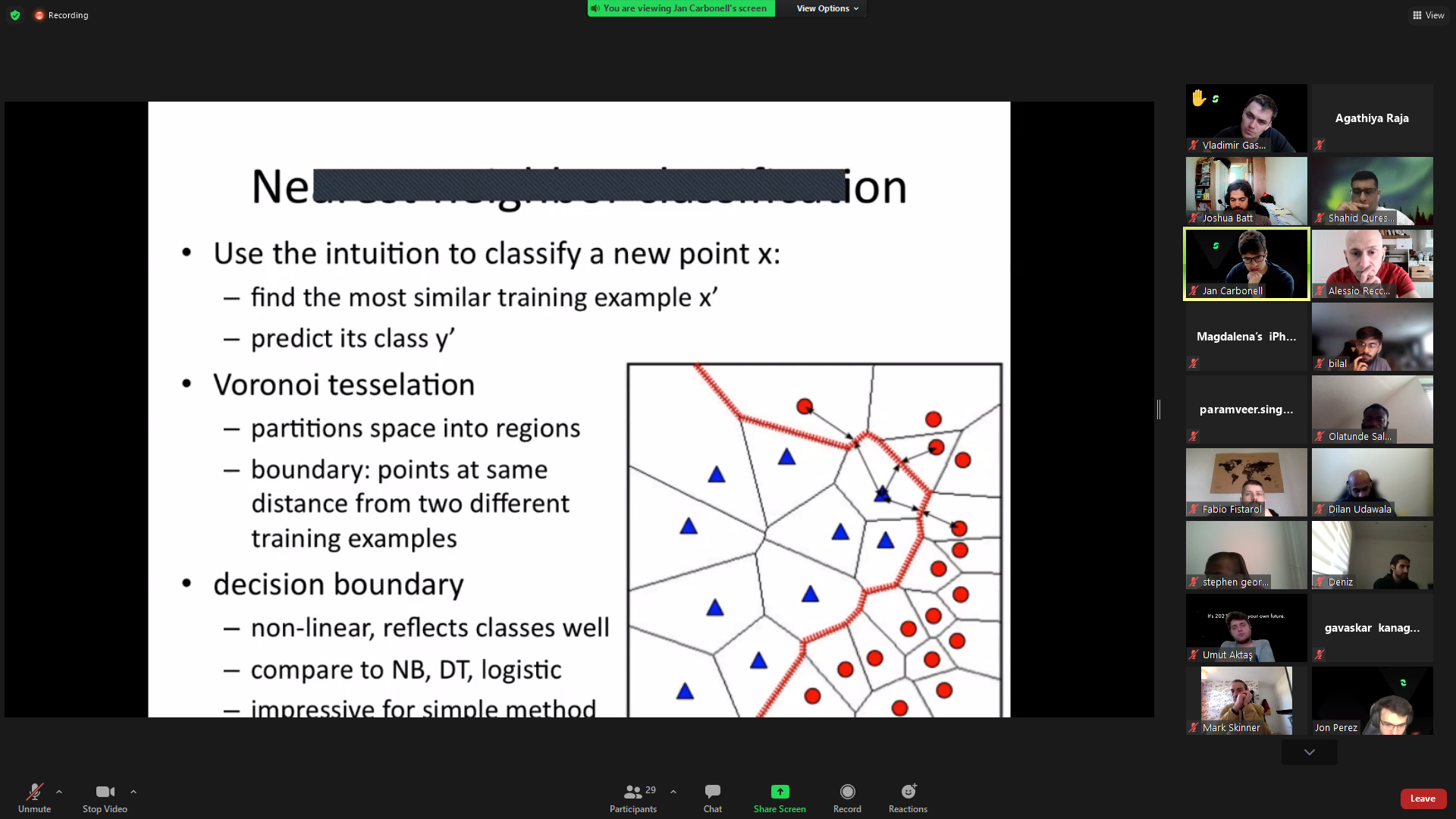
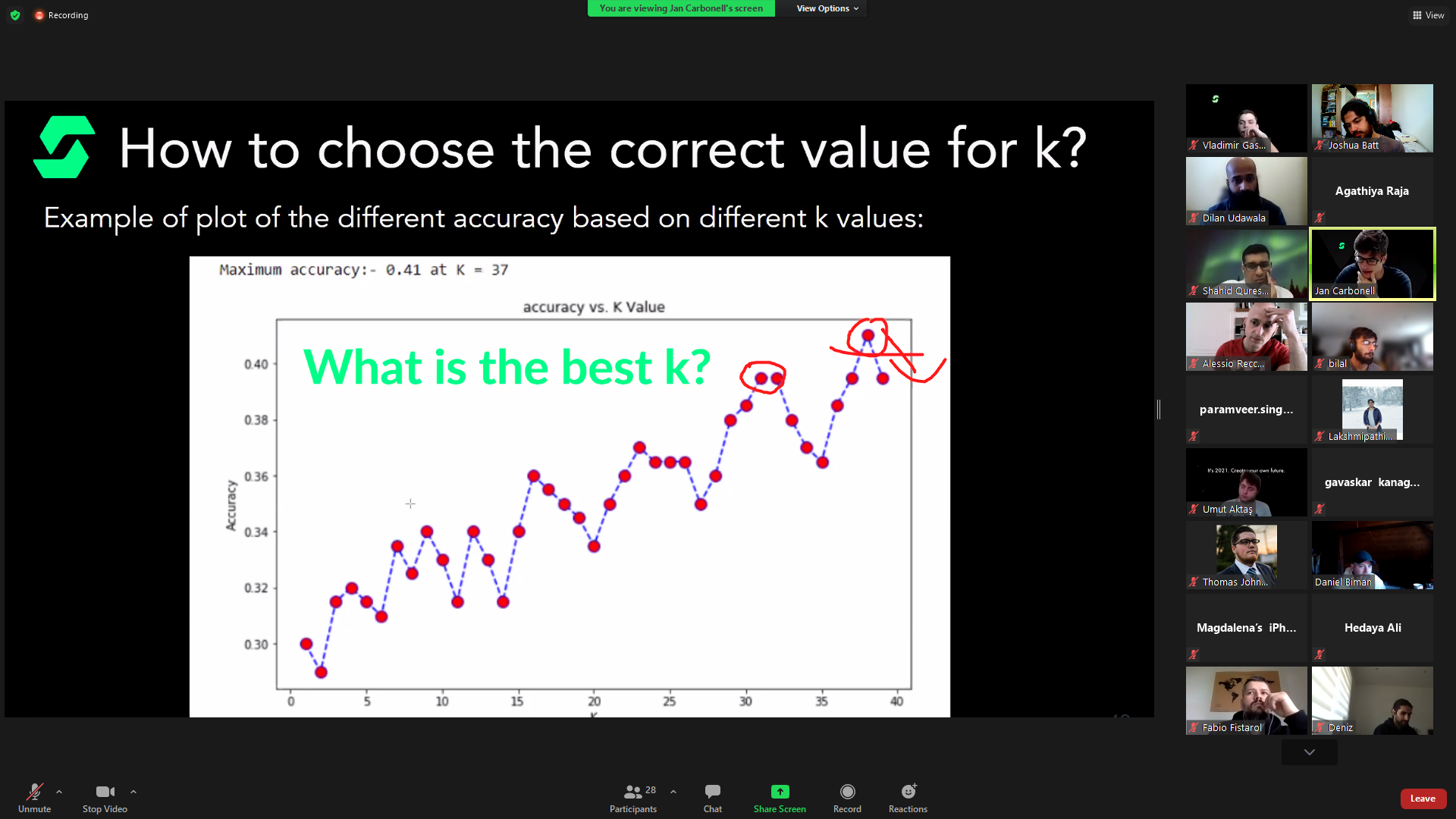
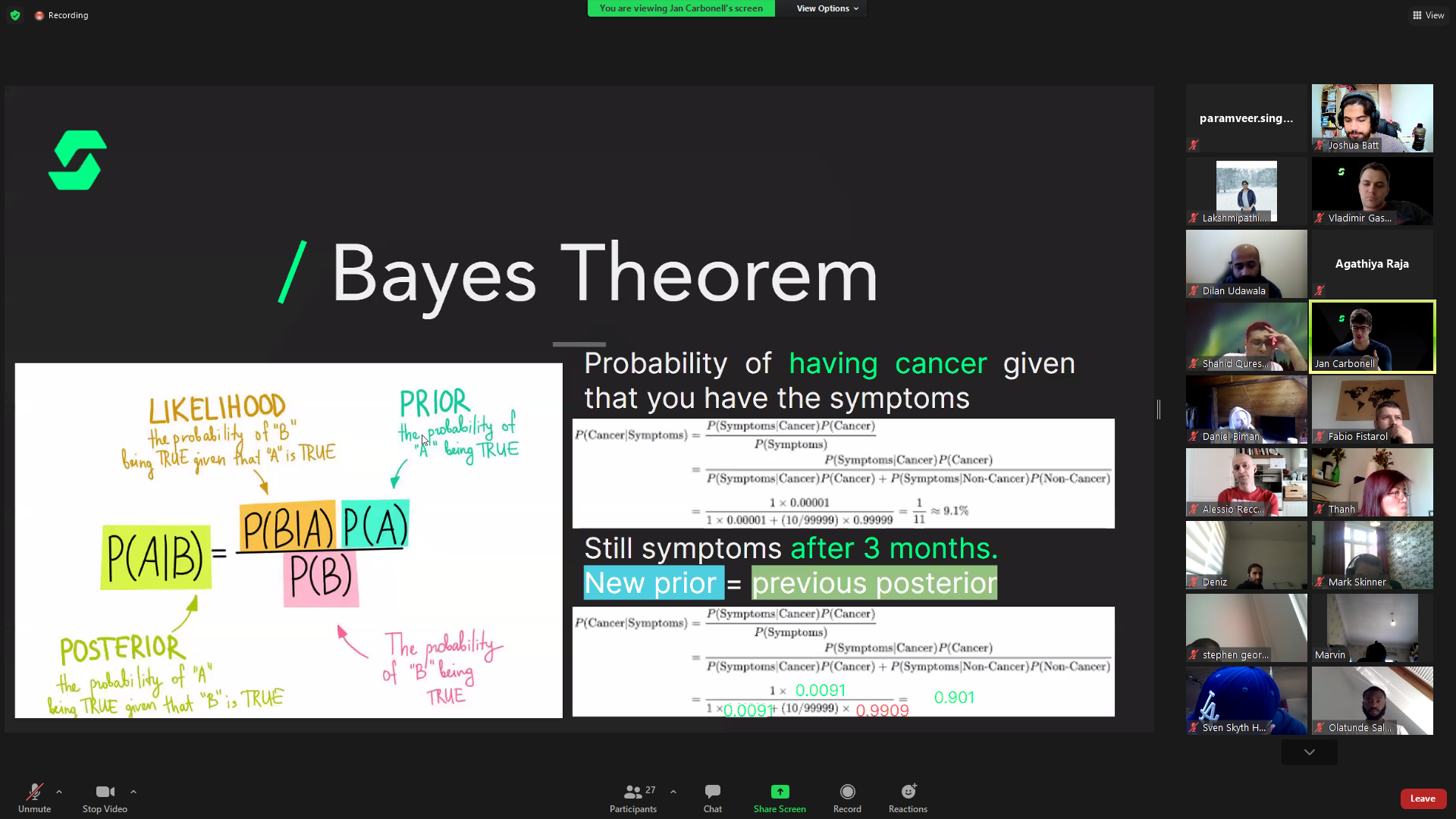
**M3 D7**

**k-NN:**

* Puts point in middle of data cluster.
* k number is how many points of data that surround the point.
  + If k=3 and 2 data points are yellow 1 is purple, then k=3 is class yellow
* Non-parametric, makes no explicit assumptions of the relationships between variables.
* Treats features (Data?) as coordinates in a multidimensional space.
* FEATURES = ~~DATA POINTS?~~ COLOUMNS
  + You find the data point and the derivative of data point and consider both??
  + ANSWER: The features are the columns like age, salary, gender, etc…
* MISSING DATA = K-NN CLASSES THEM AS WHAT THEIR CLOSE TO
  + ANSWER: ~~k-NN does not deal with missing data.~~ k-NN will class NaN values as things its close to, it must deal with those NaNs.
* HOW DOES KNN PICK ITS START POINT = NO CLUE!
  + Based on one random data point?????
  + ANSWER: Picks a random point of data to start form??
* Euclidean distance used in most instances. Exception is words.
  + E.g., Ham is close to Jam if using Euclidean distance.
* Distance calculation for k-NN is heavily dependant on measurement scale of input features. Meaning **we need to normalize our data**.
* Pseudo-code
  + Split your training (x) and test (y)
  + Choose k value (how many neighbours that get picked)
    - **k value must always be an odd number** to never get an even number of classes.
  + For loop through each point in test (y) data:
    - Find distance to all training (x) data points.
    - **Sort the calculated distances** in ascending order.
    - Choose first k points???
    - Assign a class (colour of data points) to the k point.
* If k=1 then every data point becomes its own class
  + Voronoi tessellation: Creates boundaries around each data point.
    - Allows you to create Voronoi maps.
    - Each data point called Voronoi cell.
    - 
    - [place holder]
* **Higher k = Underfitting. Lower k = Overfitting. Trial & error to find appropriate fit**.
* How to choose the right k value:
  + Odd number
  + Try k = square root of N (N is the total number of data points)
    - WHAT IF sqrt(N) is far too big???
  + Test different k=values accuracy
    - Split data in half and see accuracy.
    - Then split in half again (now quarters) to see which one has the best accuracy.
    - Can plot accuracy against k number chosen.
      * K vs Accuracy vs Computation time
      * Example without time, to show how important time axis is:
      * 
      * [place holder]
* Curse of dimensionality
  + kNN does worse with more dimensions
  + Remember: More dimensions you have, the more empty space you have. More work for kNN.
* Advantages of k-NN:
  + Intuitive and simple
  + Has no assumptions.
  + No training step.
  + User can choose the distance variable.
  + Memory based learning approach – Meaning it constantly evolves.
  + Easy to implement for multi-class problems.
* Disadvantage of k-NN:
  + Slow algorithm – Due to memory-based learning (lazy learning). Needs to store the data individually every time.
  + Cannot work with high dimensional data. Meaning number of attributes needs to be small.
  + Difficult to choose ideal number of neighbours (ideal k value)
  + Overly sensitive to outliers in the data
  + Imbalanced data causes problems – If one class has more points than another, kNN will give preference to the class with more. This can cause wrongful representation of the class with less points.
  + Has no way of dealing with missing data (doesn’t deal with NaNs)
* In a real-world business, would you allow k-NN to basically guess the data?
  + If you have more than ~~21%~~ 40% of missing values, then you cannot compute the mean. You drop the rows.
  + If only one column contains the NaNs then drop that column.
  + Basically, preserve as much of the data as possible.

**Bayes Theorym**:

* Probabilities are treated as expressions of belief.
  + Beliefs are updated as more evidence is provided.
  + Bayes is seen as the subjective view on probability.
* 
* This is the base of how AI models work. They update their beliefs based on evidence/ conclusions.
* The more data (evidence) you have, the more accurate your results will be (Including your AI model)
* KEY TAKEAWAY : Bayes theorym is about updating the probability of something based on evidence

**Naïve Bayes:**

* Same as bayes theorym but has no prior evidence to work with. It is the naïve version of bayes theorym.

**Tasks**

**00\_2D\_Knn\_No\_Classes:**

* Use Sklearns make\_blobs() to generate data to test our k-NN.
* Create helper functions:
  + Squaring the difference of two vectors
  + Summing the squared result and returning the square root
  + Calculating the Euclidian distances
  + Evaluation function to evaluate predictions.
* Create k-NN predict function (The k-NN function itself)
* Two data sets are different dimensions:
  + X.shape = 300,2
  + y.shape = 300,
  + Use .T on the multidimensional array to have them operate with each other
    - What does .T do: **Swaps the rows and columns!**
* Get Euclidean distance from vector 1 and vector 2 (V1 = X) (V2 = y) (?):
  + Square the difference of the two vectors.
  + Summing the square differences and returning the square root