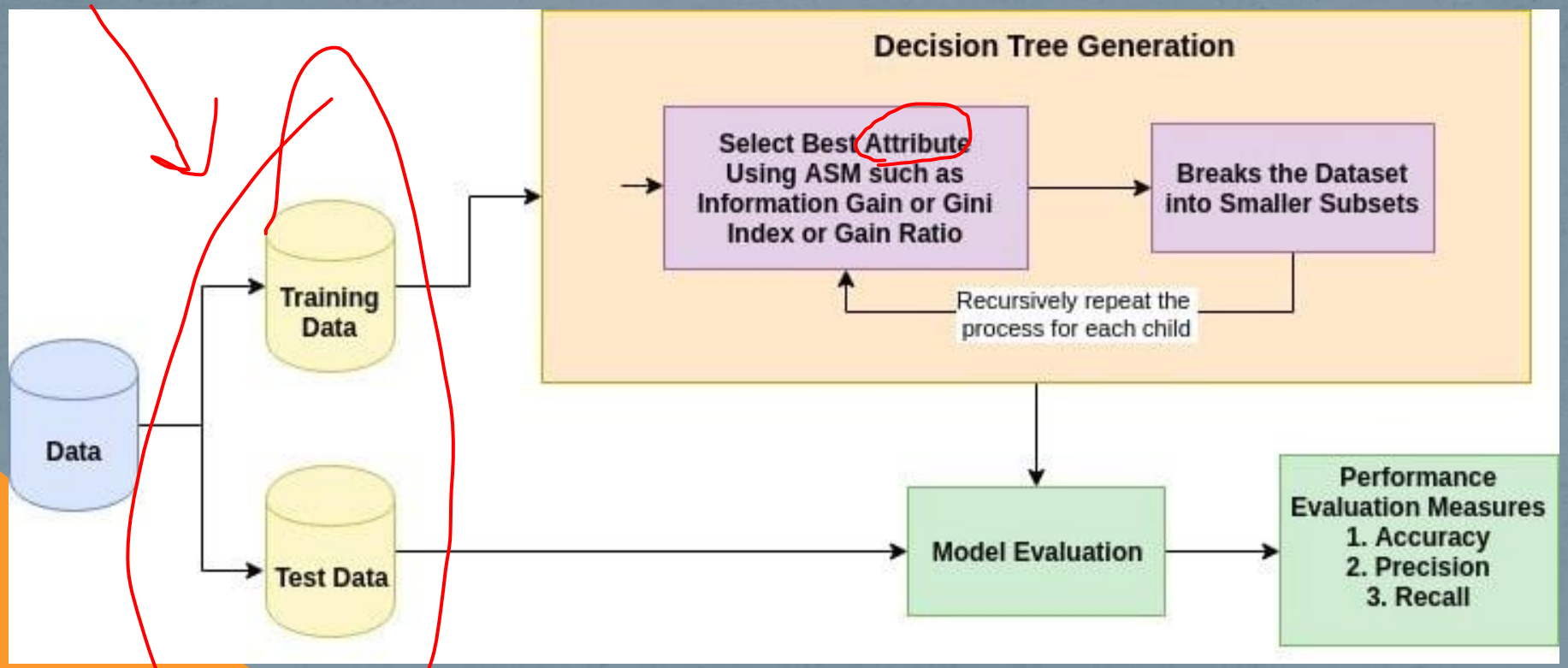
Example : Whether to approve a loan?



Example: Predicting Car Mileage (High/Low)



How it Works


Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Learning with Decision Tree Classifier

- To predict the output of classification problem
  - Generate the Decision Tree from Training Set
  - Predict the output of test data with the decision tree formed.
  - Use the metrics (such as accuracy , precision, recall) to evaluate model
  - Tune hyper parameters if required.

How it Works: To generate Decision Tree

1. Select the best attribute using Attribute Selection Measures(ASM) to split the records.
2. Make that attribute a decision node and breaks the dataset into smaller subsets.
3. Starts tree building by repeating this process recursively for each child until one of the condition will match:
   - All the tuples belong to the same attribute value.
   - There are no more remaining attributes.
   - There are no more instances.

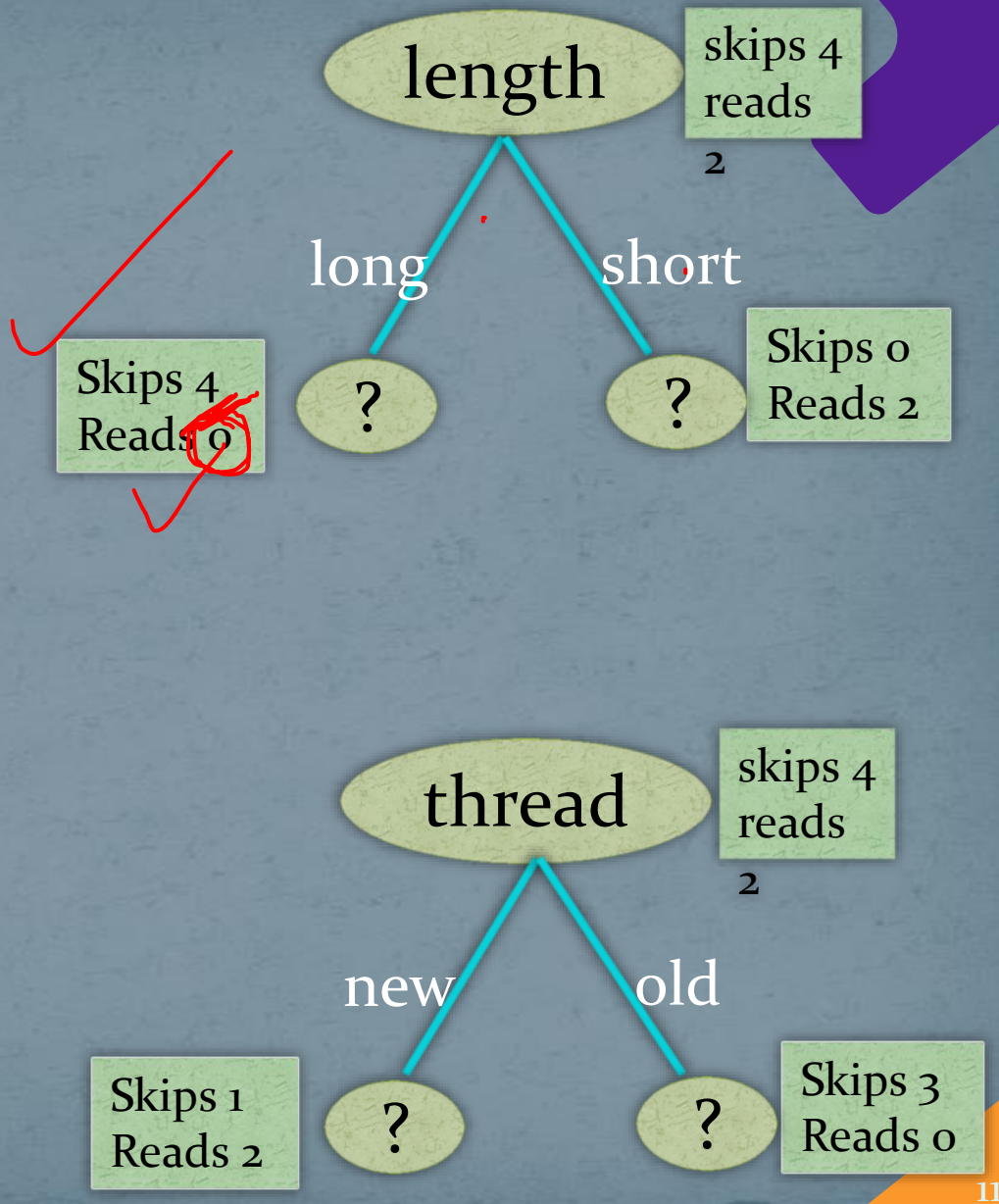
Example

Training Examples:

	Action	Author	Thread	Length	Where
e1	skips	known	new	long	Home
e2	reads	unknown	new	short	Work
e3	skips	unknown	old	long	Work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work

New Examples:

e7	???	known	new	short	work
e8	???	unknown	new	short	work

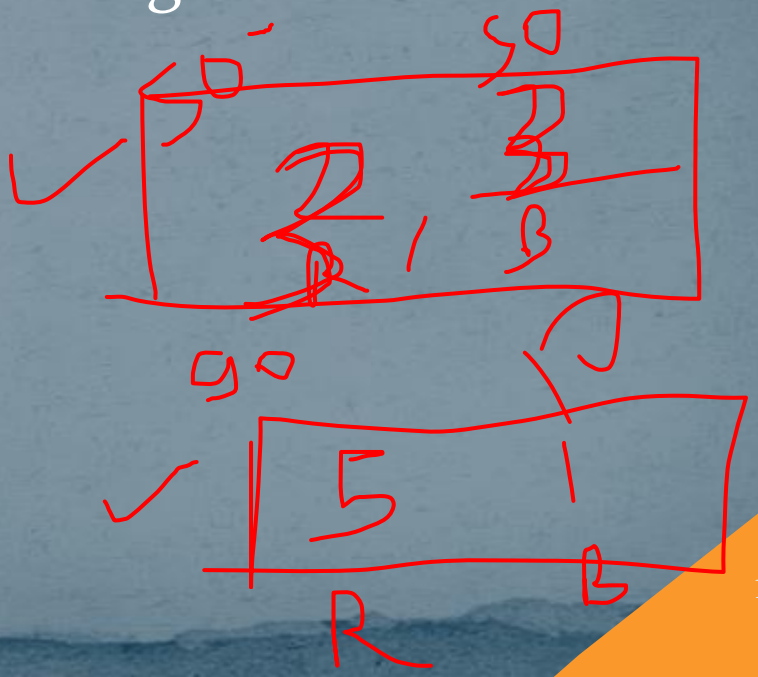


How to choose attribute for splitting?

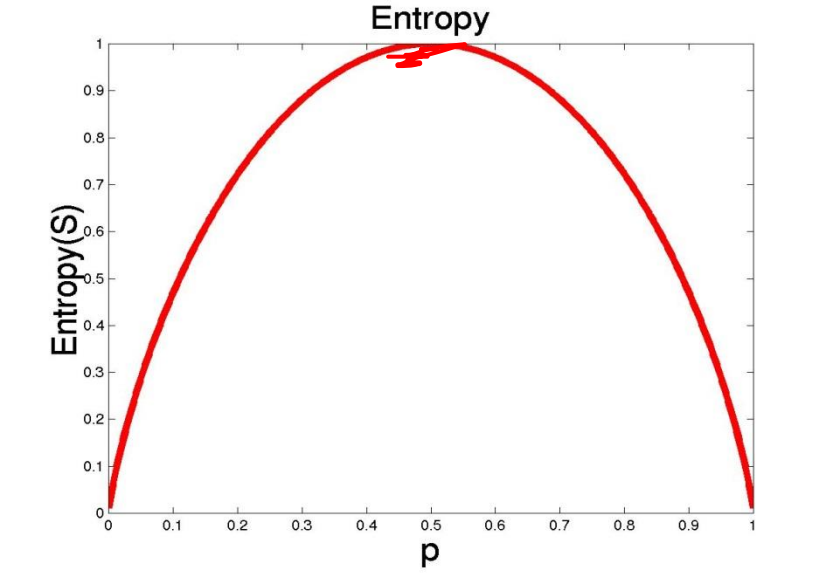
- Attribute selection measure is a heuristic for selecting the splitting criterion that partition data into the best possible manner.
- Attribute with best score of ASM will be selected as a splitting attribute .
- Most popular selection measures are
  - Information Gain (ID3 Algorithm)
  - Gini Index (CART Algorithm)

Information Gain

- Entropy referred as the randomness or the impurity in the system. In information theory, it refers to the impurity in a group of examples.
- Less Entropy means high information gain.
- Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values



Entropy



- The entropy is o if the outcome is ``certain".
- The entropy is maximum if we have no knowledge of the system (or any outcome is equally possible).

Entropy

- If S is set of training instances
- P_1 is the proportion of class 1 instances
- P_2 is the proportion of class 2 instances
- Entropy measures the impurity of S, using the formula

  $\text{Entropy}(S) = -P_1 \log_2 P_1 - P_2 \log_2 P_2$
- Similarly for m classes,
  - $\text{Entropy}(S) = -\sum_{i=1}^{m} P_i \log_2 P_i$

Information Gain

- Information Gain of an attribute A in a set S is given by

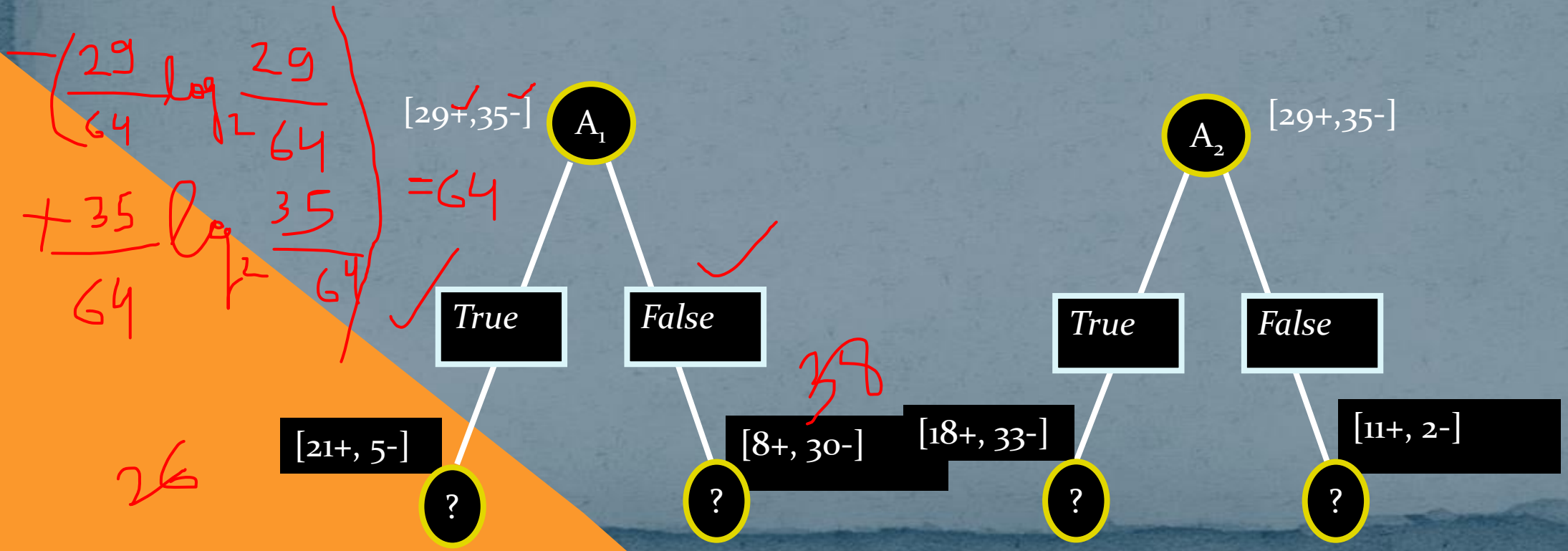
$$Gain(S,A)=Entropy(S) - \sum_{v \in values(A)} |S_v|/|S| \; Entropy(S_v)$$

- where v - Values of attribute A

$|S_v|$- Number of instances in the set S having value v for attribute A

Example

- Let us consider a case where target output has two possible classes positive (+) and negative(-). Total number of training instances is 64.
- Out of two attributes A_1 and A_2 , which will be considered as best are to be chosen



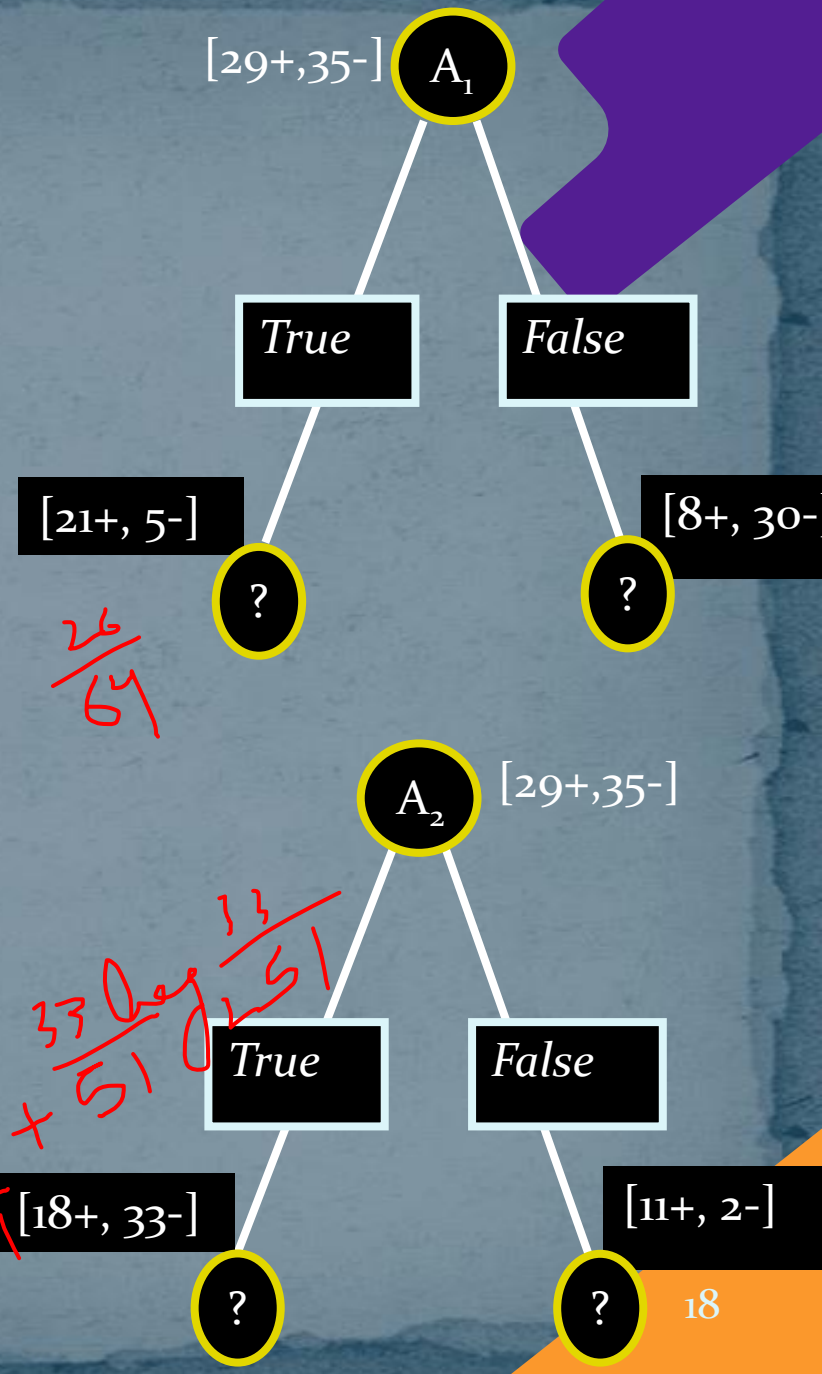
Example: Find Information Gain

- Entropy(S) = Entropy([29+,35-])
  = -29/64 $\log_2$ 29/64 – 35/64 $\log_2$ 35/64
  = 0.99

Entropy([21+,5-]) = 0.71
Entropy([8+,30-]) = 0.74
Gain(S,A_1)=Entropy(S)
-26/64*Entropy([21+,5-])
-38/64*Entropy([8+,30-])
=0.27

Entropy([18+,33-]) = 0.94
Entropy([8+,30-]) = 0.62
Gain(S,A_2)=Entropy(S)
-51/64*Entropy([18+,33-])
-13/64*Entropy([11+,2-])
=0.12

Gain(S,A)=Entropy(S) $-\sum_{v \in values(A)} |S_v|/|S|$ Entropy(S_v)



Another Measure : GINI Index

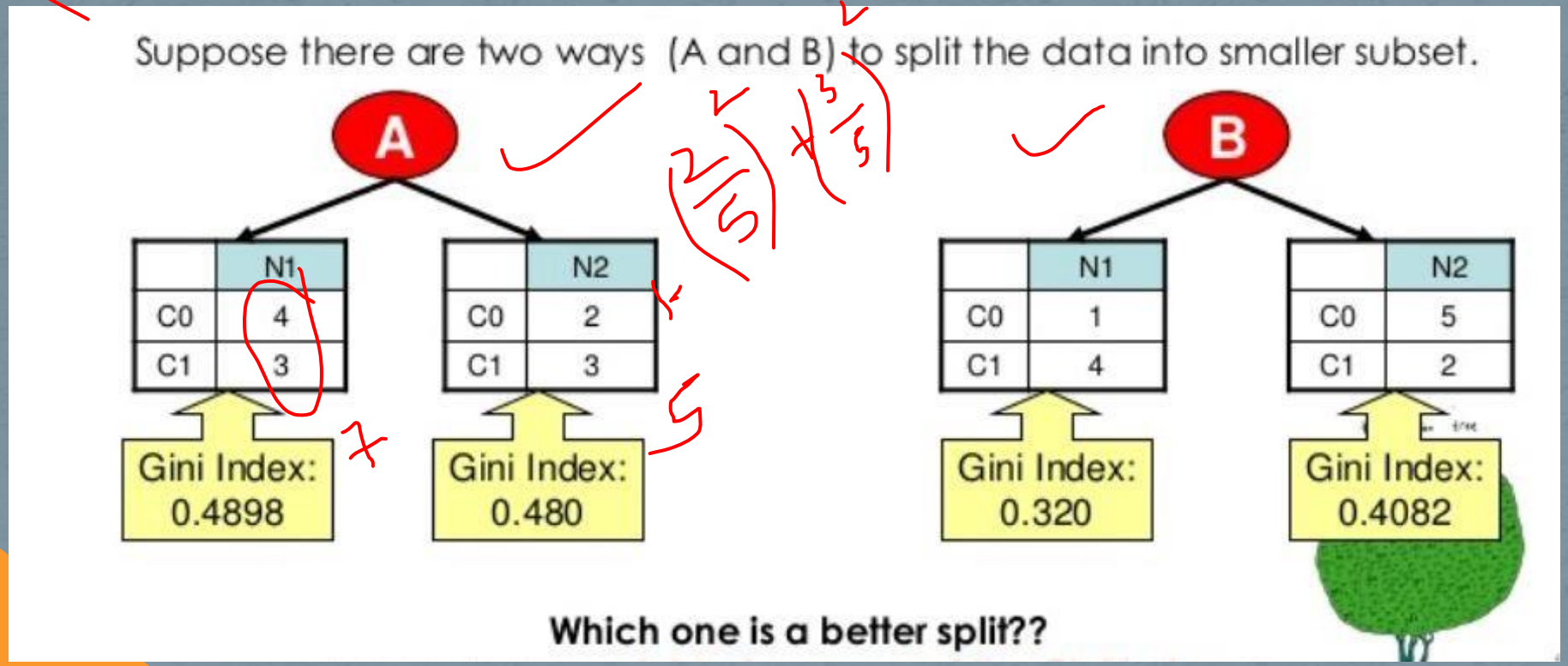
$$GINI_{node}(Node) = 1 - \sum_{c \in classes} [p(c)]^2$$

$$GINI_{split}(A) = \sum_{v \in Values(A)} \frac{|S_v|}{|S|} GINI(N_v)$$

- where

p(c) - proportion of class c instances in the node

$GINI(N_v)$ - represent the GINI index for node with value v of attribute A



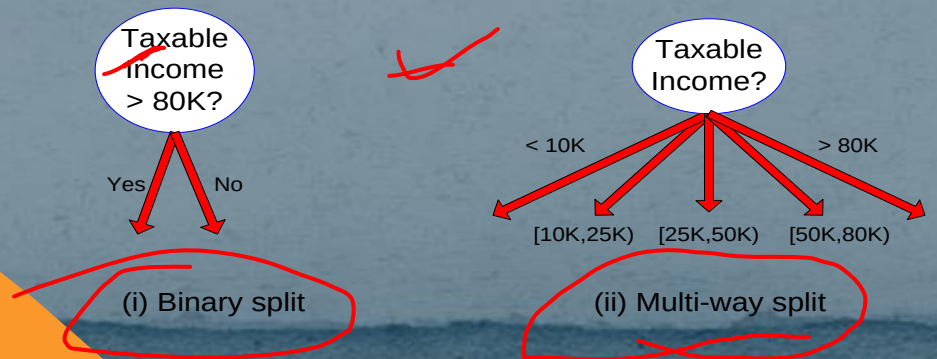
- $GINI_{split}(A)=(7/12)*0.4898+(5/12)*0.480=0.4857$
- $GINI_{split}(B)=(5/12)*0.320+(7/12)*0.4082=0.3715$

Generate decision tree
(Target Feature is Tennis?)

Day	Outlook	Temp	Humidity	Wind	Tennis?
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

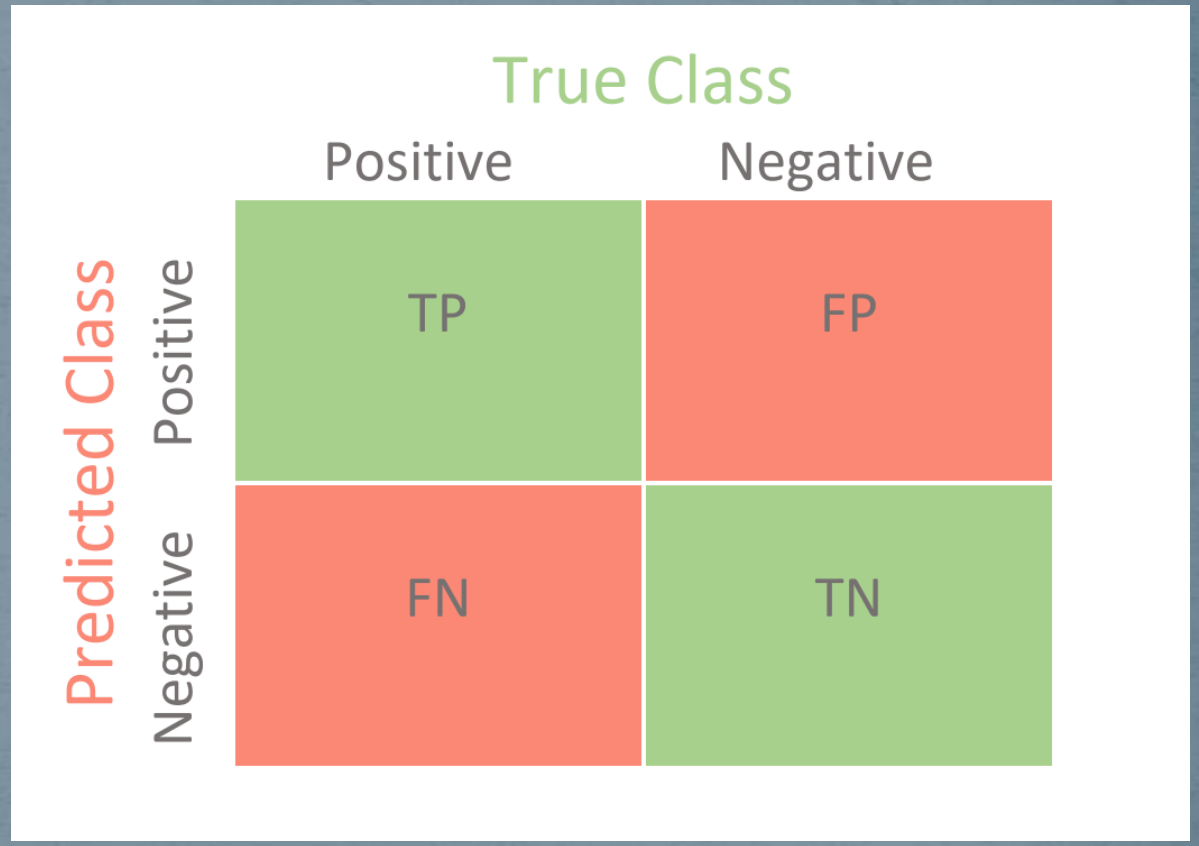
Limitations

- A small change in the data can cause a large change in the structure of the decision tree causing instability.
- For a Decision tree sometimes calculation can go far more complex compared to other algorithms.
- Decision tree often involves higher time to train the model.
- Handling missing values could be challenge.
- The Decision Tree algorithm is inadequate for applying regression and predicting continuous values but can be handled as follows



Metrics for Classifiers

- A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known
- It is extremely useful for measuring
  - Recall
  - Precision
  - Accuracy
  - F-Score



Metrics for Classifiers

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

$$\textit{Accuracy} = \frac{TP+TN}{Total}$$

$$\textbf{F - measure} = \frac{2*Recall*Precision}{Recall + Precision}$$

THANK YOU!!!!

Online Student Training for "Artificial Intelligence & Machine Learning"

(4th Feb, 2021 – 17th Mar, 2021)

KNN Classifier

Faculty Trainer

Naw Varsha Pipada

Department of Computer Science & Engineering

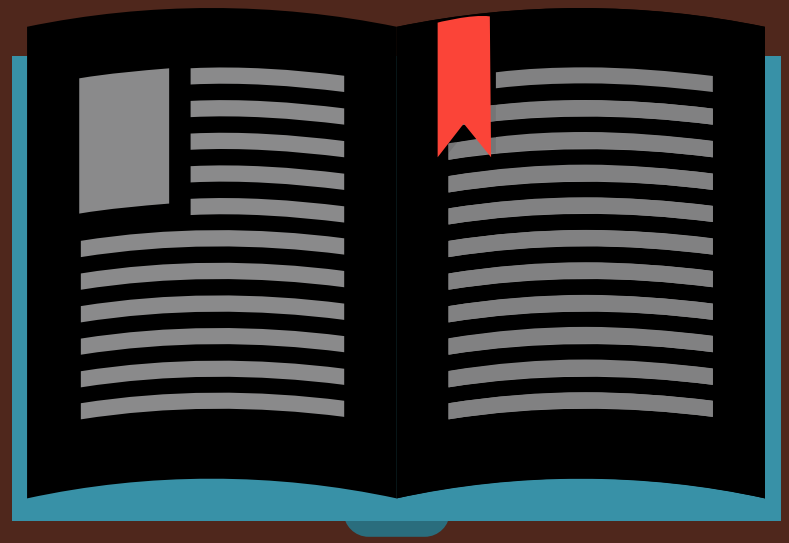
Engineering College Bikaner

Contents

Introduction
Introduction to KNN

How it Works
Basic steps for KNN working

Working Example
An example to understand the concept



Points to Think of
Some issues related to KNN

Pros of KNN
To understand the advantages of KNN

Cons of KNN
To understand the limitations of KNN

KNN Classification

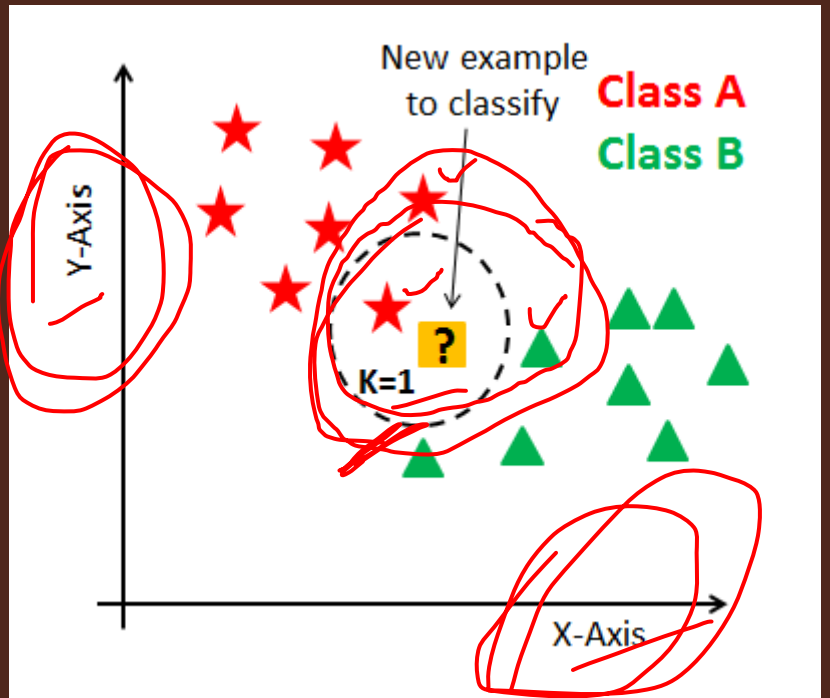
- K Nearest Neighbor(KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms.
- KNN used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition.
- KNN algorithm used for both classification and regression problems

KNN Classification

- KNN is a non-parametric and lazy learning algorithm.
- Non-parametric
  - means there is no assumption for underlying data distribution.
  - the model structure determined from the dataset.
  - This will be very helpful in practice where most of the real world datasets do not follow mathematical theoretical assumptions.
- Lazy algorithm
  - means it does not need any training data points for model generation.
  - All training data used in the testing phase.
  - This makes training faster and testing phase slower and costlier.

How it works

- K is the number of nearest neighbors.
  - The number of neighbors is the core deciding factor.
  - K is generally an odd number if the number of classes is 2.
  - When K=1, then the algorithm is known as the nearest neighbor algorithm.

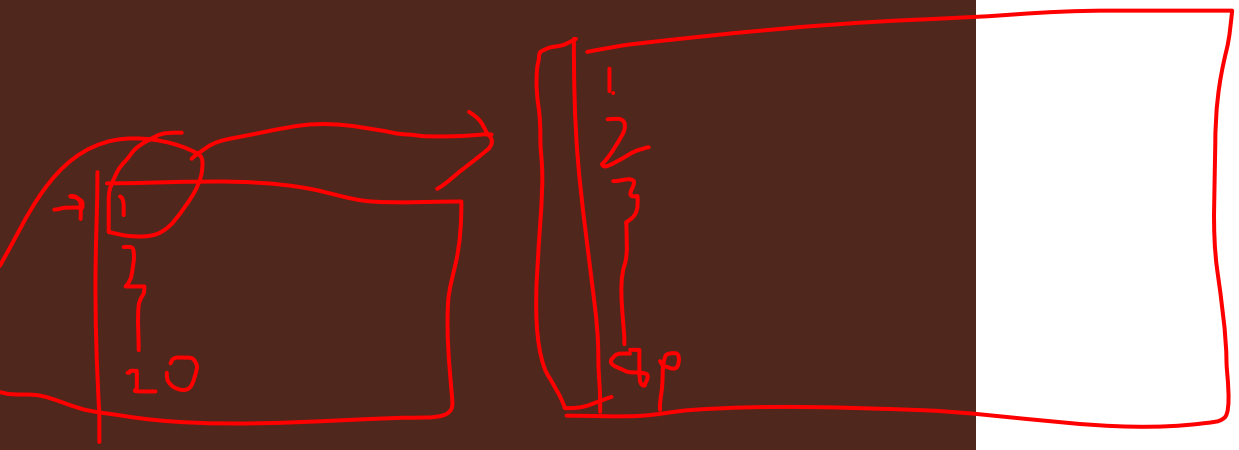


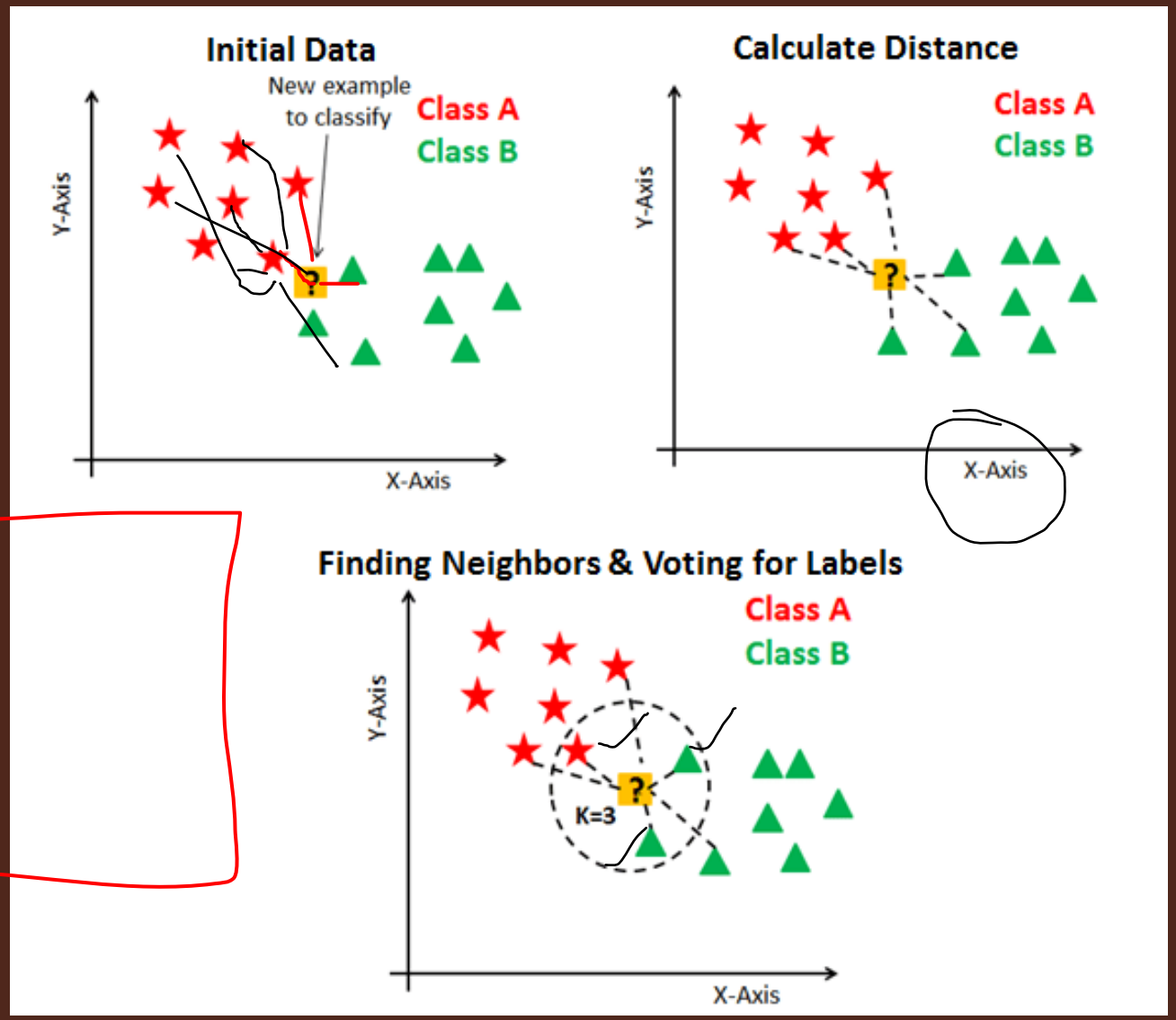
How it works

- Suppose P1 is the point, for which label needs to predict.
- First, you find the k closest point to P1 and then classify points by majority vote of its k neighbors.
- Each object votes for their class and the class with the most votes is taken as the prediction.

Basic Steps

- Calculate distance
- Find closest neighbors
- Vote for labels





Distance Calculation

- Euclidean distance
- Manhattan distance
- Minkowski distance

Euclidean

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

- Hamming distance
  - Generally, used for categorical features

$$D_H = \sum_{i=1}^{k}|x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

Basic KNN Pseudocode

1. Load the data
2. Initialize the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
   a. Calculate the distance between test data and each row of training data.
   b. Sort the calculated distances in ascending order based on distance values
   c. Get the labels(target) of the top K entries
   d. If regression, return the mean of the K labels
   e. If classification, return the mode of the K labels

Working Example

S.No	X1	X2	Y
1	7	7	Bad
2	7	4	Bad
3	3	4	Good
4	1	4	Good

1. Calculate Distance

S.No	X1	X2	Squared Euclidean Distance
1	7	7	$(7-3)^2+(7-7)^2=16$
2	7	4	$(7-3)^2+(4-7)^2=25$
3	3	4	$(3-3)^2+(4-7)^2=9$
4	1	4	$(1-3)^2+(4-7)^2=13$

2. Rank acc. to distance

S.No	X1	X2	Y
3	3	4	Good
4	1	4	Good
1	7	7	Bad
2	7	4	Bad

3. Return mode of K labels

Testing Instance=>
(X1,X2)=(3,7)

Let K=3

Out of top K rows, i.e., 3 rows, the majority class is 'Good'
Hence the test instance is classified as 'Good'

Points to think about

How to choose value of K

Limitations of KNN

Pros of KNN

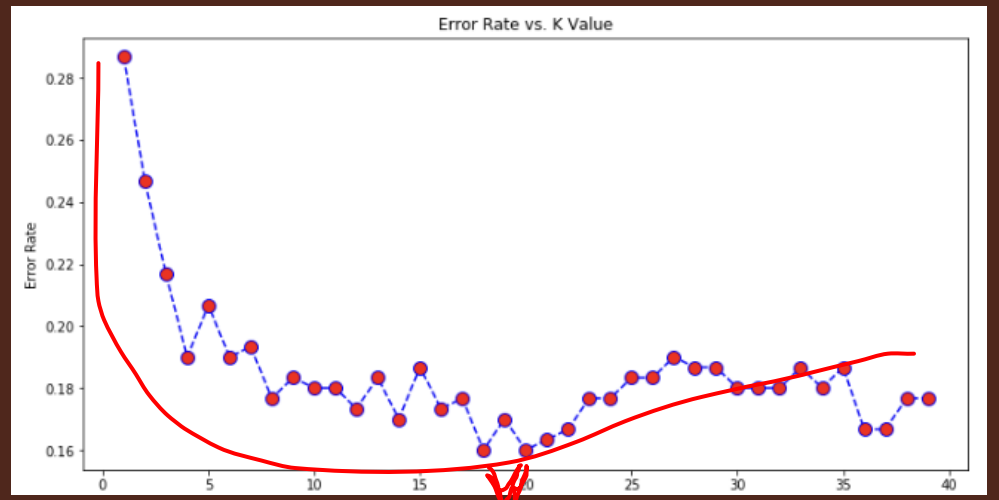
To find solution to limitations

How to choose K: The hyperparameter

- According to research, no optimal number of neighbors suits all kind of data sets. Each dataset has it's own requirements.
- Large k:
  - less sensitive to noise (particularly class noise)
  - better probability estimates for discrete classes
  - larger training sets allow larger values of k
- Small k:
  - captures fine structure of problem space better
  - may be necessary with small training sets
- Balance must be struck between large and small k
- Generally, an odd value of K is chosen for even number of classes.
- You can also check by generating the model on different values of k and check their performance

How to choose K : Elbow Method

- Run KNN classifier on the dataset for a range of values of *k* (say, *k* from 1 to 40)
- For each value of *k* calculate the error rate.
- Then, plot a line chart of the error rate for each value of *k*.
- If the line chart looks like an arm, then the "elbow" on the arm is the value of *k* that is the best or the value of k with least error rate.



KNN algorithm : Pros

- Quick calculation time – works good with small feature set
- Simple algorithm – to interpret
- Versatile – useful for regression and classification
- High accuracy – you do not need to compare with better-supervised learning models

KNN algorithm : Cons

- Accuracy depends on the quality of the data
- With large data, the prediction stage might be slow
- Sensitive to the scale of the data and irrelevant features
- Require high memory – need to store all of the training data
- Given that it stores all of the training, it can be computationally expensive

KNN algorithm : Cons (Sensitive to Scale of data)

- In the given dataset, income will have a much higher influence on the distance calculated
- Methods to improve: Normalization and Min-max scaling

Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	$142,000	?		

Euclidean Distance

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

Use Min-Max Scaling

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	0.61	?	

Standardized Variable

$$X_s = \frac{X - Min}{Max - Min}$$

KNN algorithm : Cons (Irrelevant features or noise)

- Similar to scaling problem, there might be the influence of irrelevant features.
- Feature selection techniques are used before applying KNN.
- Can use weighted distance,
  - large weights => attribute is more important
  - small weights => attribute is less important
  - zero weights => attribute doesn't matter

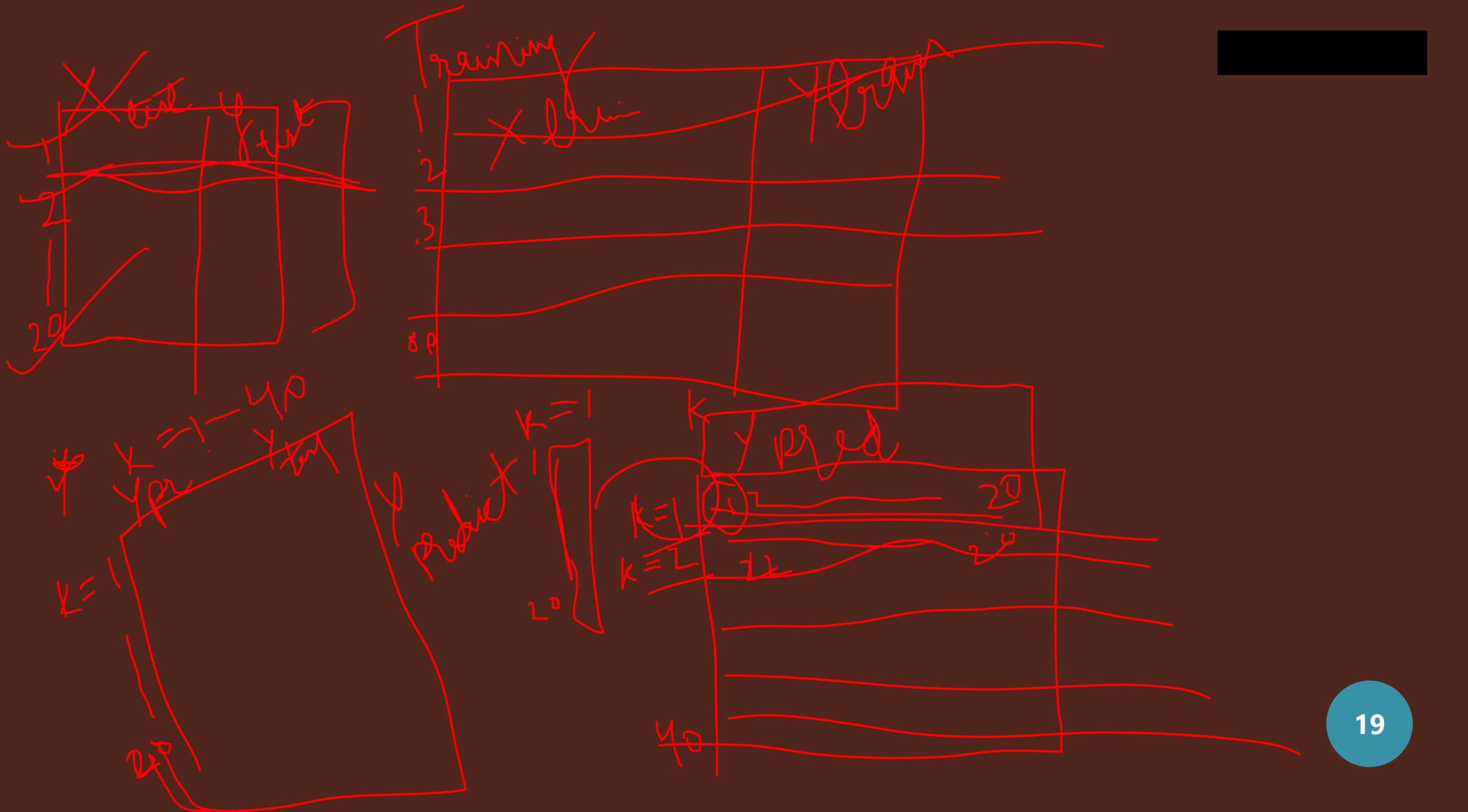
$$D(c1,c2) = \sqrt{\sum_{i=1}^{N} w_i \left(attr_i(c1) - attr_i(c2)\right)^2}$$

KNN algorithm : Cons (Curse of Dimensionality)

- As number of dimensions increases, distance between points becomes larger and more uniform

$$D(c1,c2)=\sqrt{\sum_{i=1}^{relevant}\left(attr_i(c1)-attr_i(c2)\right)^2+\sum_{j=1}^{irrelevant}\left(attr_j(c1)-attr_j(c2)\right)^2}$$

- if number of relevant attributes is fixed, increasing the number of less relevant attributes may swamp distance
- when more irrelevant than relevant dimensions, distance becomes less reliable
- Possible Solutions
  - Feature Extraction
  - PCA to reduce dimensions
  - Large K
  - Weighted KNN – using Kernel Functions
  - More complex distance function


Training
1
2
3
80
20
K=1
40
20
K=1
K=2
20
40

Thank You For Your Time

Naïve Bayes Classifier

A probabilistic Approach

Naw varsha Pipada
Asst. Prof., dept. of CSE
Engineering College Bikaner

Introduction

- Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.
- Naive Bayes model is easy to build and particularly useful for very large data sets.
- Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.
- Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems.

Introduction

- The technique is easiest to understand when described using binary or categorical input values.
- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is a **probabilistic classifier**, which means it predicts on the basis of the probability of an object.

Conditional Probability

- For two events A and B,

$$P(B|A) = P(A \text{ and } B) / P(A)$$

- Which is also equal to

$$P(A|B) = P(A \text{ and } B) / P(B)$$

- From both the equations above,

$$P(A|B) = P(B|A)*P(B) / P(A)$$

Why is it called Naïve Bayes

- **Naïve**: It is called Naïve because it assumes that all features are conditionally independent of one another, that is, the probability of occurrence of a certain feature given the target feature is independent of the probability of occurrence of other features given the target feature.
  - Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Conditional Independence: Example

- Let the two events be the probabilities of persons A and B getting home in time for dinner, and the third event is the fact that a snow storm hit the city.
- While both A and B have a lower probability of getting home in time for dinner, the lower probabilities will still be independent of each other. That is, the knowledge that A is late does not tell you whether B will be late. (They may be living in different neighborhoods, traveling different distances, and using different modes of transportation.)
  - Conditional Probability of A getting home given there will be snow storm can be written as

    $$P(A=getting\_home|Weather=snow\_storm)$$

  - given there will be snow storm can be written as

    $$P(B=getting\_home|Weather=snow\_storm)$$

Conditional Independence: Example

- If the two above two events are independent, then theorem of conditional independence says,

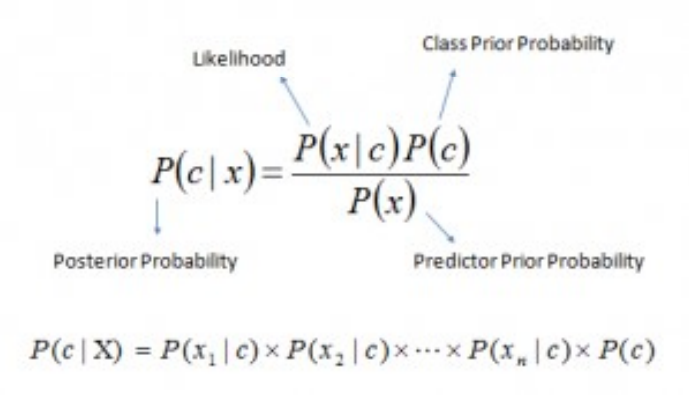
$$P(A=\text{getting\_home and } B=\text{getting\_home}|\text{Weather=snow\_storm}) = P(A=\text{getting\_home}|\text{Weather=snow\_storm}) * P(B=\text{getting\_home}|\text{Weather=snow\_storm}$$

- Similarly , if x_1 ,x_2 ….x_n are the input features and Y is the target feature and the input features are conditionally independent, then

$$P(x_1 \text{ and } x_2 \text{ and } x_3 \ldots . \text{and } x_n|Y) = P(x_1,x_2,x_3,\ldots .x_n|Y) = P(x_1|Y)*P(x_2|Y)*P(x_3|Y)\ldots .P(x_n|Y)$$

Bayes Theorem

- Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:
- Such that



- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor.

How probability used for classification

- After calculating the posterior probability for a number of different classes, you can select the class with the highest probability.
- This is the maximum probable class and may formally be called the maximum a posteriori (MAP) hypothesis.
- This can be written as:

$$MAP(h) = max(P(c|x))$$

or

$$MAP(h) = max((P(x|c) * P(c)) / P(x))$$

How probability used for classification

- The P(x) is a normalizing term which allows us to calculate the probability.
- We can drop it when we are interested in the most probable hypothesis as it is constant and only used to normalize.

$$MAP(h) = max((P(x|c) * P(c)))$$

- Also, if we have an even number of instances in each class in our training data, then the probability of each class (e.g. P(c)) will be equal.
- Again, this would be a constant term in our equation and we could drop it so that we end up with:

$$MAP(c) = max(P(x|c))$$

How it works

- Step 1: Convert the data set into a frequency table
- Step 2: Create Likelihood table by finding the probabilities
- Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Example

- Below is the frequency table for a given dataset,

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

- How much is the probability, that the test fruit is banana? 500/1000=0.5
- How much is the probability, that the test fruit is long? 500/1000=0.5
- How much is the probability, that the test fruit is long given that it is Banana? 400/500=0.8
- How much is the probability, that the test fruit is Banana given that it is long, sweet and yellow? P(Y|X)= ?

Example

- The possible input features for given example are Shape(x1),Taste(x2) and Color(x3)
- And Target Feature is Type(Y)

Example: 1. Compute the 'Prior' probabilities

- P(Y=Banana) = 500 / 1000 = 0.50
- P(Y=Orange) = 300 / 1000 = 0.30
- P(Y=Other) = 200 / 1000 = 0.20

Example: 2. Compute the probability of predictors

- P(x1=Long) = 500 / 1000 = 0.50
- P(x2=Sweet) = 650 / 1000 = 0.65
- P(x3=Yellow) = 800 / 1000 = 0.80

Example: 3. Compute the probability of likelihood

- **Probability of Likelihood for Banana**
- P(x1=Long | Y=Banana) = 400 / 500 = 0.80
- P(x2=Sweet | Y=Banana) = 350 / 500 = 0.70
- P(x3=Yellow | Y=Banana) = 450 / 500 = 0.90

- **Probability of Likelihood for Orange**
- P(x1=Long | Y=Orange) = 0 / 300 = 0
- No need to calculate others

- **Probability of Likelihood for Other**
- P(x1=Long | Y=Other) = 100 / 200 = 0.5
- P(x2=Sweet | Y=Other) = 150 / 200 = 0.75
- P(x3=Yellow | Y=Other) = 50 / 200 = 0.25

Example: 4. Substitute the values in Bayes Theorem

- Banana

$$P(\frac{Banana}{Long, Sweet, Yellow}) = \frac{P(\frac{Long}{Banana}) \times P(\frac{Sweet}{Banana}) \times P(\frac{Yellow}{Banana}) \times P(Banana)}{P(Long)\,P(Sweet)\,P(Yellow)}$$

$$P(\frac{Banana}{Long, Sweet, Yellow}) = \frac{(0.8) \times (0.7) \times (0.9) \times (0.5)}{0.25 \times 0.33 \times 0.41}$$

$$P(\frac{Banana}{Long, Sweet, Yellow}) = 0.252$$

- Orange

$$P(\frac{Orange}{Long, Sweet, Yellow}) = 0$$

$$P(\frac{Other}{Long, Sweet, Yellow}) = \frac{P(\frac{Long}{Other}) \times P(\frac{Sweet}{Other}) \times P(\frac{Yellow}{Other}) \times P(Other)}{P(Long)\,P(Sweet)\,P(Yellow)}$$

- $P(\frac{Other}{Long, Sweet, Yellow}) = \frac{(0.5) \times (0.75) \times (0.25) \times (0.2)}{0.25 \times 0.33 \times 0.41}$

$$P(\frac{Other}{Long, Sweet, Yellow}) = 0.01875$$

In this case, based on the higher score (0.252 for banana) we can assume this Long, Sweet and Yellow fruit is in fact, a Banana

Types of Naïve Bayes Model

- **Gaussian**
  - The Gaussian model assumes that features follow a normal distribution.
  - This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**
  - The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed.
  - It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc.
  - The classifier uses the frequency of words for the predictors.
- **Bernoulli**
  - The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables.
  - Such as if a particular word is present or not in a document.
  - This model is also famous for document classification tasks.

Pros

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency".
  - To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent
- Require to **remove correlated features** because they are voted twice in the model and it can lead to **over inflating importance**.

Example of Smoothing

- Let us consider that test case has Medium long as one of the value for Shape attribute. Then
- P(Shape=Medium_Long|Y=Banana) = P(Shape=Medium_Long and Y=Banana)/P(Y=Banana) = count(Shape=Medium_Long in class Y=Banana)/count(Y=Banana)
- But, since medium long value was not in training data set, so P(Shape=Medium_Long and Y=Banana) will be 0.
- To consider this,
- P(Shape=Medium_Long|Y=Banana) given by

$$\frac{\text{count(Shape=Medium\_Long in class Y=Banana)}+\alpha}{\text{count(Y=Banana)}+\text{number\_of\_features}*\alpha}$$

- Where α is the smoothing parameter, which is generally set as 1 to remove this zero probability problem

Thank You!!!!

Online Student Training for "Artificial Intelligence & Machine Learning"

(4th Feb, 2021 – 17th Mar, 2021)

LOGISTIC REGRESSION

Another probabilistic approach for classification

Faculty Trainer

Naw Varsha Pipada

Department of Computer Science & Engineering

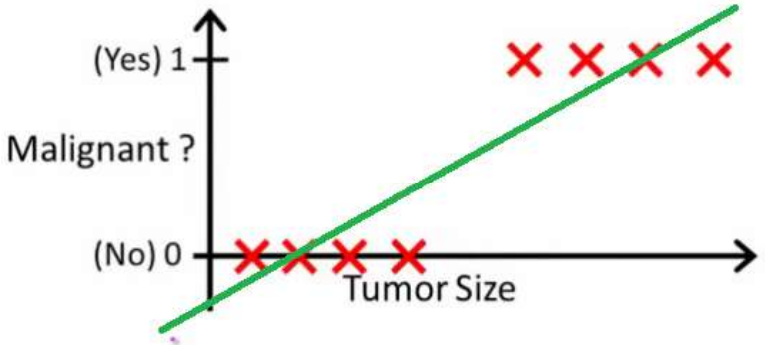
Engineering College Bikaner

CONTENTS

- Classification with Linear Regression
- Logistic Regression
- Types of Logistic Regression
- Prediction Function
- Cost Function

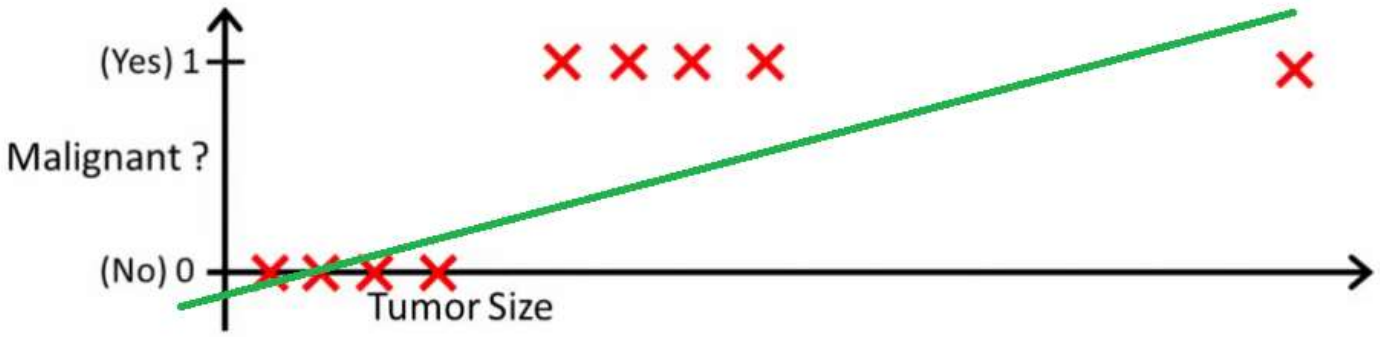
LINEAR REGRESSION FOR CLASSIFICATION?

- The example below we're fitting a straight line through *{tumor size, tumor type}* sample set
- Above, malignant tumors get 1 and non-malignant ones get 0, and the green line is our hypothesis h(x) or regression line.
- To make predictions we may say that for any given tumor size x, if h(x) gets bigger than 0.5 we predict malignant tumor, otherwise we predict benign.



LINEAR REGRESSION FOR CLASSIFICATION?

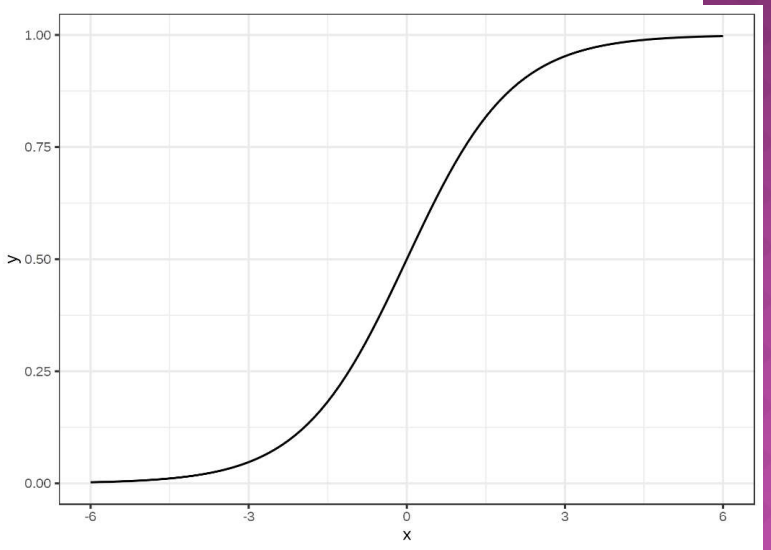
- After adding another sample with a huge tumor size and running linear regression again, $h(x) > 0.5 \rightarrow$ malignant doesn't work anymore. To keep making correct predictions we need to change it to $h(x)>0.2$ or something - but that not how the algorithm should work.
- We cannot change the hypothesis each time a new sample arrives



LOGISTIC REGRESSION

- A solution for classification is logistic regression.
- Instead of fitting a straight line, the logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1.
- The logistic function is defined as:

$$\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)}$$



LINEAR TO LOGISTIC REGRESSION

- The step from linear regression to logistic regression is kind of straightforward. In the linear regression model, we have modelled the relationship between outcome and features with a linear equation:

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}$$

- For classification, we prefer probabilities between 0 and 1, so we wrap the right side of the equation into the logistic function. This forces the output to assume only values between 0 and 1.

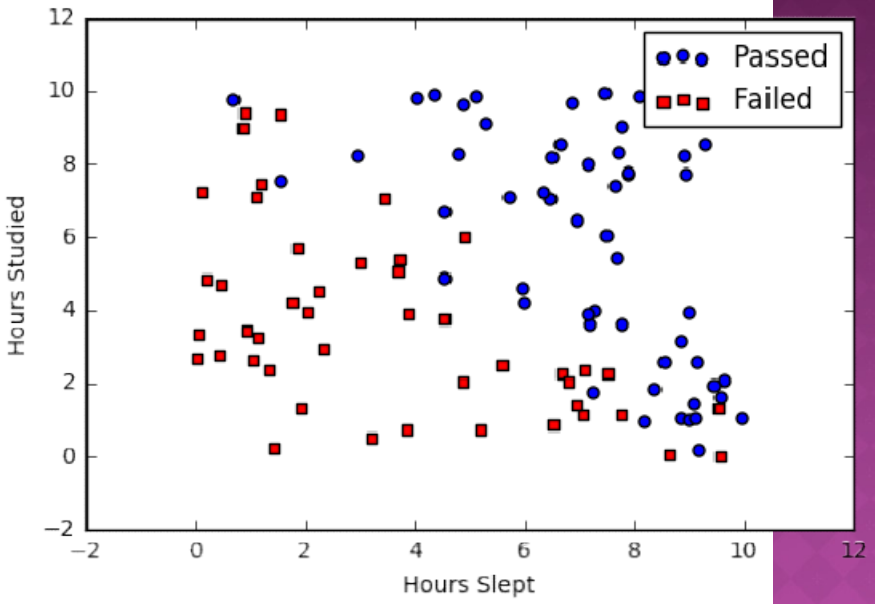
$$P(y^{(i)} = 1) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}))}$$

TYPES OF LOGISTIC REGRESSION

- Binary
- Multiclass
- Ordinal

EXAMPLE

Studied	Slept	Passed
4.85	9.63	1
8.62	3.23	0
5.43	8.23	1
9.21	6.34	0



EXAMPLE

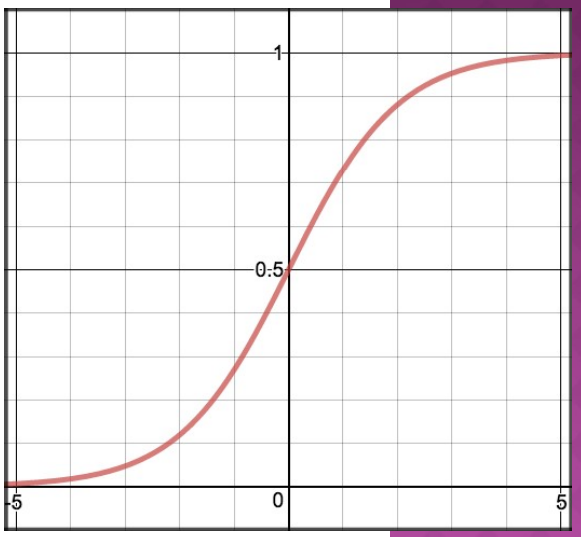
- In order to map predicted values to probabilities, we use the sigmoid function.
- The function maps any real value into another value between 0 and 1.
- In machine learning, we use sigmoid to map predictions to probabilities.

DECISION BOUNDARY

- Our current prediction function returns a probability score between 0 and 1.
- In order to map this to a discrete class, we select a threshold value or tipping point above which we will classify values into class 1 and below which we classify values into class 2.

  $p \geq 0.5, class=1$

  $p < 0.5, class=0$



PREDICTION FUNCTION

- A prediction function in logistic regression returns the probability of our observation being positive, True, or "Yes".
- We call this class 1 and its notation is P(class=1)

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

- For example, $\theta^T x = \theta x^i := \theta_0 + \theta_1 x_1^i + \cdots + \theta_p x_p^i.$
- This is the equation of multiple linear regression

COST FUNCTION

- We can't (or at least shouldn't) use the same cost function Mean Squared Error(MSE) as we did for linear regression because squaring this prediction (logistic function) as we do in MSE results in a non-convex function with many local minimums.
- If our cost function has many local minimums, gradient descent may not find the optimal global minimum
- So, here the cost function used is

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if } y = 1$$

$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if } y = 0$$

COST FUNCTION

- The cost function is compressed to form a single function:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^i \log(h_\theta(x^i)) + (1 - y^i)\log(1 - h_\theta(x^i))$$

- Multiplying by y and (1−y) in the above equation is a sneaky trick that let's us use the same equation to solve for both y=1 and y=0 cases.

MINIMIZE COST FUNCTION : GRADIENT DESCENT

- Finding log values before derivative of cost function,

$$\log h_\theta(x^i) = \log \frac{1}{1+e^{-\theta x^i}} = -\log(1+e^{-\theta x^i}),$$

$$\log(1-h_\theta(x^i)) = \log(1-\frac{1}{1+e^{-\theta x^i}}) = \log(e^{-\theta x^i}) - \log(1+e^{-\theta x^i}) = -\theta x^i - \log(1+e^{-\theta x^i}),$$

- Putting in cost function,

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^i \log(h_\theta(x^i)) + (1-y^i)\log(1-h_\theta(x^i))$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} \left[-y^i(\log(1+e^{-\theta x^i})) + (1-y^i)(-\theta x^i - \log(1+e^{-\theta x^i}))\right]$$

MINIMIZE COST FUNCTION : GRADIENT DESCENT

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i\theta x^i - \theta x^i - \log(1+e^{-\theta x^i})\right] = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i\theta x^i - \log(1+e^{\theta x^i})\right], \quad (*)$$

$$-\theta x^i - \log(1+e^{-\theta x^i}) = -\left[\log e^{\theta x^i} + \log(1+e^{-\theta x^i})\right] = -\log(1+e^{\theta x^i})$$

$$\frac{\partial}{\partial\theta_j} y_i\theta x^i = y_i x_j^i,$$

$$\frac{\partial}{\partial\theta_j}\log(1+e^{\theta x^i}) = \frac{x_j^i e^{\theta x^i}}{1+e^{\theta x^i}} = x_j^i h_\theta(x^i),$$

MINIMIZE COST FUNCTION : GRADIENT DESCENT

- So the value of cost function derivative is,

$$\frac{\partial}{\partial \theta_j} J(\theta) = \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i$$

Remember that the general form of gradient descent is:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

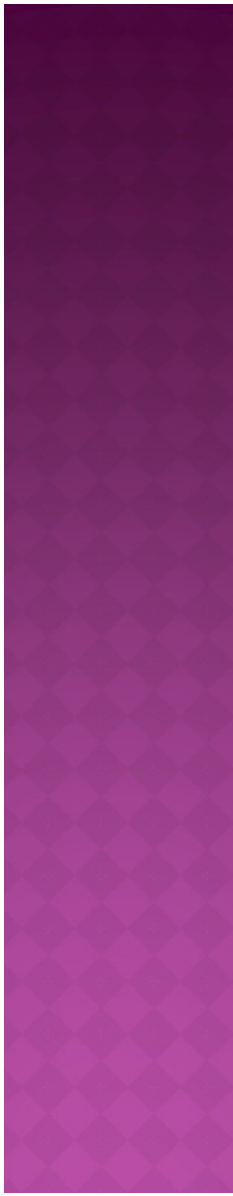
}

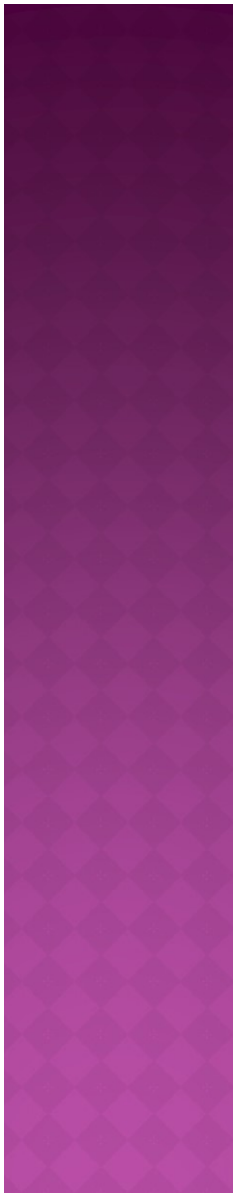
PROS

- Logistic Regression is one of the simplest machine learning algorithms and is easy to implement yet provides great training efficiency in some cases. Also due to these reasons, training a model with this algorithm doesn't require high computation power.
- It makes no assumptions about distributions of classes in feature space.
- This algorithm allows models to be updated easily to reflect new data, unlike decision trees or support vector machines. The update can be done using stochastic gradient descent.

CONS

- On high dimensional datasets, this may lead to the model being over-fit on the training set
- Non linear problems can't be solved with logistic regression since it has a linear decision surface.
- It is difficult to capture complex relationships using logistic regression





THANK YOU!!!!

Online Student Training for "Artificial Intelligence & Machine Learning"

(4th Feb, 2021 – 17th Mar, 2021)

Data Preprocessing in Machine Leaning

Faculty Trainer

Naw Varsha Pipada

Asst. Prof.

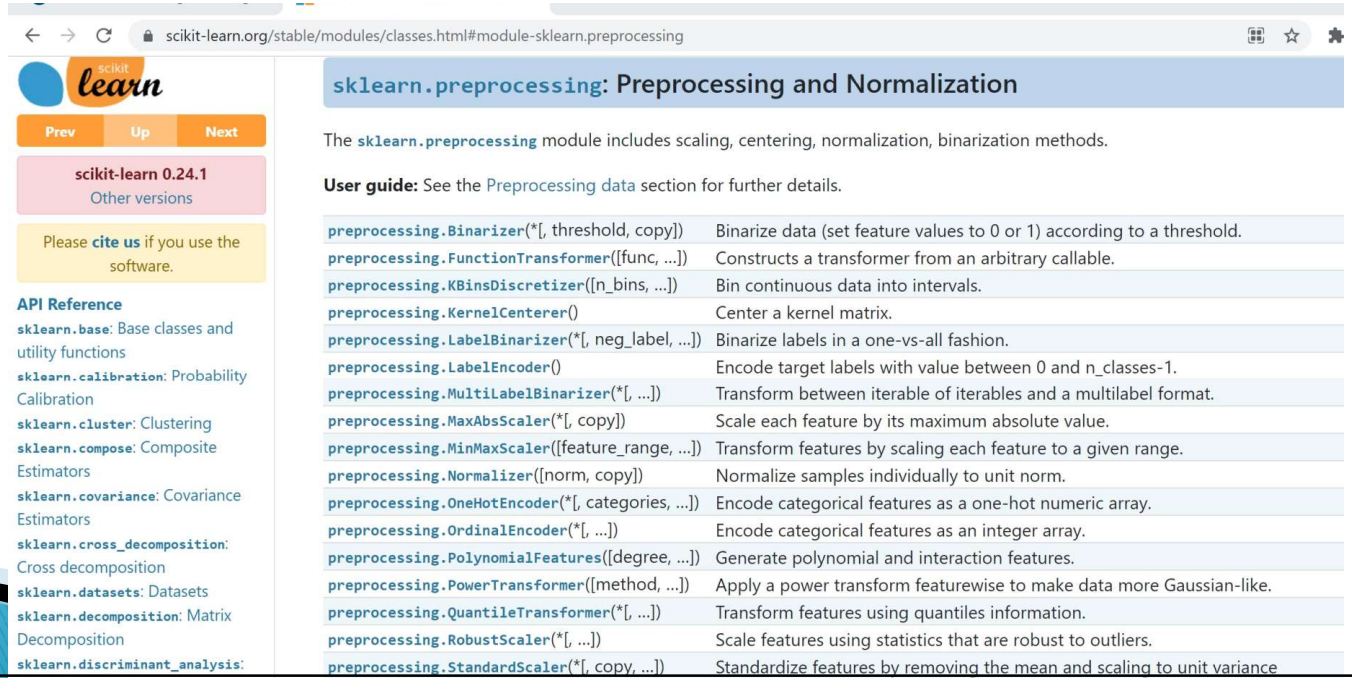
Department of Computer Science & Engineering

Contents

- What is Data Preprocessing
- What's the need
- Steps for Data Preprocesing
- Data Preprocessing Techniques

Data Preprocessing

- https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing



Data Preprocessing

- It involves transforming raw data into an understandable format.
- Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends and is likely to contain many errors.
- Data Preprocessing is a proven method of resolving such issues.
- We should assess the quality of data
  - Mismatching in data types
  - Different dimensions of data arrays
  - Mixture of data values
  - Outliers in the dataset
  - Missing Values

Why We Need Data Preprocessing?

- Real-world data tend to be incomplete, noisy, and inconsistent.
- This can lead to a poor quality of collected data and further to a low quality of models built on such data.
- In order to address these issues, data preprocessing provides operations which can organise the data into a proper form for better understanding in machine process.

Why We Need Data Preprocessing?

- Make our database more accurate
  - We eliminate the incorrect or missing values that are there as a result of the human factor or bugs.
- Boost consistency
  - When there are inconsistencies in data or duplicates, it affects the accuracy of the results.
- Make the database more complete
  - We can fill in the attributes that are missing if needed.
- Smooth the data
  - This way we make it easier to use and interpret.

Steps in Data Preprocessing

- 1. Import libraries
- 2. Read data
- 3. Checking for missing values
- 4. Checking for categorical data
- 5. Standardize the data
- 6. PCA transformation
- 7. Data splitting

Data Preprocessing Techniques

- Data Cleaning/Cleansing
- Data Transformation
- Data Reduction

Data Cleaning

- Handling Missing data
- Handling Noisy Data

Missing Data: Handling Null Values

- In any real-world dataset there are always few null values.
- No model can handle these NULL or NaN values on its own so we need to intervene.
- To check whether we have null values in our dataset or not, isnull() method can be used

```
df.isnull()
# Returns a boolean matrix, if the value is NaN then True otherwise
False


df.isnull().sum()
# Returns the column names along with the number of NaN values in
that particular column
```

Handling Null Values

- Dropping Rows and Columns with Null Values
- Imputation

Handling Null Values

- 1. By dropping the rows or columns containing null values, dropna() method can be used

```
df.dropna()
```

- Various parameters of this function are:
  - axis
    - We can specify axis=0 if we want to remove the rows and axis=1 if we want to remove the columns.
  - how
    - If we specify how = 'all' then the rows and columns will only be dropped if all the values are NaN.By default how is set to 'any'.
  - thresh
    - It determines the threshold value so if we specify thresh=5 then the rows having less than 5 real values will be dropped.
  - subset
    - If we have 4 columns A, B, C and D then if we specify subset=['C'] then only the rows that have their C value as NaN will be removed.
  - inplace
    - By default, no changes will be made to your dataframe. So if you want these changes to reflect onto your dataframe then you need to use inplace = True.

Handling Null Values

- 2. Imputation
  - Imputation is simply the process of substituting the missing values of our dataset.
  - We can do this by defining our own customised function or we can simply perform imputation by using the **SimpleImputer** class provided by sklearn.

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer = imputer.fit(df[['Weight']])
df['Weight'] = imputer.transform(df[['Weight']])
```

Handling Null Values

- Imputer class takes in a few parameters
  - missing_values
    - This is the value which has to be replaced in the dataset. This could either be an integer, or NaN. NaN will be the default value.
  - strategy
    - This is the strategy we'll be using to calculate the value which has to replace the NaN occurrences in the dataset. There are three different strategies we can use mean, median and most_frequent. "mean" is the default value here.
  - axis
    - This can take one of two values — 0 and 1. This will decide if the Imputer will apply the strategy along the rows or along the columns. 0 for columns, and 1 for rows.
  - verbose
    - This will just decide the verbosity of the Imputer. By default, it's set to 0.
  - copy
    - This will decide if a copy of the original object has to be made, or if the Imputer should change the dataset in-place. By default, it is set to True.

Handling Null Values

- Some other methods can also be used to impute like fillna() method

```
df.fillna(df.mean())
```

Noisy Data

- A large amount of additional meaningless data is called *noise.*
  - duplicates or semi-duplicates of the data records;
  - data segments, which have no value for a particular research;
  - unnecessary information fields for each of the variables.
- Possible solution methods
  - Binning
  - Regression
  - Clustering

Data Transformation

- Data transformation consists of the methods of turning the data into an appropriate format for the computer to learn from.
  - Encoding Categorical Variables
  - Feature Scaling
  - Discretization
  - Generalization

Encoding Categorical Variables

- Sometimes our data is in qualitative form, that is we have texts as our data. We can find categories in text form
- To encode such categorical variables, *LabelEncoder* library can be used.

```
from sklearn.preprocessing import LabelEncoder

labelencoder_X = LabelEncoder()

X[:,0] = labelencoder_X.fit_transform(X[:,0])
```

Difficulty saving money	Counts	Frequencies
Very	231	46%
Somewhat	196	39%
Not very	58	12%
Not at all	14	3%
Not sure	1	~0%
Total	500	100%

Example: map() method

```
1  import pandas as pd
2  df_cat = pd.DataFrame(data =
3                        [['green','M',10.1,'class1'],
4                         ['blue','L',20.1,'class2'],
5                         ['white','M',30.1,'class1']])
6  df_cat.columns = ['color','size','price','classlabel']
7  print(df_cat)
8  size_mapping = {'M':1,'L':2}
9  df_cat['size'] = df_cat['size'].map(size_mapping)
10 print(df_cat)
```

```
   color size  price classlabel
0  green    M   10.1     class1
1   blue    L   20.1     class2
2  white    M   30.1     class1
   color  size  price classlabel
0  green     1   10.1     class1
1   blue     2   20.1     class2
2  white     1   30.1     class1
```

Example – LabelEncoder()

```
1  import pandas as pd
2  df_cat = pd.DataFrame(data =
3                        [['green','M',10.1,'class1'],
4                         ['blue','L',20.1,'class2'],
5                         ['white','M',30.1,'class1']])
6  df_cat.columns = ['color','size','price','classlabel']
7  print(df_cat)
8  size_mapping = {'M':1,'L':2}
9  df_cat['size'] = df_cat['size'].map(size_mapping)
10 print(df_cat)
11 from sklearn.preprocessing import LabelEncoder
12 class_le = LabelEncoder()
13 df_cat['classlabel'] = class_le.fit_transform(df_cat['classlabel'].values)
14 print(df_cat)
```

```
   color size  price classlabel
0  green    M   10.1     class1
1   blue    L   20.1     class2
2  white    M   30.1     class1
   color  size  price classlabel
0  green     1   10.1     class1
1   blue     2   20.1     class2
2  white     1   30.1     class1
   color  size  price  classlabel
0  green     1   10.1           0
1   blue     2   20.1           1
2  white     1   30.1           0
```

Feature Scaling

- Normalization
  - Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1.
  - It is also known as Min-Max scaling.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

  - Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

Feature Scaling : Example

```
# data normalization with sklearn
from sklearn.preprocessing import MinMaxScaler

# fit scaler on training data
norm = MinMaxScaler().fit(X_train)

# transform training data
X_train_norm = norm.transform(X_train)

# transform testing dataabs
X_test_norm = norm.transform(X_test)
```

Feature Scaling

- Standardization
  - Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation.
  - This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

$$X' = \frac{X - \mu}{\sigma}$$

- μ is the mean of the feature values and σ is the standard deviation of the feature values.

Feature Scaling : Example

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()


X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Discretization & Generalization

- Data discretization is defined as a process of converting continuous data attribute values into a finite set of intervals with minimal loss of information
- Data Generalization is to convert low-level data features to high-level data features. For example, house addresses can be generalized to higher-level definitions, such as town or country. Similarly, the values for numeric attributes may be mapped to higher level concepts like, age into young, middle-aged, or senior

Data Reduction

- Feature Selection
  - Feature Selection is the process of selecting a subset of relevant features for use in model construction
  - Feature selection methods can be used to identify and remove unneeded, irrelevant and redundant attributes from data that do not contribute to the accuracy of a predictive model or may in fact decrease the accuracy of the model.
- Dimensionality Reduction
  - Dimensionality reduction, or dimension reduction, is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension
- Numerosity reduction
  - Numerosity Reduction is a data reduction technique which replaces the original data by smaller form of data representation.

Thank You!!!

Online Student Training for "Artificial Intelligence & Machine Learning"

(4th Feb, 2021 – 17th Mar, 2021)

Random Forest Classifier

Faculty Trainer

Naw Varsha Pipada

Department of Computer Science & Engineering

Engineering College Bikaner

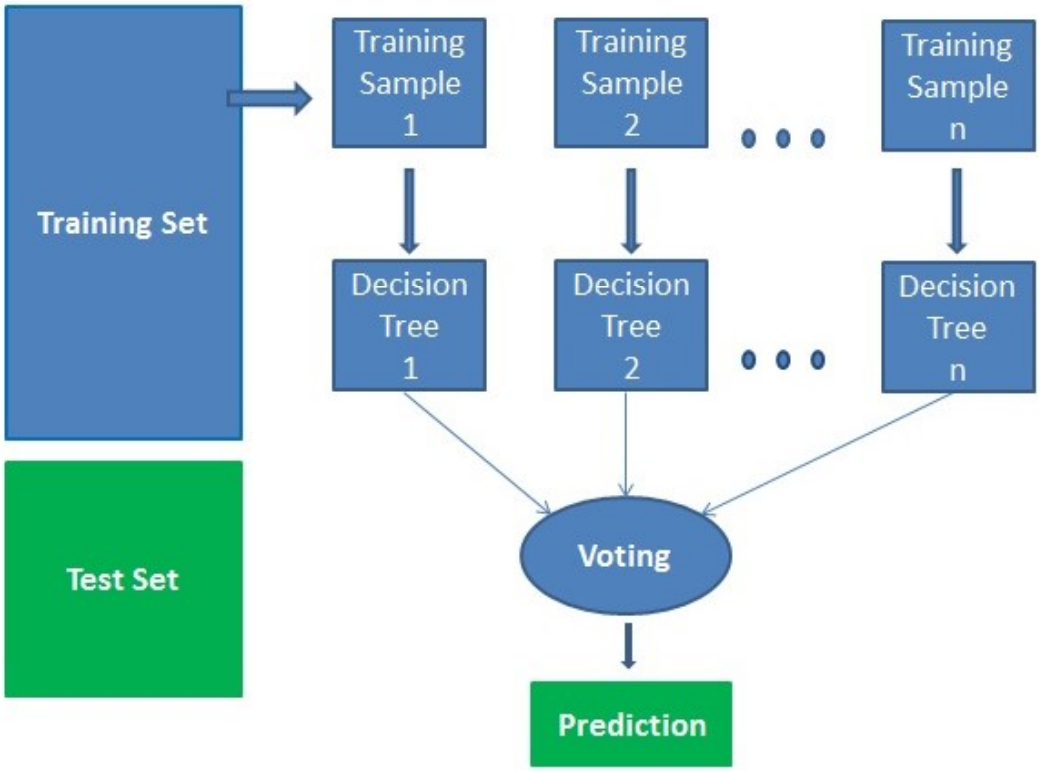
Contents

- Introduction
- How it works
- Pros & Cons
- Random Forest Vs Decision Trees
- Applications

Introduction

- Random forests is a supervised learning algorithm. It can be used both for classification and regression.
- It is also the most flexible and easy to use algorithm.
- A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is.
- Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting.
- It also provides a pretty good indicator of the feature importance.

How it Works


Training Set
Test Set
Training Sample 1
Training Sample 2
Training Sample n
Decision Tree 1
Decision Tree 2
Decision Tree n
Voting
Prediction

How it Works

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.

Pros

- Highly accurate and robust method because of the number of decision trees participating in the process.
- Does not suffer from the overfitting problem.
- The algorithm can be used in both classification and regression problems.
- Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.
- You can get the relative feature importance, which helps in selecting the most contributing features for the classifier.

Cons

- Random forests is slow in generating predictions because it has multiple decision trees.
- The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree.

Random Forests vs Decision Trees

- Random forests is a set of multiple decision trees.
- Deep decision trees may suffer from overfitting, but random forests prevents overfitting by creating trees on random subsets.
- Decision trees are computationally faster.
- Random forests is difficult to interpret, while a decision tree is easily interpretable and can be converted to rules.

Applications

- Recommendation engines
- Image classification
- Feature selection
- Classify loyal loan applicants
- Identify fraudulent activity
- Predict diseases

See You

End of slides

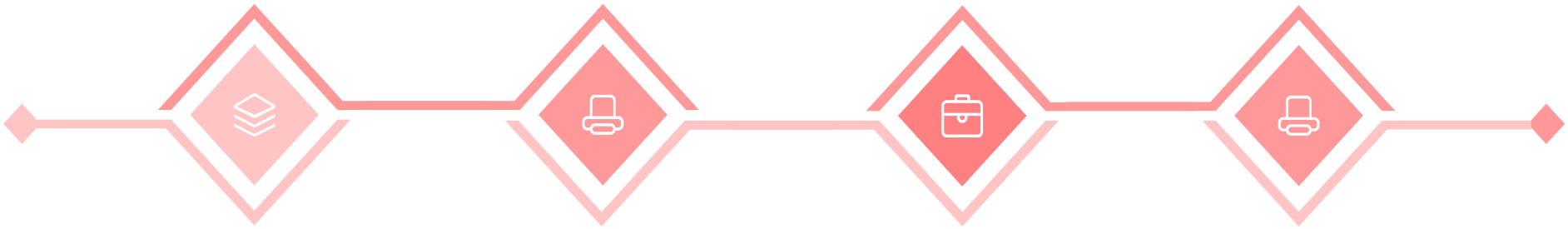
Thank You for Your Attention

Unsupervised Learning

K-means Clustering

Faculty Trainer
Naw Varsha Pipada
Department of Computer Science & Engineering
Engineering College Bikaner

Contents



Unsupervised Learning

Quick Review and its types

Clustering

Introduction and its types

K-means Clustering

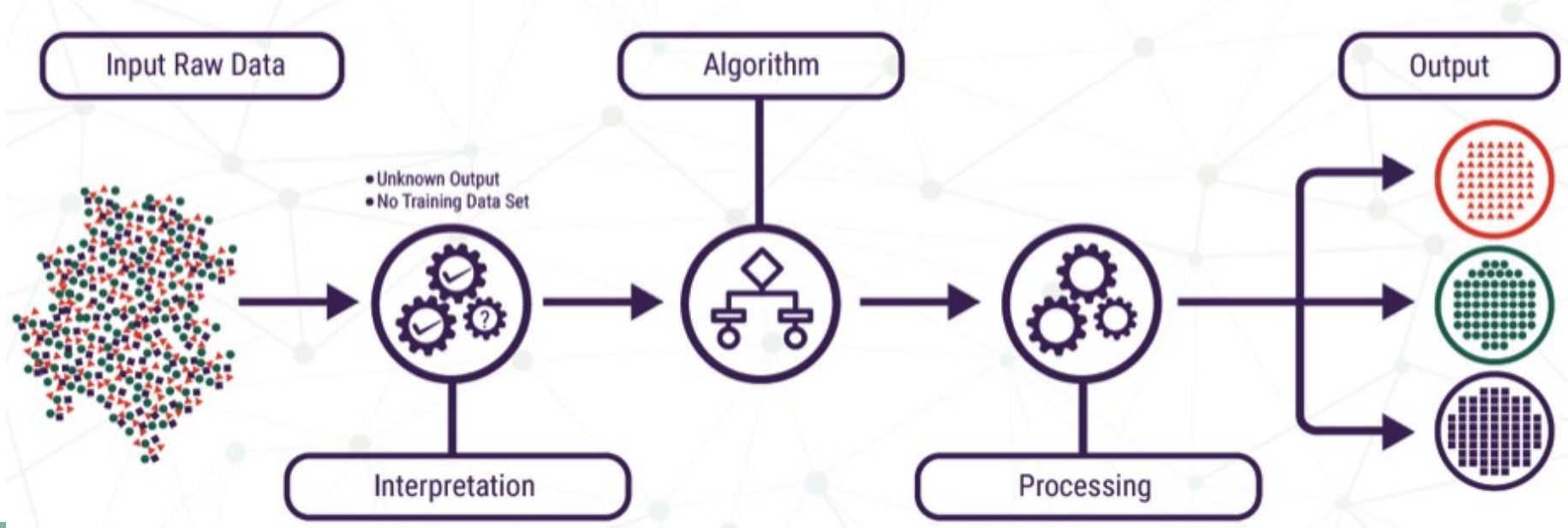
How it works and Algorithm

Pros and Cons

Issues and their solutions

Quick Review

- Typically, unsupervised learning algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.



Why Unsupervised Learning

- Issues

- Unsupervised Learning is harder as compared to Supervised Learning tasks..
- How do we know if results are meaningful since no answer labels are available?
- Let the expert look at the results (external evaluation)
- Define an objective function on clustering (internal evaluation)

- Still Needed

- Annotating large datasets is very costly and hence we can label only a few examples manually. Example: Speech Recognition
- There may be cases where we don't know how many/what classes is the data divided into. Example: Data Mining
- We may want to use clustering to gain some insight into the structure of the data before designing a classifier.

Unsupervised Learning Classification

Parametric Unsupervised Learning

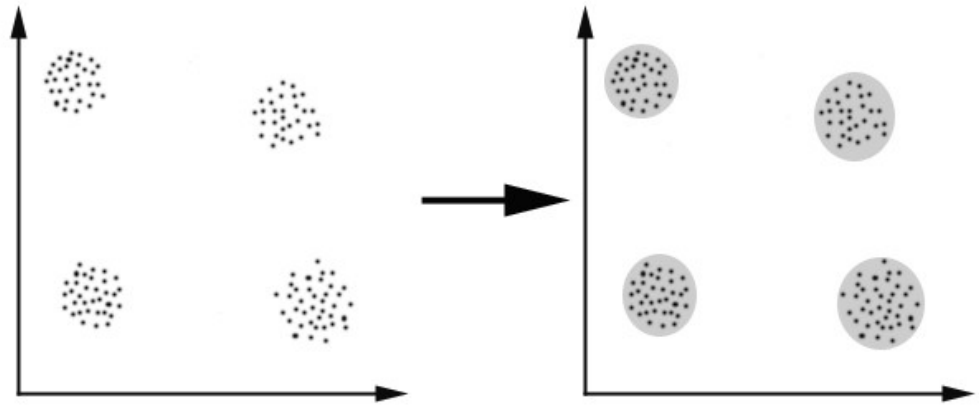
- It assumes that sample data comes from a population that follows a probability distribution based on a fixed set of parameters.
- Examples: Gaussian Mixture Models, Expectation-Maximization Algorithm, Probabilistic Clustering

Non-Parametric Unsupervised Learning

- Data is grouped into clusters, where each cluster(hopefully) says something about categories and classes present in the data
- Sometimes referred to as a distribution-free method
- Examples: K-means Clustering, Hierarchical Clustering, Association Rule Mining

Clustering

- It deals with finding a *structure* in a collection of unlabeled data.
- A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some way”.
- A *cluster* is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.



Types of Clustering

- K-means Clustering
- Hierarchical Clustering
- Density Based Clustering
- Probabilistic Clustering

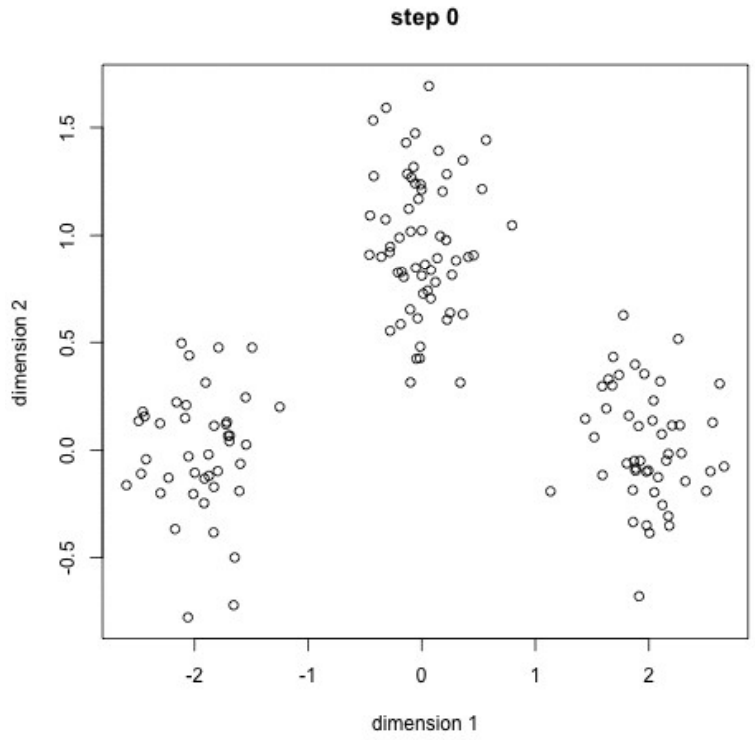
K-means Clustering

- The *'means'* in the K-means refers to averaging of the data; that is, finding the centroid.
- A centroid is the imaginary or real location representing the center of the cluster.
- *K* refers to the number of centroids you need in the dataset.

K-means Clustering – How it works

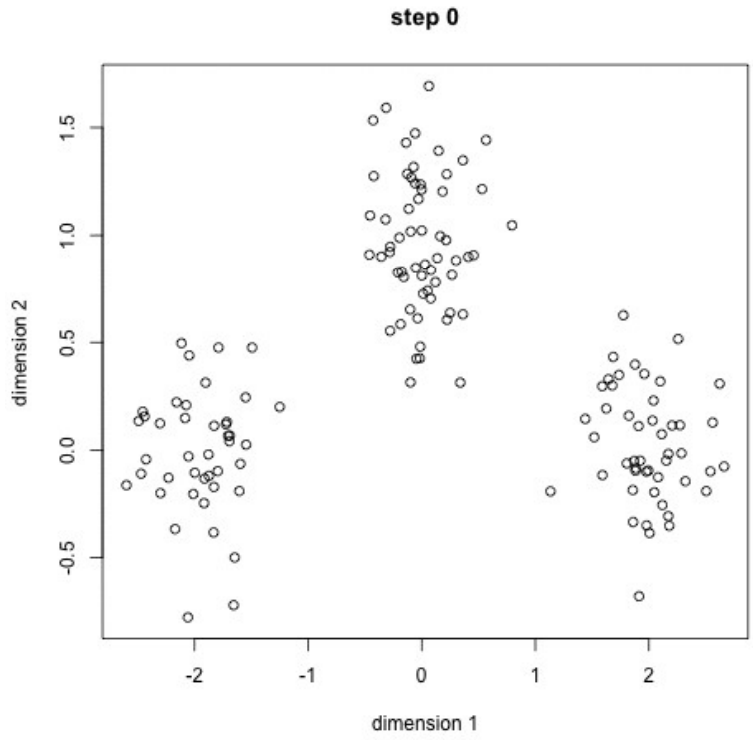
- Assume K=3, before starting to cluster.
- Algorithm:

1. Choose k (random) data points (seeds) to be the initial centroids
2. Assign each data point to the closest centroid, i.e., having minimum distance.



K-means Clustering - Illustration

4. Re-compute the centroids using the current cluster memberships, that is, by taking mean of all data points.
5. If convergence criteria is not met Go to step 2 else stop.



K-means Clustering: Convergence Criteria

- K-means performs iterative (repetitive) calculations to optimize the positions of the centroids
- It halts creating and optimizing clusters when either of the below criteria met:
  - The centroids have stabilized — there is no change in their values because the clustering has been successful.
  - The defined number of iterations has been achieved.

K-means Clustering: Convergence Criteria

- In other words, the iterations can be stopped if:

1. defined number of iterations has been achieved
2. no (or minimum) re-assignments of data points to different clusters, or
3. no (or minimum) change of centroids, or
4. minimum decrease in the sum of squared error

Distance (Dissimilarity) Measures

- Euclidean Distance

$$Dist_{xy} = \sqrt{\sum_{k=1}^{m}(x_{ik} - y_{ik})^2}$$

- Manhattan(City Block)

$$Dist_{xy} = \sum_{k=1}^{m}|x_{ik} - y_{ik}|$$

- Minkowski Distance

$$Dist_{xy} = \left(\sum_{k=1}^{d}|x_{ik} - x_{jk}|^{\frac{1}{p}}\right)^{p}$$

- Chebychev Distance

$$Dist_{xy} = max_k\ |x_{ik} - y_{ik}|$$

Example

- Cluster the following eight points (with (x, y) representing locations) into three clusters:
  - A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)
- Initial cluster centers are: A1(2, 10), A4(5, 8) and A7(1, 2).
- The distance function between two points a = (x1, y1) and b = (x2, y2) is defined as-
  - $D(a, b) = |x2 – x1| + |y2 – y1|$
- Use K-Means Algorithm to find the three cluster centers after the second iteration.

Calculating Distance Between A1 and Clusters

- D(A1, C1) = |2 – 2| + |10 – 10|= 0
- D(A1, C2) = |5 – 2| + |8 – 10|= 0
- D(A1, C3) = |1 – 2| + |2 – 10|= 0

Similarly for all points

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (5, 8) of Cluster-02	Distance from center (1, 2) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	5	9	C1
A2(2, 5)	5	6	4	C3
A3(8, 4)	12	7	9	C2
A4(5, 8)	5	0	10	C2
A5(7, 5)	10	5	9	C2
A6(6, 4)	10	5	7	C2
A7(1, 2)	9	10	0	C3
A8(4, 9)	3	2	10	C2

Finding new cluster

- For Cluster-01:
  - We have only one point A1(2, 10) in Cluster-01.
  - So, cluster center remains the same.
- For Cluster-02:
  - Center of Cluster-02

    = ((8 + 5 + 7 + 6 + 4)/5, (4 + 8 + 5 + 4 + 9)/5)

    = (6, 6)
- For Cluster-03:
  - Center of Cluster-03

    = ((2 + 1)/2, (5 + 2)/2)

    = (1.5, 3.5)

Iteration 2

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (6, 6) of Cluster-02	Distance from center (1.5, 3.5) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	8	7	C1
A2(2, 5)	5	5	2	C3
A3(8, 4)	12	4	7	C2
A4(5, 8)	5	3	8	C2
A5(7, 5)	10	2	7	C2
A6(6, 4)	10	2	5	C2
A7(1, 2)	9	9	2	C3
A8(4, 9)	3	5	8	C1

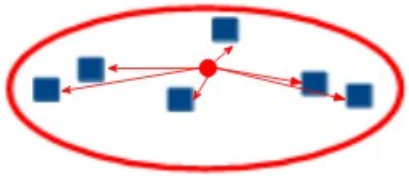
Finding new cluster

- For Cluster-01:
  - Center of Cluster-02
    = ((2 + 4)/2, (10+ 9)/2)
    = (3, 9.5)
- For Cluster-02:
  - Center of Cluster-02
    = ((8 + 5 + 7 + 6)/4, (4 + 8 + 5 + 4)/4)
    = (6.5, 5.25)
- For Cluster-03:
  - Center of Cluster-03
    = ((2 + 1)/2, (5 + 2)/2)
    = (1.5, 3.5)

- So the new clusters after iteration 2 are
- C1(3, 9.5)
- C2(6.5, 5.25)
- C3(1.5, 3.5)

Different Evaluation Metrics for Clustering

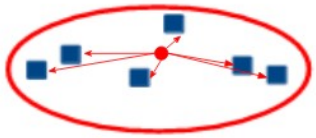
- Inertia
  - calculates the sum of distances of all the points within a cluster from the centroid of that cluster.
  - Also known as intracluster distance



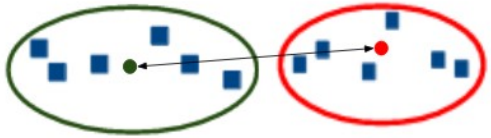
Intra cluster distance

- Dunn Index

$$\text{Dunn Index} = \frac{\min(\text{Inter cluster distance})}{\max(\text{Intra cluster distance})}$$

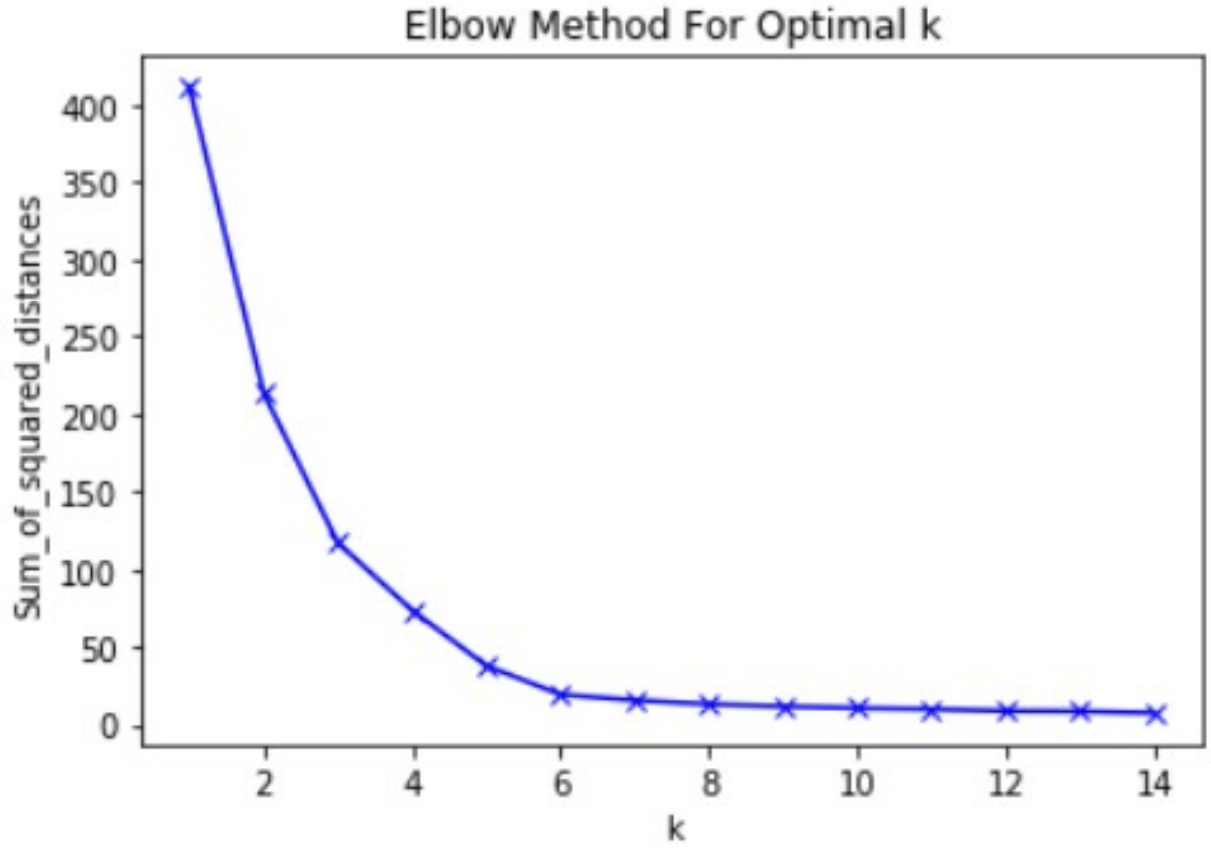


Intra cluster distance



Inter cluster distance

How to find optimum K : Elbow method



Pros

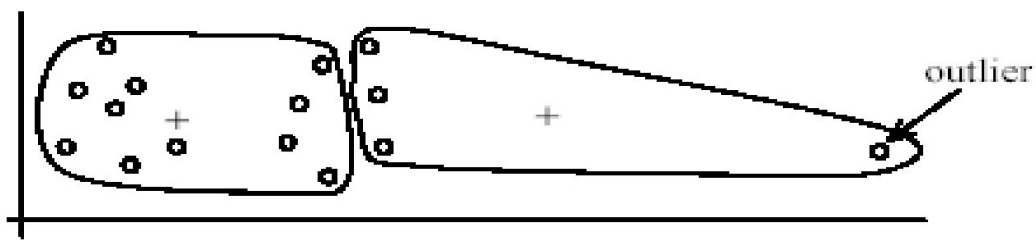
- Simple & easy to understand and to implement
- Efficient: Time complexity: O(tkn),
  - where n is the number of data points,
  - k is the number of clusters, and
  - t is the number of iterations.
- Since both k and t are small. k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.

Cons

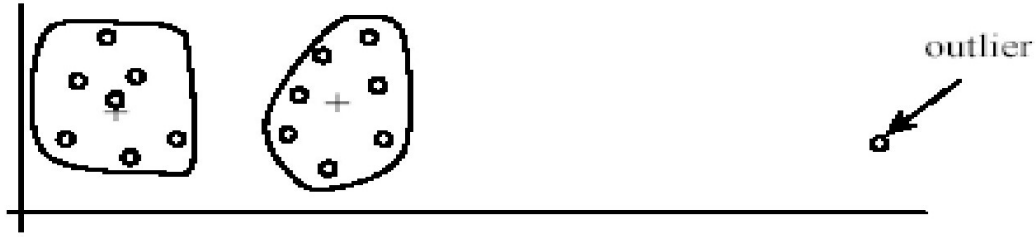
- The algorithm is only applicable if the mean is defined.
  - For categorical data, k-mode - the centroid is represented by most frequent values.
- The user needs to specify k.
- The algorithm is sensitive to outliers
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.
- Sensitive to initially selected random centroids
- Not suitable for hyper-ellipsoids (or hyper-sphere)

Dealing with outliers

- Remove some data points that are much further away from the centroids than other data points
- Perform random sampling: by choosing a small subset of the data points, the chance of selecting an outlier is much smaller

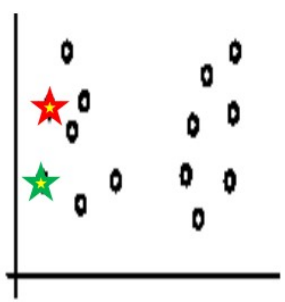


(A): Undesirable clusters

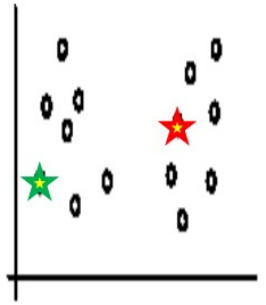


(B): Ideal clusters

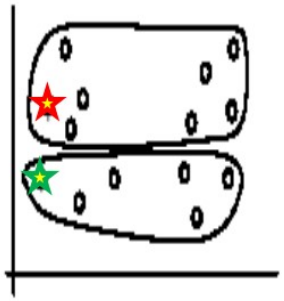
Sensitivity to initial seeds



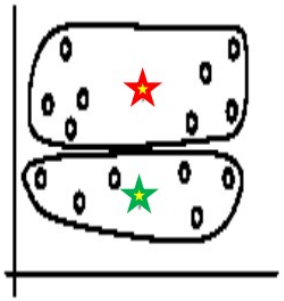
Random selection of seeds (centroids)



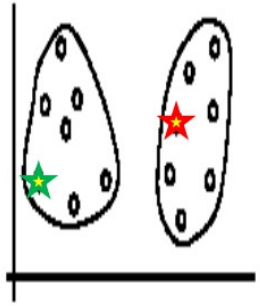
Random selection of seeds (centroids)



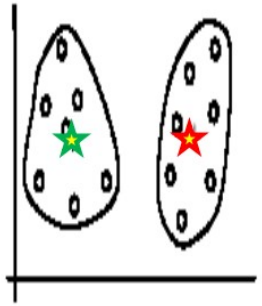
Iteration 1



Iteration 2

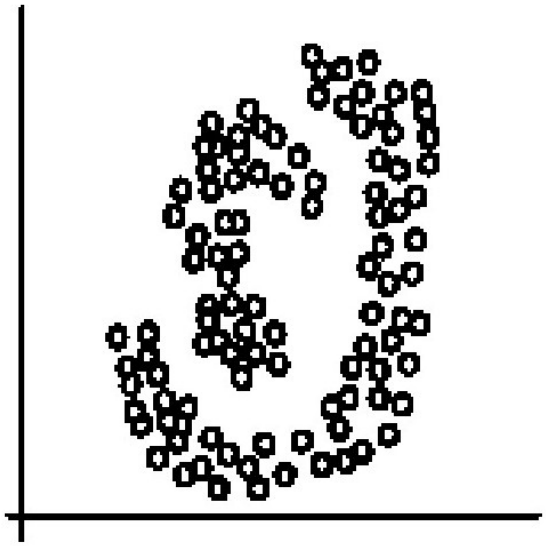


Iteration 1

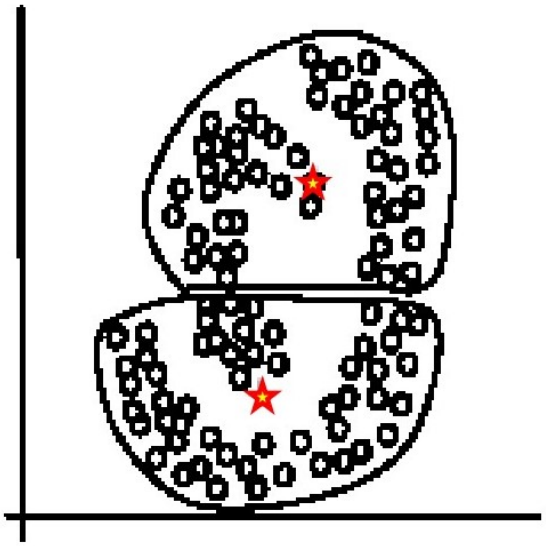


Iteration 2

Not suitable for hyper-ellipsoids(or sphere)



(A): Two natural clusters



(B): *k*-means clusters

THANK YOU!!!!

Scalars, Vectors, Matrices & Tensors

An introduction to Linear Algebra (Part 1)

Contents

- Introduction to Scalars, Vectors, Matrices, Tensors
- Matrix Operations
  - Transposition
  - Addition
  - Dot Product
  - Broadcasting
- System of Linear Equations
- Solution to linear equations with matrices

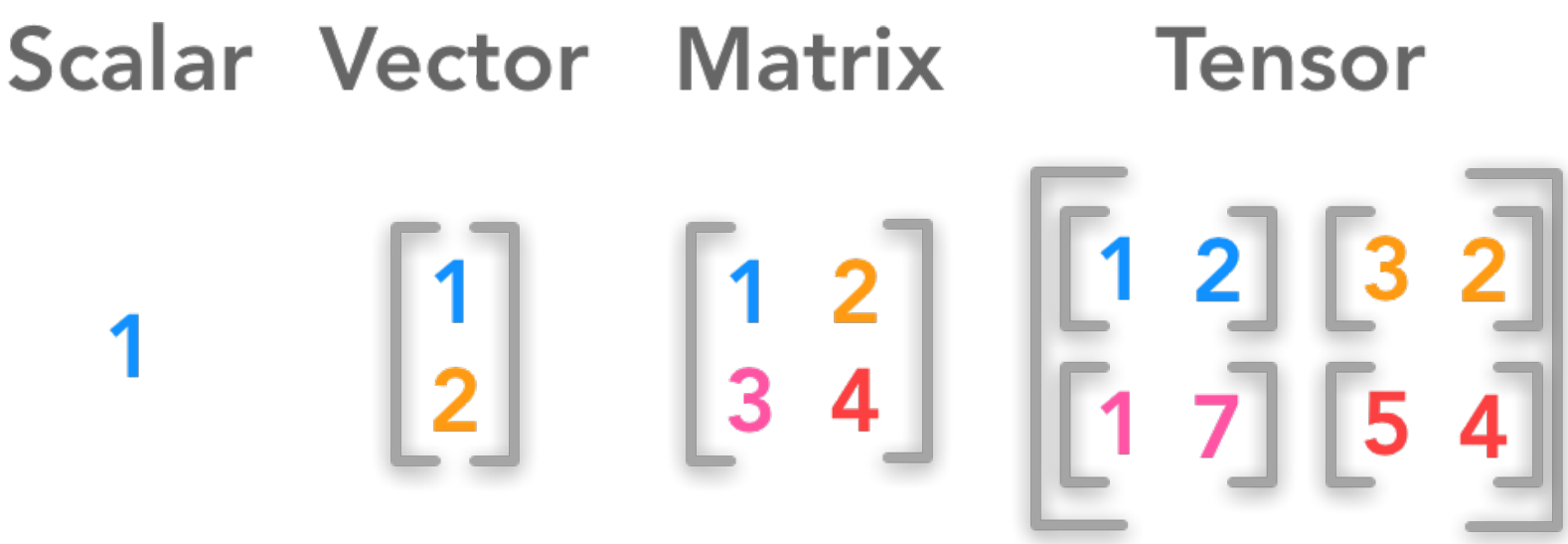
Introduction

- A scalar is a single number
- A vector is an array of numbers.
- A matrix is a 2-D array
- A tensor is a n-dimensional array with n>2

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}$$

$$\boldsymbol{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,n} \end{bmatrix}$$

Introduction



Conventions

- scalars are written in lowercase and italics. For instance: n
- vectors are written in lowercase, italics and bold type. For instance: $\mathbf{x}$
- matrices are written in uppercase, italics and bold. For instance: $\mathbf{X}$

Transposition

The superscript $^{\mathrm{T}}$ is used for transposed matrices.

$$\boldsymbol{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix}$$

$$\boldsymbol{A}^{\mathrm{T}} = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

Vector Transpose

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 2 \end{bmatrix}$$

Matrix Transpose

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

Addition

- Matrices can be added if they have the same shape:

  A+B=C

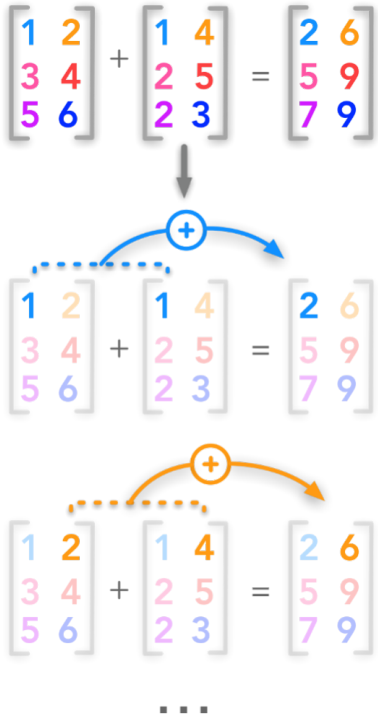
- Each cell of **A** is added to the corresponding cell of **B**:

  $\mathbf{A}_{i,j} + \mathbf{B}_{i,j} = \mathbf{C}_{i,j}$

ii is the row index and jj the column index.

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \\ B_{3,1} & B_{3,2} \end{bmatrix} = \begin{bmatrix} A_{1,1} + B_{1,1} & A_{1,2} + B_{1,2} \\ A_{2,1} + B_{2,1} & A_{2,2} + B_{2,2} \\ A_{3,1} + B_{3,1} & A_{3,2} + B_{3,2} \end{bmatrix}$$

The shape of $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are identical.



Broadcasting

- Numpy can handle operations on arrays of different shapes.
- The smaller array will be extended to match the shape of the bigger one.

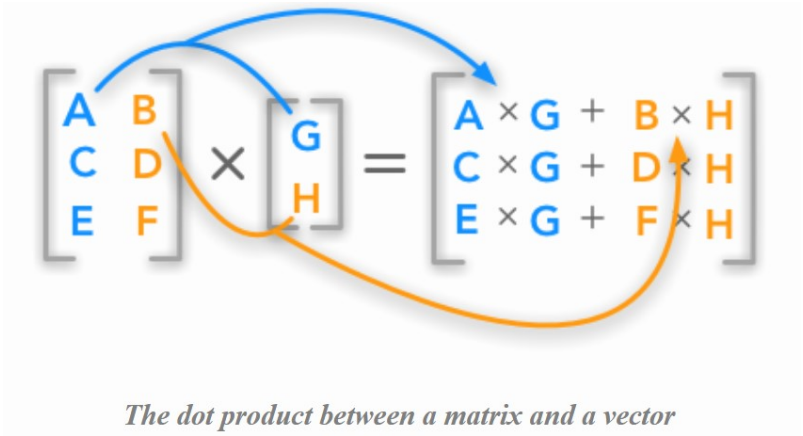
$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} \\ B_{2,1} \\ B_{3,1} \end{bmatrix}$$

is equivalent to

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} + \begin{bmatrix} B_{1,1} & B_{1,1} \\ B_{2,1} & B_{2,1} \\ B_{3,1} & B_{3,1} \end{bmatrix} = \begin{bmatrix} A_{1,1} + B_{1,1} & A_{1,2} + B_{1,1} \\ A_{2,1} + B_{2,1} & A_{2,2} + B_{2,1} \\ A_{3,1} + B_{3,1} & A_{3,2} + B_{3,1} \end{bmatrix}$$

Matrix Multiplication

- The standard way to multiply matrices is not to multiply each element of one with each element of the other (called the *element-wise product*) but to calculate the sum of the products between rows and columns.
- The number of columns of the first matrix must be equal to the number of rows of the second matrix.
- The matrix product, also called **dot product**, is calculated as following:



The dot product between a matrix and a vector

Matrix Multiplication

- The dot product can be formalized through the following equation:

$$C_{i,j} = A_{i,k}B_{k,j} = \sum_k A_{i,k}B_{k,j}$$

- Properties of the dot product:
  - 1. $(\boldsymbol{AB})^{\mathrm{T}} = \boldsymbol{B}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}$
  - 2. Matrix mutliplication is distributive $\boldsymbol{A}(\boldsymbol{B}+\boldsymbol{C}) = \boldsymbol{AB}+\boldsymbol{AC}$
  - 3. Matrix mutliplication is associative $\boldsymbol{A}(\boldsymbol{BC}) = (\boldsymbol{AB})\boldsymbol{C}$
  - 4. Matrix multiplication is not commutative $\boldsymbol{AB} \neq \boldsymbol{BA}$
  - 5. Vector multiplication is commutative $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{x}$

System of linear equations

- A system of equations is a set of multiple equations (at least 1). For instance we could have:

$$\begin{cases} y = 2x + 1 \\ y = \frac{7}{2}x + 3 \end{cases}$$

- It is defined by its number of equations and its number of unknowns.
- In this example, there are 2 equations (the first and the second line) and 2 unknowns (x and y).
- In addition we call this a system of linear equations because each equation is linear.

Using matrices to describe the system

- Matrices can be used to describe a system of linear equations of the form **Ax**=**b**

$$
\begin{aligned}
&A_{1,1}x_1 + A_{1,2}x_2 + A_{1,n}x_n = b_1 \\
&A_{2,1}x_1 + A_{2,2}x_2 + A_{2,n}x_n = b_2 \\
&\cdots \\
&A_{m,1}x_1 + A_{m,2}x_2 + A_{m,n}x_n = b_n
\end{aligned}
$$

- The left-hand side can be considered as the product of a matrix **A** containing weights for each variable (n columns) and each equation (m rows):

Using matrices to describe the system

- The left-hand side can be considered as the product of a matrix **A** containing weights for each variable (n columns) and each equation (m rows):

$$\boldsymbol{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,n} \end{bmatrix}$$

- with a vector **x** containing the n unknowns

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix}$$

- The dot product of **A** and **x** gives a set of equations

Using matrices to describe the system: Example

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1x_1 + 2x_2 \\ 3x_1 + 4x_2 \end{bmatrix}$$

Matrix form of a system of linear equations

Using matrices to describe the system

$$\begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ A_{m,1} & A_{m,2} & \cdots & A_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_m \end{bmatrix}$$

Or simply:

$$\boldsymbol{Ax} = \boldsymbol{b}$$

Identity matrices

- The identity matrix $\mathbf{I}_n$ is a special matrix of shape (n×n) that is filled with 0 except the diagonal that is filled with 1.

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A 3 by 3 identity matrix

- An identity matrix can be created with the Numpy function *eye()*
- When 'apply' the identity matrix to a vector the result is this same vector:

$$\boldsymbol{I}_n \boldsymbol{x} = \boldsymbol{x}$$

Inverse Matrices

- The matrix inverse of **A** is denoted $\mathbf{A^{-1}}$. It is the matrix that results in the identity matrix when it is multiplied by **A**

$$\mathbf{A^{-1}A = I_n}$$

Solving a system of linear equations

- The inverse matrix can be used to solve the equation **Ax**=**b** by adding it to each term:

$$\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

- Since we know by definition that $\boldsymbol{A}^{-1}\boldsymbol{A}=\boldsymbol{I}$, we have:

$$\boldsymbol{I}_n\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

- We saw that a vector is not changed when multiplied by the identity matrix. So we can write:

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

Singular matrices

- Some matrices are not invertible. They are called **singular**.

Thank You!!!!

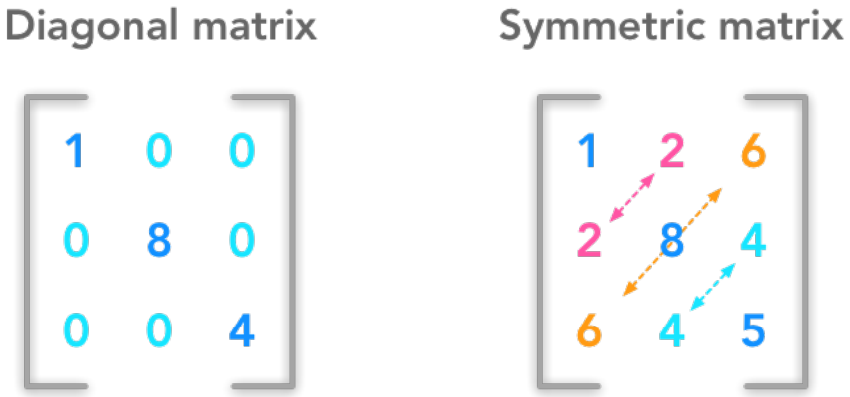
Eigen Decomposition & PCA

An introduction to Linear Algebra (Part 2)

Contents

- Special Matrices and Vectors
- Linear Transformation
- Eigenvectors and eigenvalues
- Eigendecomposition
- Introduction to PCA
- Steps for PCA
- Example

Special Kinds of Matrices and Vectors



- A matrix $A_{i,j}$ is diagonal if its entries are all zeros except on the diagonal (when i=j)
- The diagonal matrix can be denoted *diag(**v**)* where **v** is the vector containing the diagonal values.
- The matrix **A** is symmetric if it is equal to its transpose. This concerns only square matrices

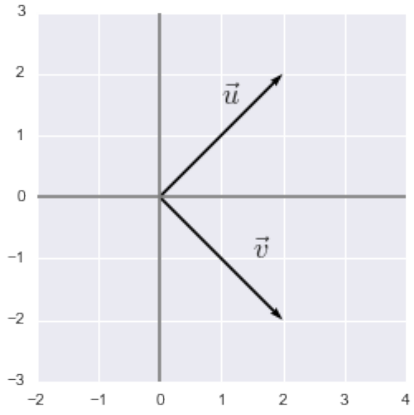
$$\mathbf{A}=\mathbf{A}^T$$

Special Kinds of Matrices and Vectors

- Unit vectors
  - A unit vector is a vector of length equal to 1. It can be denoted by a letter with a hat: $\hat{u}$
- Orthogonal vectors
  - Two orthogonal vectors are separated by a 90° angle. The dot product of two orthogonal vectors gives 0.
  - Example:

$$\boldsymbol{x} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$$

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} = \begin{bmatrix} 2 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ -2 \end{bmatrix} = [2 \times 2 + 2 \times -2] = 0$$

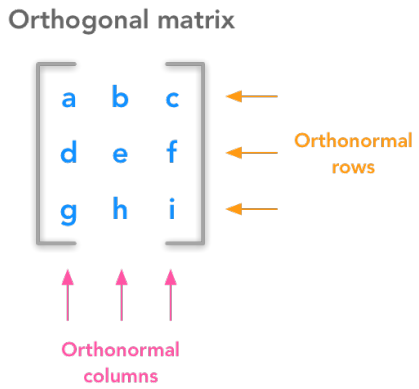


Special Kinds of Matrices and Vectors: Orthogonal matrices

- A matrix is orthogonal if columns are mutually orthogonal and have a unit norm (norm means length or magnitude of the vector) and rows are mutually orthonormal and have unit norm.

Property 1: $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \boldsymbol{I}$

Property 2: $\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{A}^{-1}$



Special Kinds of Matrices and Vectors: Proof

$$\boldsymbol{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \boldsymbol{A}^{\mathrm{T}} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} aa + cc & ab + cd \\ ab + cd & bb + dd \end{bmatrix}$$
$$= \begin{bmatrix} a^2 + c^2 & ab + cd \\ ab + cd & b^2 + d^2 \end{bmatrix}$$

$a^2 + c^2 = 1$ and $b^2 + d^2 = 1$. So we now have:

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \begin{bmatrix} 1 & ab + cd \\ ab + cd & 1 \end{bmatrix}$$

And we know that the columns are orthogonal which means that:

$$\begin{bmatrix} a & c \end{bmatrix} \begin{bmatrix} b \\ d \end{bmatrix} = 0$$

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Matrices as linear transformations

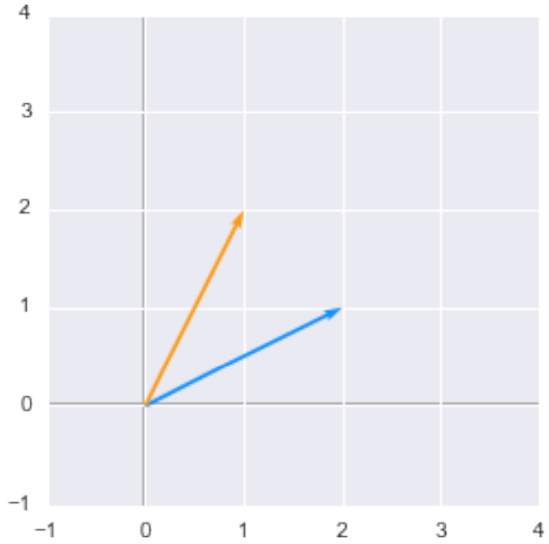
- Some matrices will rotate your space, others will rescale it.
- When we apply a matrix to a vector, we end up with a transformed version of the vector.
- When we say that we *apply* the matrix to the vector, it means that we calculate the dot product of the matrix with the vector

Example: Transformation using matrix

- Let **A**= $\begin{pmatrix} -1 & 3 \\ 2 & -2 \end{pmatrix}$

and **v**= $\begin{pmatrix} 2 \\ 1 \end{pmatrix}$

So applying **A** to **v** gives us $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$



In figure the blue line shows the older vector and orange shows the new vector

- Hence, matrix **A** transformed the vector with an angle.

Using matrices to describe the system

- If the transformation of the initial vector gives us a new vector that has the exact same direction and the scale is different , this special vector is called an *eigenvector* of the matrix.
- Mathematically, we have the following equation:

$$\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v}$$

- v- is the eigenvector and
- λ - is the eigenvalue

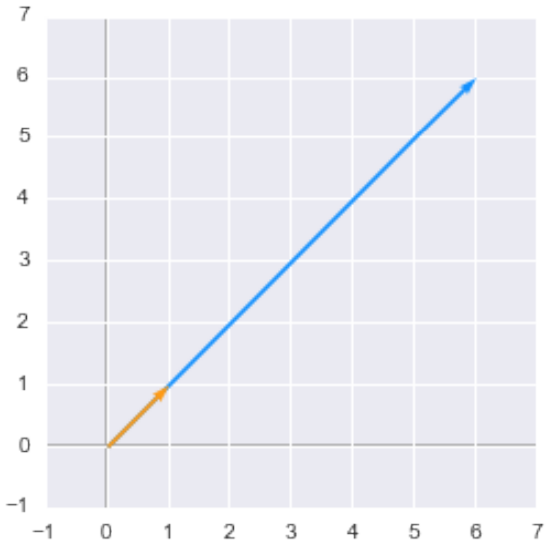
Example: Eigenvector and Eigenvalues

$$\boldsymbol{A} = \begin{bmatrix} 5 & 1 \\ 3 & 3 \end{bmatrix} \qquad \boldsymbol{v} = \begin{bmatrix} 1 \\ \\ 1 \end{bmatrix}$$

- Applying **A** to **v** gives us

$$\begin{bmatrix} 5 & 1 \\ 3 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$$

- Hence λ is 6



Find eigenvalues and eigenvectors in Python

- Numpy provides a function (*np.linalg.eig( )*) returning eigenvectors and eigenvalues.
- Two arrays re returned as a result from the function such that:
  - the first array corresponds to the eigenvalues and
  - the second to the eigenvectors concatenated in columns

Eigendecomposition

- Let **V** be a matrix such that all eigenvectors of a matrix **A** can be concatenated in a matrix with each column corresponding to each eigenvector.
- The vector **λ** can be created from all eigenvalues.
- Then the eigen decomposition is given by:

$$\boldsymbol{A} = \boldsymbol{V} \cdot diag(\boldsymbol{\lambda}) \cdot \boldsymbol{V}^{-1}$$

- For a symmetric matrix, the eigenvectors are orthogonal.

PCA

Naw Varsha Pipada
Asst. Prof, Dept. of CSE
Eng. College Bikaner

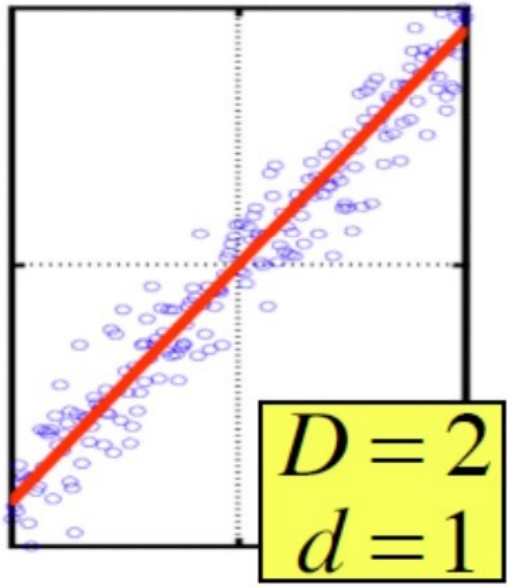
Principle Component Analysis

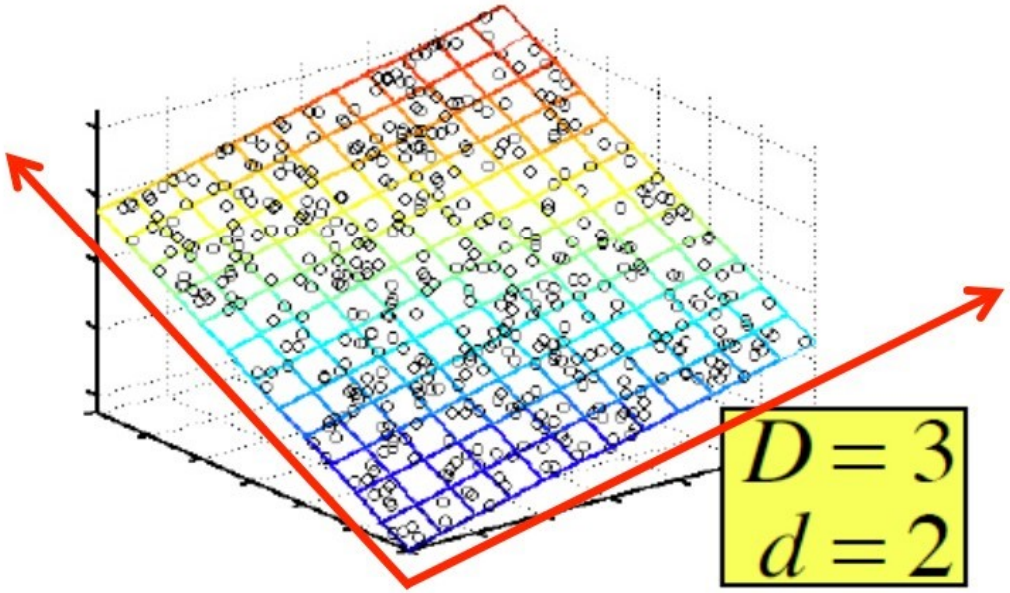
Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process.

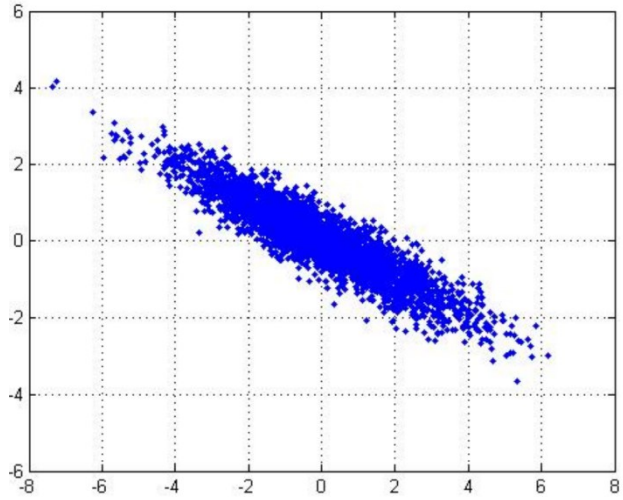
So to sum up, the idea of PCA is simple — reduce the number of variables of a data set, while preserving as much information as possible.

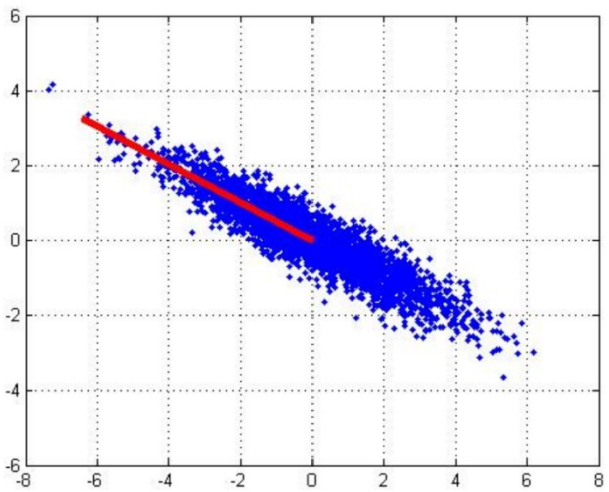
PCA Example

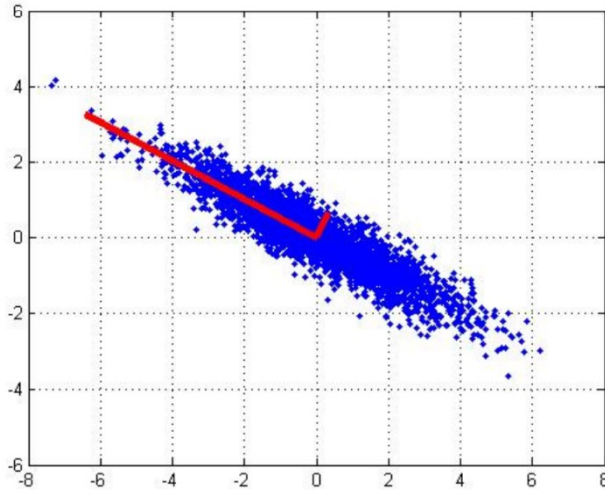




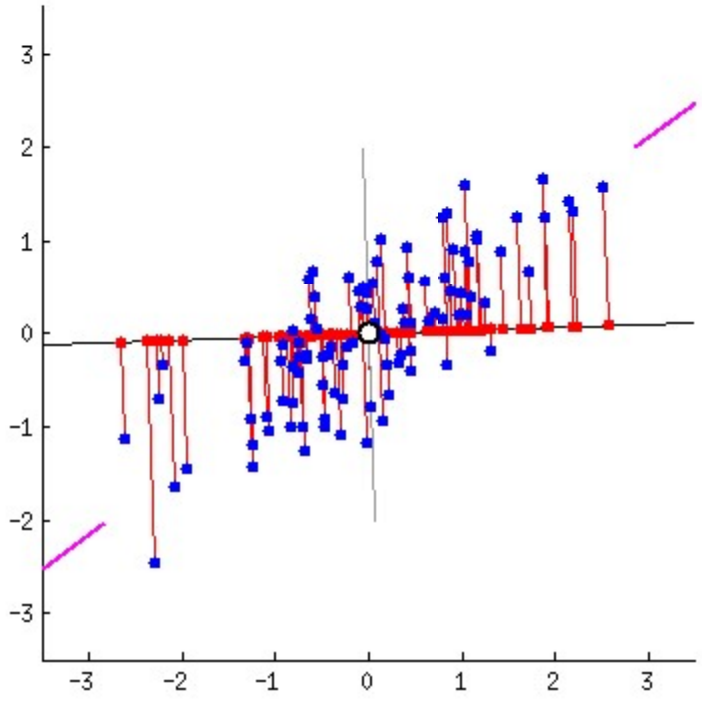
PCA Example







PCA Example



PCA

It is a linear transformation that chooses a new coordinate system for the data set such that

greatest variance by any projection of the data set comes to lie on the first axis (then called the first principal component),

the second greatest variance on the second axis, and so on.

PCA can be used for reducing dimensionality by eliminating the later principal components.

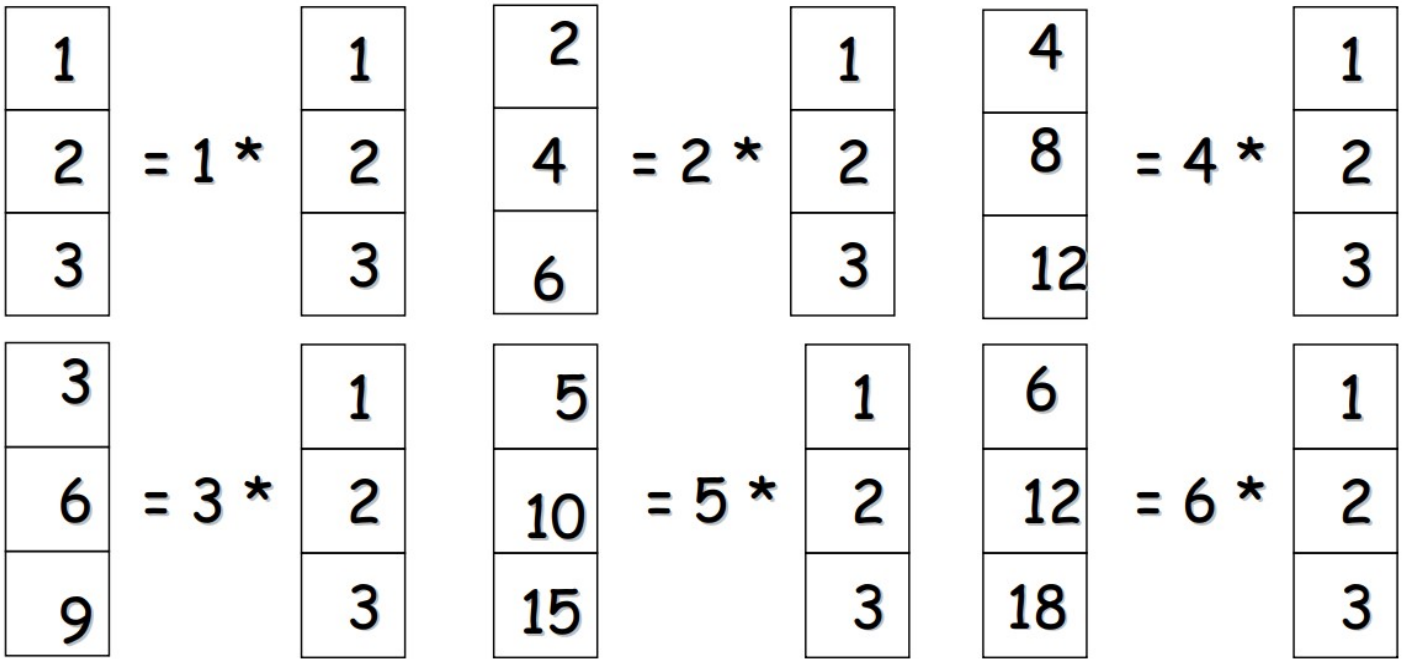
Why PCA?

Consider the following 3D points

1	2	4	3	5	6
2	4	8	6	10	12
3	6	12	9	15	18

If each component is stored in a byte,
we need 18 = 3 x 6 bytes

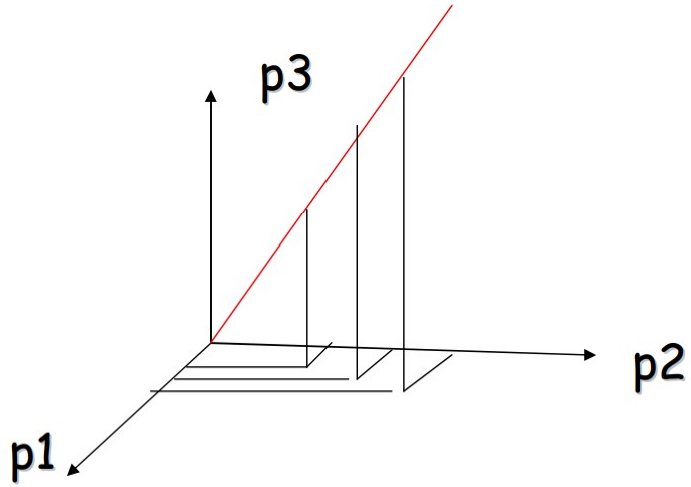
Why PCA?



They can be stored using only 9 bytes (50% savings!): Store one point Store one point (3 bytes) (3 bytes) + the multiplying constants + the multiplying constants (6 bytes)

Why PCA: Geometrical Interpretation

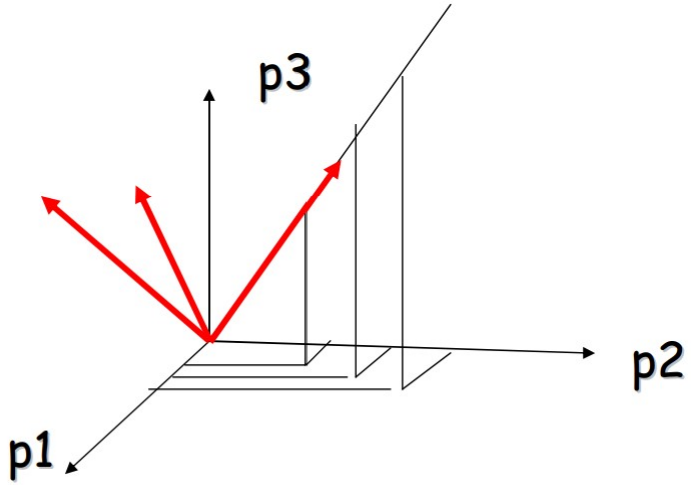
View each point in 3D space.



But in this example, all the points happen to belong to a But in this example, all the points happen to belong to a line: a 1D subspace of the original 3D space.

Why PCA: Geometrical Interpretation

Consider a new coordinate system where one of the axes is along the direction of the line: is along the direction of the line:



In this coordinate system, every point has only one non-zero coordinate: we only need to store the direction of the line (a 3 bytes image) and the nonzero coordinate for each of the points zero coordinate for each of the points (6 bytes).

How does PCA work?

We are going to calculate a matrix that summarizes how our variables all relate to one another. (Covariance Matrix)

We'll then break this matrix down into two separate components: direction and magnitude. We can then understand the "directions" of our data and its "magnitude" (or how "important" each direction is).(Eigendecomposition)

We will transform our original data to align with these important directions (which are combinations of our original variables)

By identifying which "directions" are most "important," we can compress or project our data into a smaller space by dropping the "directions" that are the "least important." By projecting our data into a smaller space, we're reducing the dimensionality of our feature space… but because we've transformed our data in these different "directions," we've made sure to keep all original variables in our model!

Covariance

Variance and Covariance are a measure of the “spread” of a set of points around their center of mass (mean)

Variance – measure of the deviation from the mean for points in one dimension e.g. heights

Covariance as a measure of how much each of the dimensions vary from the mean with respect to each other.

Covariance is measured between 2 dimensions to see if there is a relationship between the 2 dimensions e.g. number of hours studied & marks obtained.

The covariance between one dimension and itself is the variance

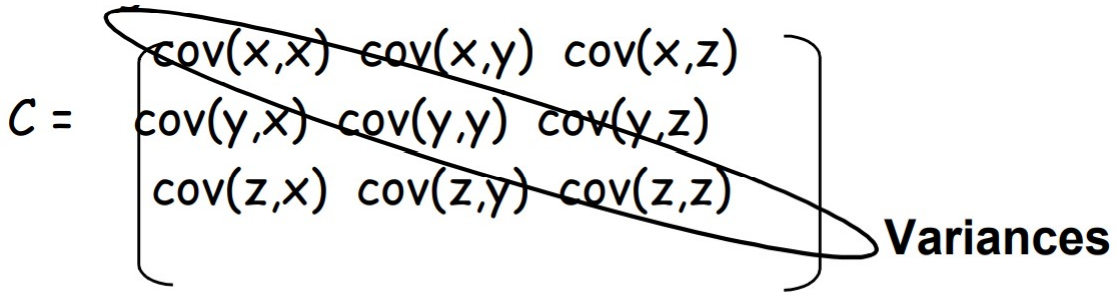
Covariance

$$\text{covariance}(X,Y) = \frac{\sum_{i=1}^{n} (\overline{X_i} - X)(\overline{Y_i} - Y)}{(n-1)}$$

So, if you had a 3-dimensional data set (x,y,z), then you could measure the covariance between the x and y dimensions, the y and z dimensions, and the x and z dimensions. Measuring the covariance between x and x , or y and y , or z and z would give you the variance of the x , y and z dimensions respectively.

Covariance Matrix

Representing Covariance between dimensions as a matrix e.g. for 3 dimensions:



Diagonal is the **variances** of x, y and z

cov(x,y) = cov(y,x) hence matrix is symmetrical about the diagonal

N-dimensional data will result in NxN covariance matrix

Covariance Matrix

Exact value is not as important as it's sign.

A positive value of covariance indicates both dimensions increase or decrease together

- e.g. as the number of hours studied increases, the marks in that subject increase.

A negative value indicates while one increases the other decreases, o vice-versa

- e.g. active social life vs performance in studies.

If covariance is zero: the two dimensions are independent of each other

- e.g. heights of students vs the marks obtained in a subject

Why Covariance?

Why bother with calculating covariance when we could just plot the 2 values to see their relationship?

Covariance calculations are used to find relationships between dimensions in high dimensional data sets (usually greater than 3) where visualization is difficult.

Steps to find PCs

STEP 1: STANDARDIZATION

STEP 2: COVARIANCE MATRIX COMPUTATION

STEP 3: COMPUTE THE EIGENVECTORS AND EIGENVALUES OF THE COVARIANCE MATRIX TO IDENTIFY THE PRINCIPAL COMPONENTS

STEP 4: Forming the FEATURE VECTOR

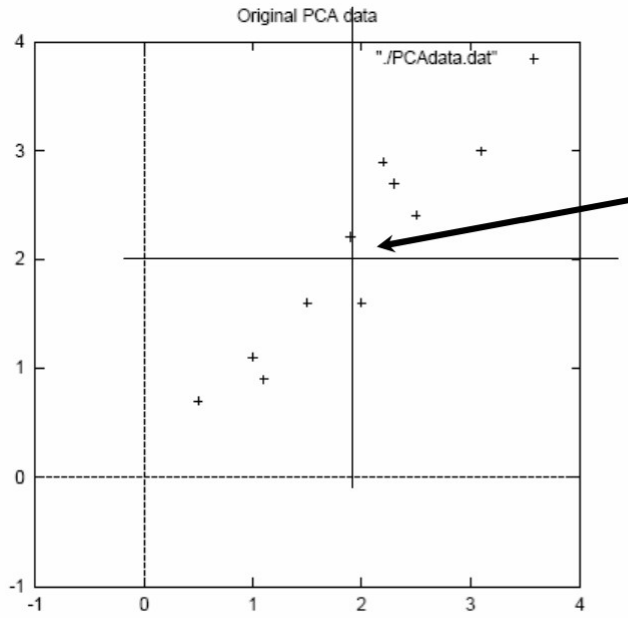
STEP 5: RECAST THE DATA ALONG THE PRINCIPAL COMPONENTS AXES

Example: STANDARDIZATION

$$z = \frac{value - mean}{standard\ deviation}$$

Once the standardization is done, all the variables will be transformed to the same scale.

Example: STANDARDIZATION



mean

this becomes the new origin of the data from now on

DATA:

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

Example: COVARIANCE MATRIX COMPUTATION

$$\text{cov} = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y variable increase together.

Example: EIGENVECTORS AND EIGENVALUES

$$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{eigenvectors} = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -735178656 \end{pmatrix}$$

Example: FEATURE VECTOR

FeatureVector = (eig_1 eig_2 eig_3 … eig_n)

We can either form a feature vector with both of the eigenvectors:

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

or, we can choose to leave out the smaller, less significant component and only have a single column:

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

Example: Transformed Data

In this step, reorient the data from the original axes to the ones represented by the principal components (hence the name Principal Components Analysis).

This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

Pros and Cons

Pros

- Removes Correlated Features
- Improves Algorithm Performance
- Reduces Overfitting
- Improves Visualization

Cons

- Independent variables become less interpretable
- Data standardization is must before PCA
- Information Loss

Thanks