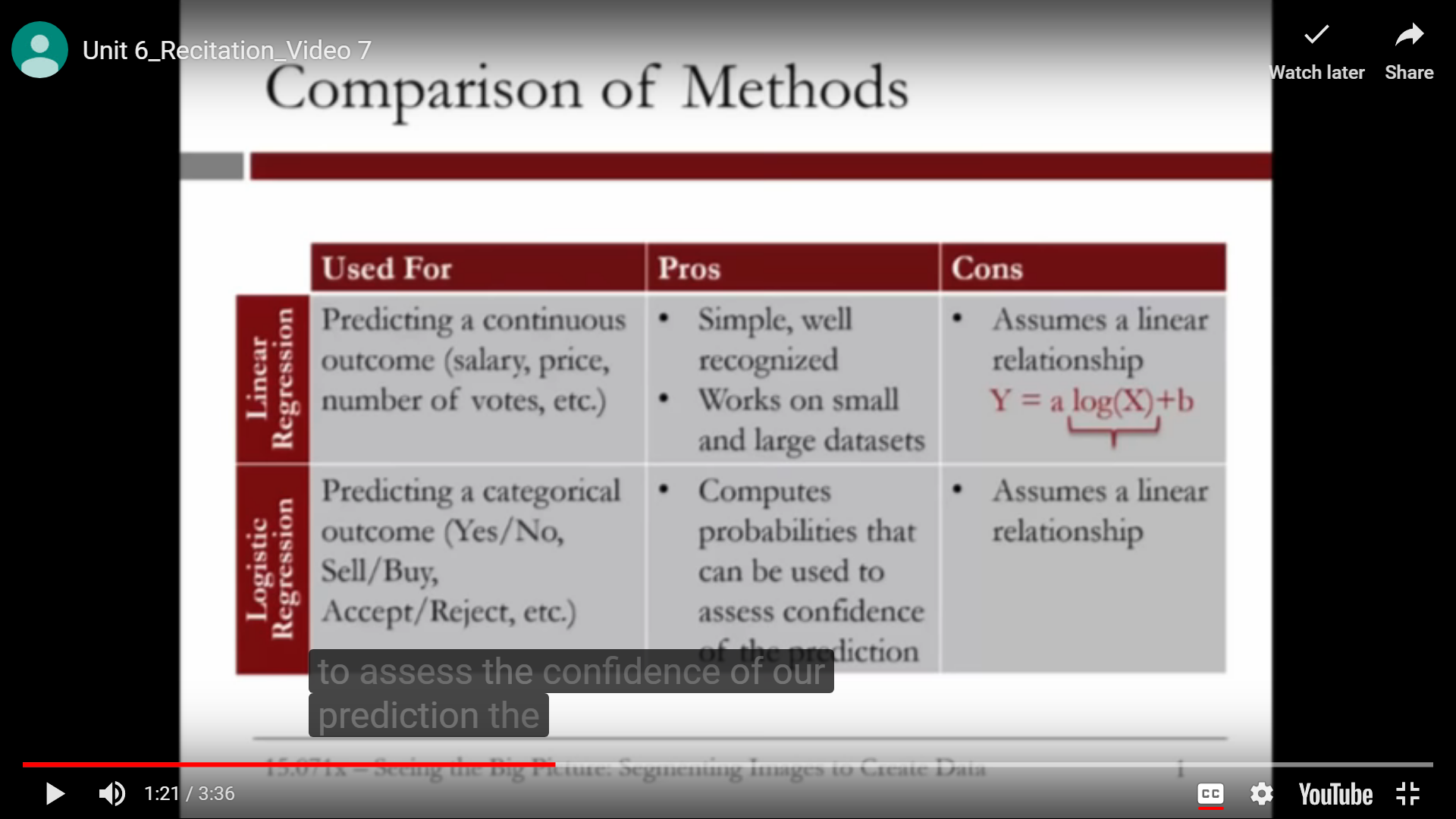
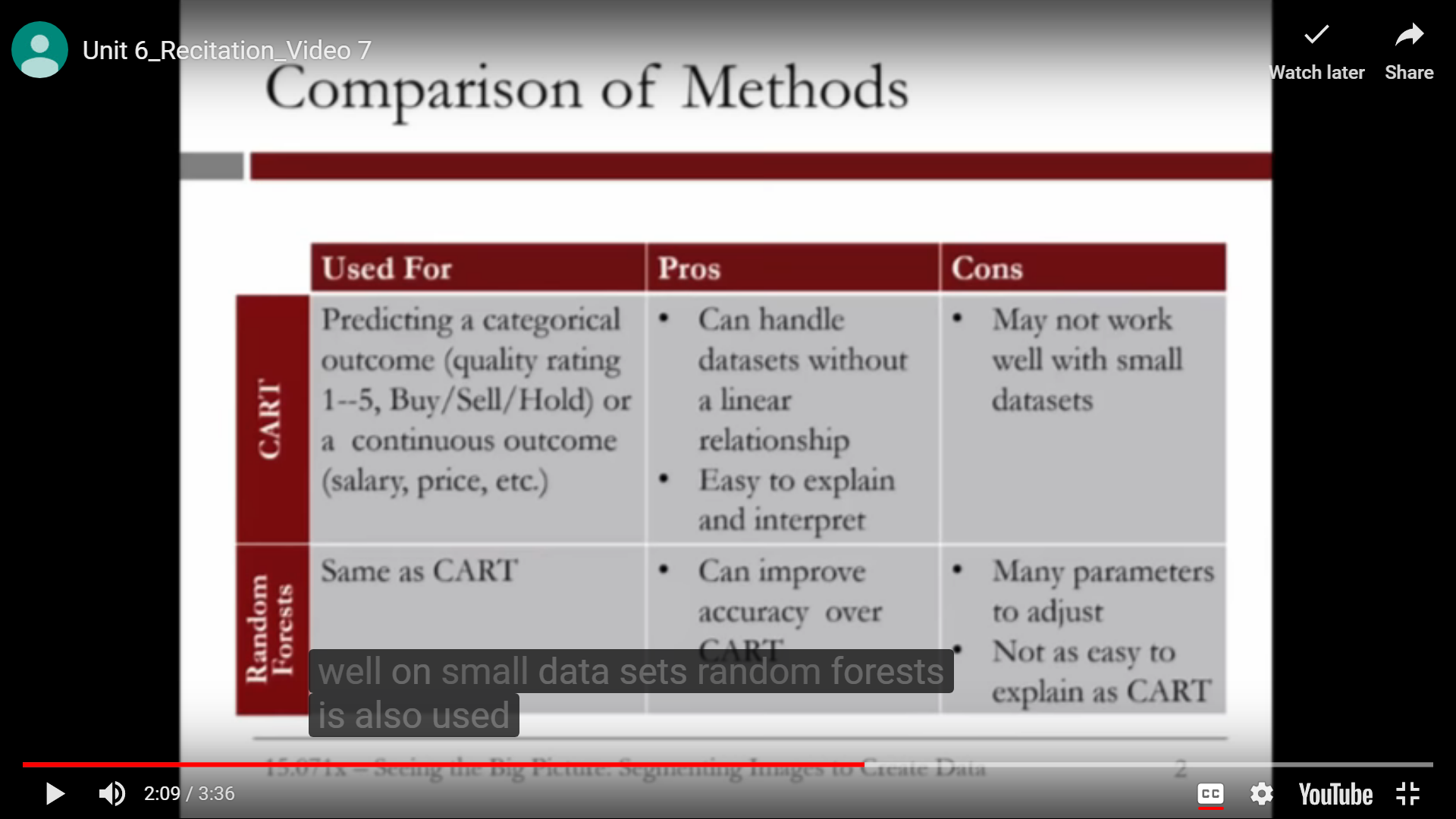
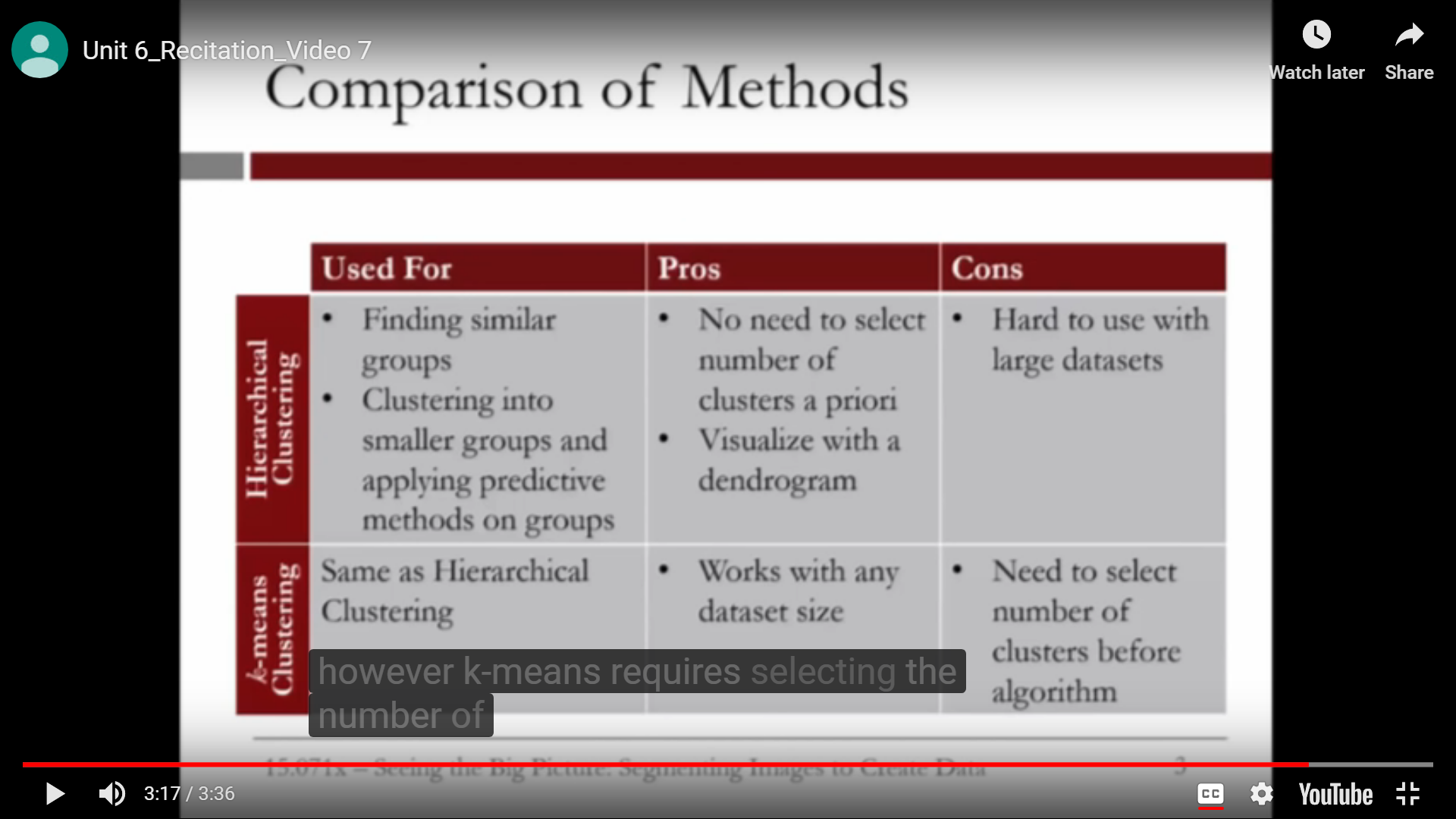
* The technique of making predictions for an observation using other observation is called **Collaborative Filtering**.
  + Can accurately suggests complex items
  + Requires lots of data and coputing power
* The technique of making predictions based on the observation details is called **content Filtering.**
  + Requires very little data
  + Can be limited in scope
* Clustering is a unsupervised learning
  + Goal is to segment the data into similar groups instead of prediction.
  + Then we can build models for each group.
  + It helps to improve the predictive model accuracy.
* Types of CLUSTERING METHODS
  + Hierarchical
  + K-means
* We need to find the distance betwwen to data points in clustering
  + Data point 1 = (x1, x2, x3, x4…...xk)
  + Data point 2 = (y1, y2, y3, y4, …. yk)
  + D = sqrt( (x1-y1)^2 + (x2 - y2)^2 + …… + (xk - yk)^2 )
* Also we need to find the distance between clusters (distance between points that are closest) using same formula.
* In Hierarchical clustering, the two nearest clusters are ecombined to form inta a single cluster. And the process is repeated till no two clusters are closer anymore.
* To read the text file in R:
  + Movies = read.table(“movieslens.txt”, header = FALSE, sep = “|”, quote = “\ “” )
    - Header ⇒ to say that it doesn’t have header or variables row
    - Sep ⇒ for spearator
    - Quote ⇒ to mmake sure that our text was read in properly
* To set the column names for the dataset
  + colnames(dataset\_name) = c(variable1, variable2, ….)
* To remove duplictaes:
  + dataset = unique(dataset)
* For hierarchical clustering we need to compute diatances and cluter distances.
  + Distances = dist(movies[2:20] , method = “euclidean”)
  + clusterMovies = hclust(distances, method = “ward”)
    - Ward method takes in conderation both point distances and cluster distaces.
* We can label the dat points with what cluster it belongs to:
  + Clustergroups = cutree(clusterMoviews, k = 10) ⇒ k defines no of clusters to select
* To know % of movies belong to a gnre:
  + tapply(movies$Action, clustergroups, mean)
* K-Means Clustering:
  + Specify desired no of clusters => k
  + Randomly assign each data point to a cluster
  + Compute cluster centroids ( distances)
  + Re-assign each point to the closest cluster centroid
  + Re-compute cluster centroids
  + Repeat 4 and 5 until no improvement is made
* If we have n points in the matrix then we have to calculate n\*(n-1)/2 pairwise distance values.
* To conevrt a data frame into matrix:
  + flowerMatrix = as.matrix(dataframe)
* TO convert a matrix into vector:
  + flowerVector = as.vector(flowerMatrix)
* Distance = dist(flowerVector, method = “euclidean”)
* clusterIntenity = hclust(distance, method = “ward”)
* flowerClusters = cutree(clusterIntensity, k = 3)
* image(flowerClusters, axes = FALSE)
* image(flowerMatrix, axes = FALSE, col = grey(seq(0,1, length = 256)))
* KMEANS:
  + set.seed(1)
  + K =5
  + KMC = kmeans(healthyVector, centers = k, iter.max = 1000)
  + healthyClusters = KMC$cluster
  + dim(healthyClusters) = c(nrow(healthyMatrix), ncol(healthyMatrix))
  + image(healthyClusters, axes = FALSE, col = rainbow(no\_of\_colors))
* Now we use kmeans result as training and test the testdata. For thsi we need
  + install.packages(“flexclust”)
  + library(“flexclust”)
  + Flexclust contains kcca ⇒ k centroid cluster analysis
  + We need to convert info from cluster algorithm to an KCCA object.
  + KMC.kcca = as.kcca(KMC, healthyVector)
  + tumorClusters = predict(KMC.kcca, newdata = tumorVector)
  + To convert k-means object to matrix to print the output image
    - dim(tumorClusters) = c(nrow(tumorMatrix), ncol(tumorMatrix))
    - image(tumorClusters, axes = FALSE, col = rainbow(no\_of\_colors))
* **COMPARISON OF METHODS:**

****