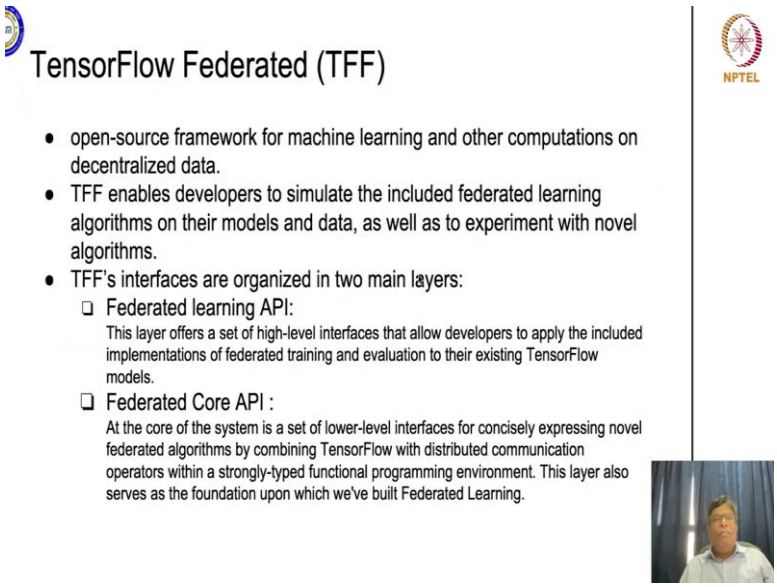
Applied Accelerated Artificial Intelligence
Prof. Satyadhyan Chickerur
School of Computer Science and Engineering
KLE Technological University
Indian Institute of Technology, Palakkad

Lecture - 59
Applied AI: Healthcare (Federated Learning, AI Assisted Annotation)
Session I - Part - 2

(Refer Slide Time: 00:16)



There is something which is called as TensorFlow federated. This is a open source framework for machine learning and other computations on decentralized data. So, you are talking of TensorFlow federated, wherein you are using the concept of federated learning and you are using TensorFlow along with it.

Now, TensorFlow Federated or TFF is going to enable developers to simulate the federated learning algorithms on their models and data as well as to experiment with novel algorithms. And TFF inferences are organized in two main layers ok. So, the idea is that this TensorFlow federated when you talk about their interfaces are organized in two main layers, one is called as the federated learning API and another one is called as federated core API.

Now, this basically means these application programming interfaces of TFF gives you two levels of sophistication. This means the federated learning API is going to give you a

set of high-level interfaces that allow the developers to apply the implementation of federated learning which basically means you can do federated training and evaluation to any of the existing TensorFlow models.

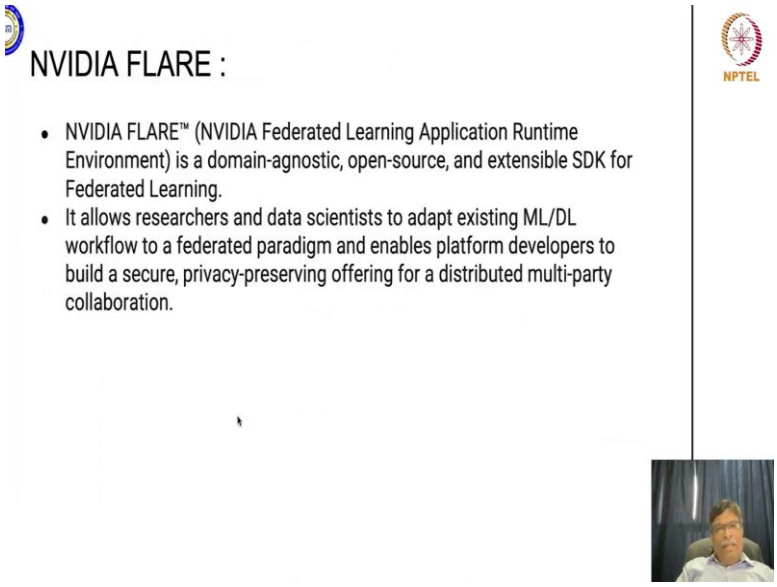
Whereas federated core API is the low-level interface for expressing these federated algorithms by mixing TensorFlow with distributed communication operators. So, see try understanding that we have told the difference between distributed deep learning and federated deep learning to some extent.

Now, in case of distributed deep learning you are trying to distribute either your data or model, in both these cases your data is actually with somebody else. In this scenario what you want is, you do not want the data to go to somebody else, but the model one strain could be used by other people right.

So, this is what is the difference between the distributed deep learning and federated deep learning and since federated deep learning also involves distribution. So, there has to be some communication operation which should happen ok.

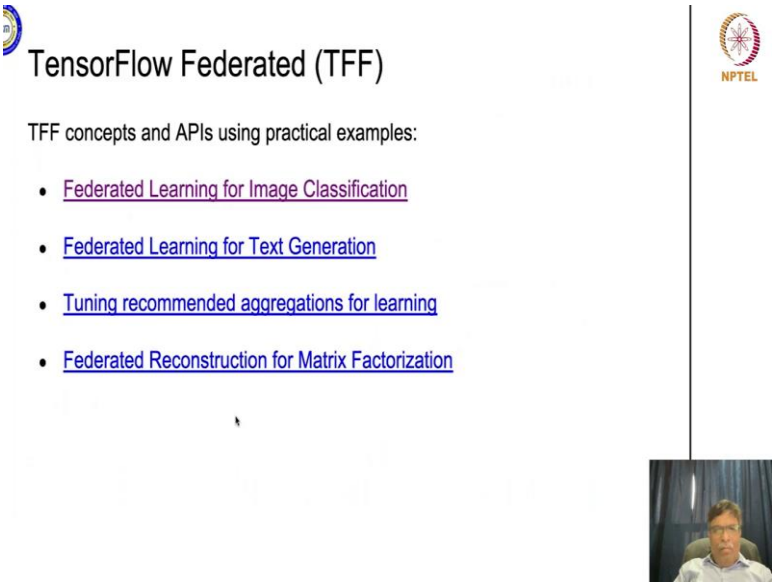
And that communication operation between the various nodes ok has to be as a API and it basically would be something called as federated core API ok. So, today we will do hands on TensorFlow federated and we will try to see four examples of it on Google Colab and tomorrow we will actually go to NVIDIA FLARE which I told.

(Refer Slide Time: 03:46)



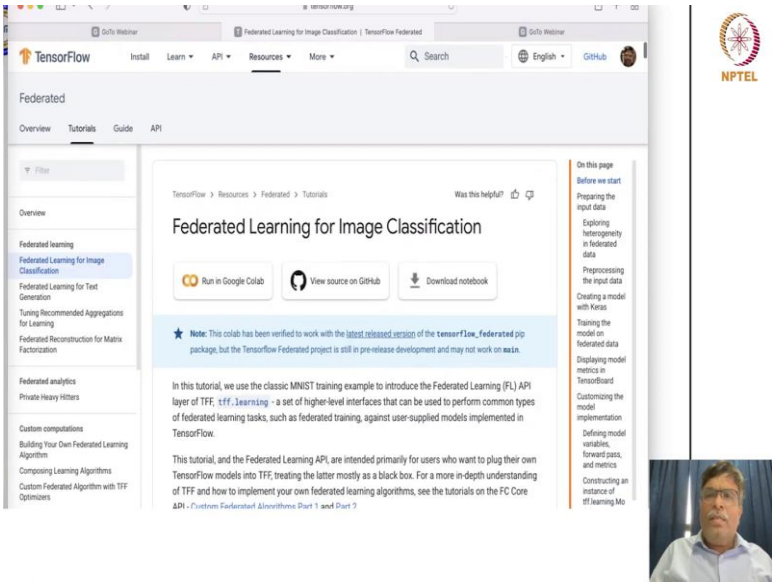
I thought that we can discuss what NVIDIA FLARE is, but tomorrow itself when we discuss this implementation that time if I take it that would be more better right. Otherwise, I was thinking of covering this theory also today, but since we have got some demos which you can do hands on on.

(Refer Slide Time: 04:11)



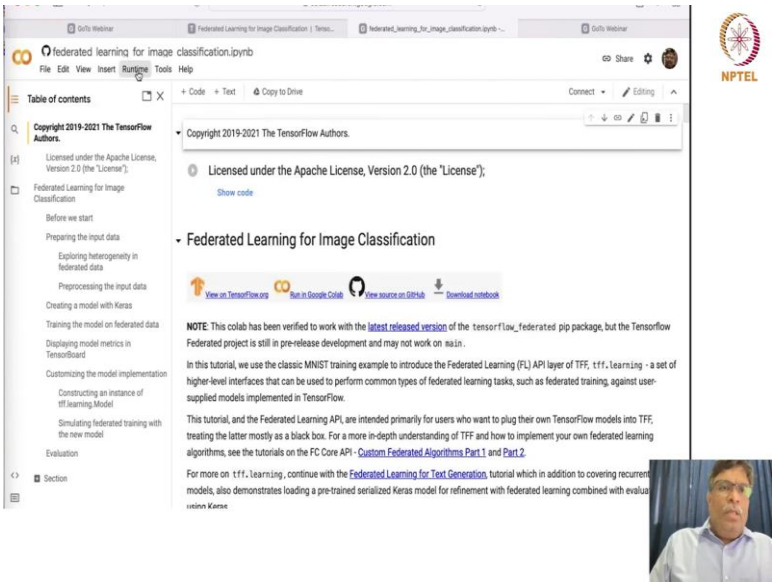
So, I will just go to the hands on portion for today and we will try to understand that how TensorFlow federated ok could be used for image classification, text generation ok and federated reconstruction for something called as matrix factorization. So, based on the time ok I will start with image classification and then to text generation. If the time permits I will try running through these two also, otherwise this link would be shared with you so that you can do some hands on on your own right. So, let us try to go to federated learning for image classification.

(Refer Slide Time: 04:54)

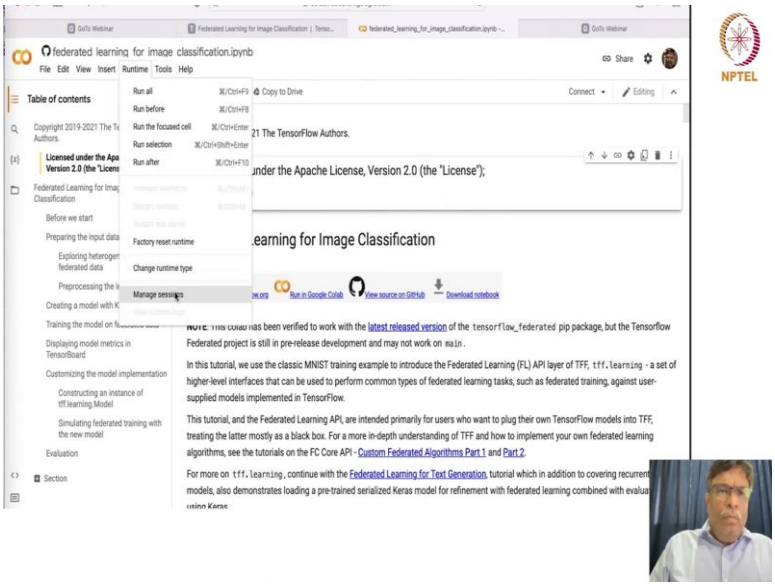


So, so let us try to understand the context and then let us try to do this ah.

(Refer Slide Time: 05:18)

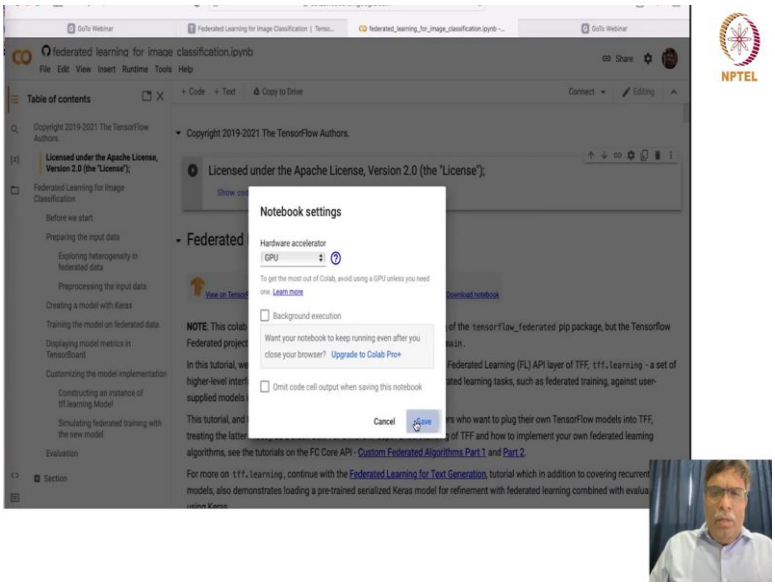


(Refer Slide Time: 05:20)



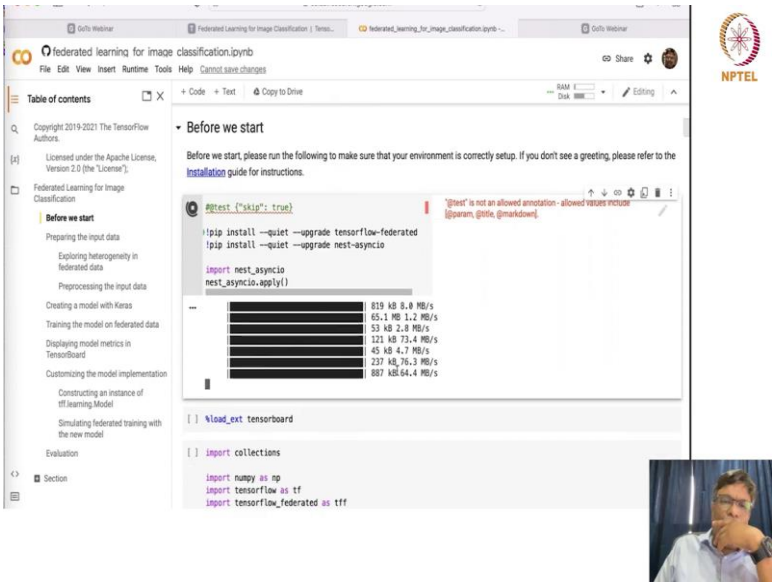
So, let me just change the run time.

(Refer Slide Time: 05:37)



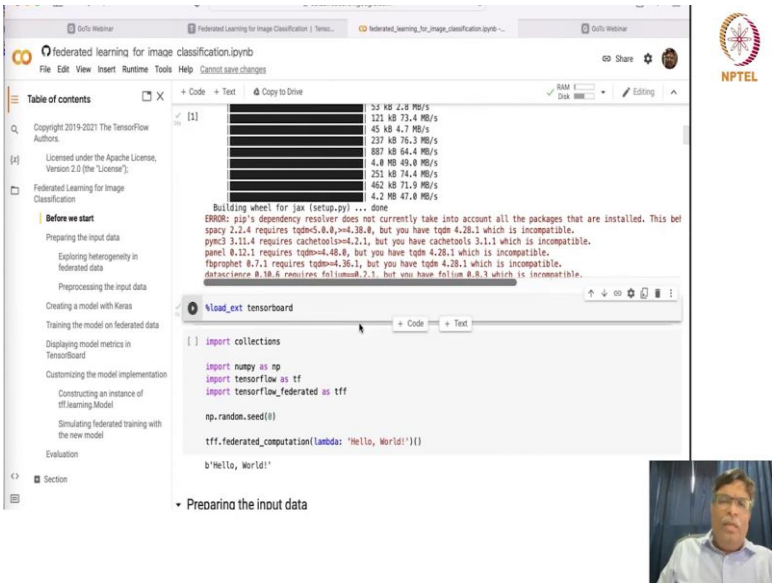
So, here I will do this, do this and then I save this and then let us try to understand this.

(Refer Slide Time: 05:47)



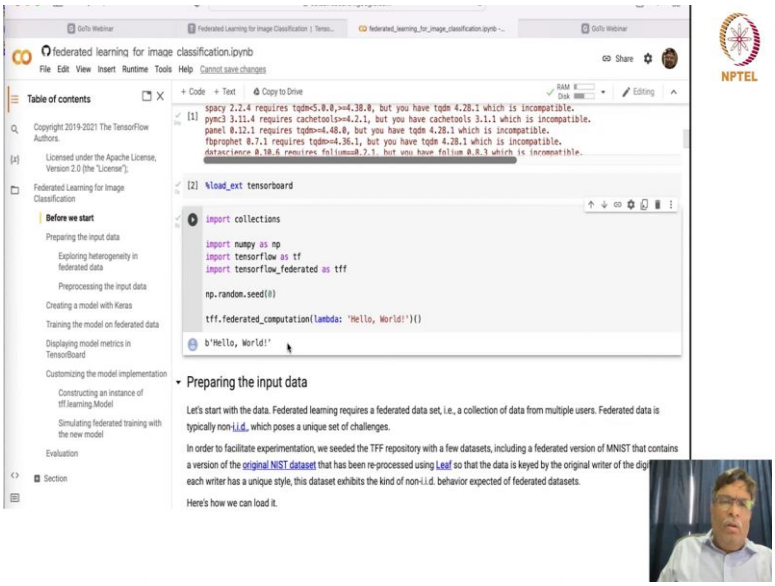
So, these are certain initial environment setups which we need to do, which will actually help us to set up the environment under tensorflow federated and then a synchronous IO operation needs to be set up right.

(Refer Slide Time: 06:20)



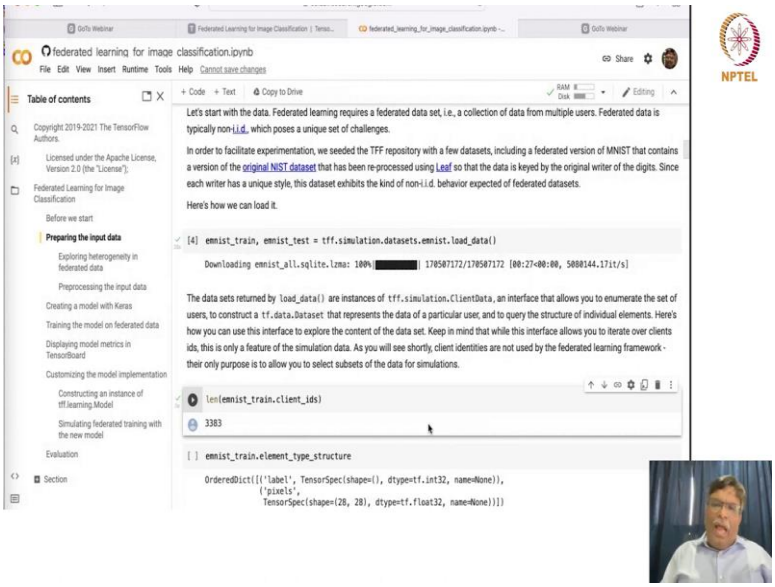
So, yeah so yes so now, there are some incompatibilities. So, let us not worry too much about it because it is not going to affect our execution. So, that is why I am skipping this and go to the next thing. So, yeah so this is basically a tensor board initialization.

(Refer Slide Time: 07:03)



Then you are trying to import in various things like tensorflow federated as tff and this computation of trying to just do some 'Hello World' type of stuff. So, that you know everything is fine. So, this we have got it like this.

(Refer Slide Time: 07:18)



So, now let us try to start with the data right. So, you are supposed to be working with something called as federated data. So, the data has to be you know split and it has to be shared. So, we are trying to work on a Colab wherein you should have the feel of trying to work with multiple users right. So, the federated data ok is typically a non-IID right.

So, when you talk of non-IID it is Independent Identical Distributed type of a random variable basically ok or random data.

So, this is in terms of probabilities when you talk of IID type of data set right. So, now, we they have actually kept some TFF repository with few data sets and we are talking of federated version of MNIST ok. That contains original MNIST data set and they have reprocessed it using this leaf so that the data is keyed by the original writer of the digits.

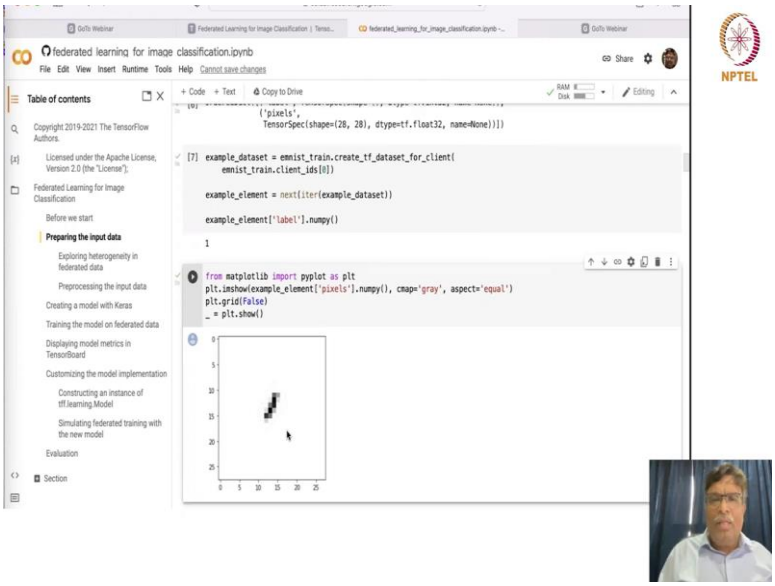
And since each writer has a unique style, the data set exhibits the kind of non-IID behavior expected of federated data sets. See the idea of having a non-IID type of a behavior is that it should be not too much of independent identical distributed random variables, but it should be like somewhat similar, right. You cannot have totally very non IID type of behavior.

So, what you can do is, once we go here we can actually load the tensorflow federated data set which is mnist here ok. So, it is it will take some time does not matter, but yeah. So, now, ok by the time its loading just let me tell you like how the simulation is done. So, here you have got something called as client data right. So, this client data basically is for a particular user ok some portion of this data set ok.

And to query the structure of individual elements you use this function ok. So, the idea is that. So, client identities are not used by federated learning framework, the only purpose is to allow you to select subsets of the data for simulation. So, please try understanding that once you are able to know the client's identity right then you can do many things right.

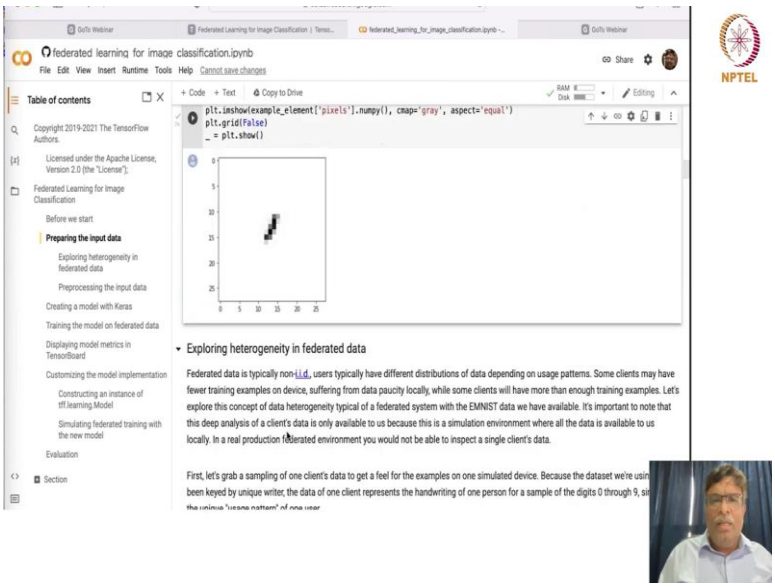
But here the idea is the interface which is being developed or used ok is going to actually iterate over the client ids ok. And you are trying to just get some information about the feature of the simulation data right. So, that is what you are going to see here right.

(Refer Slide Time: 10:28)



So, this is your data set. So, let us try to understand, let us try to see example ok for number 1 this is a data which is basically found out right.

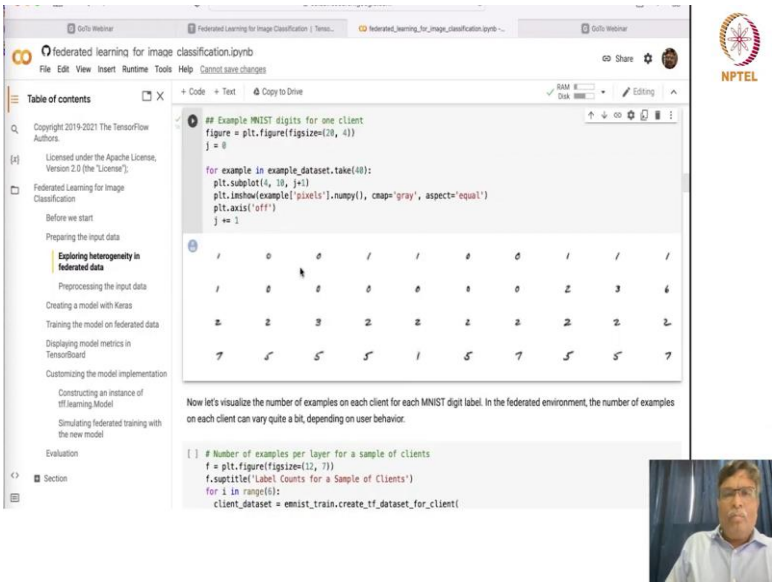
(Refer Slide Time: 10:48)



So, this is just like how federated data set is right. So, federated data is typically non-IID; that means, users typically have different distributions of data depending on the usage pattern. This is very random right probabilistic. So, the idea is that you will have clients, who will have fewer training examples on device; you will have data paucity locally while some clients will have more than enough training examples.

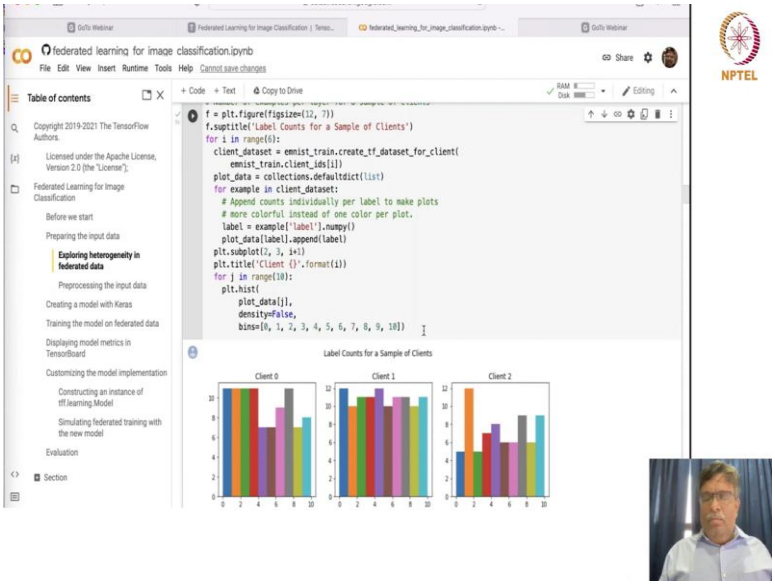
But these when you say that there are only fewer training examples, those training examples would be really very nice ok maybe in case which would going to improve your performance sometimes very better ok, this may happen. So, data heterogeneity is required for improving your system. So, you get that.

(Refer Slide Time: 11:37)



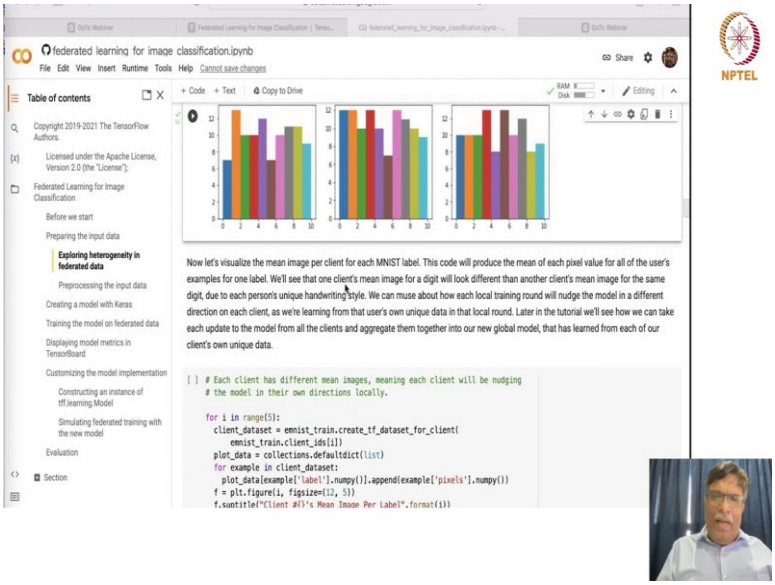
So, let us try to understand that for one client ok. This is how the spread of data is, this is just the example ok. So, you will have so many 1's ok, so many 0's and then you have 5 written in these fashion, 7 in this so like this.

(Refer Slide Time: 11:55)



So, for each client right you will have got some different types of frequencies right, for each of these digits ok.

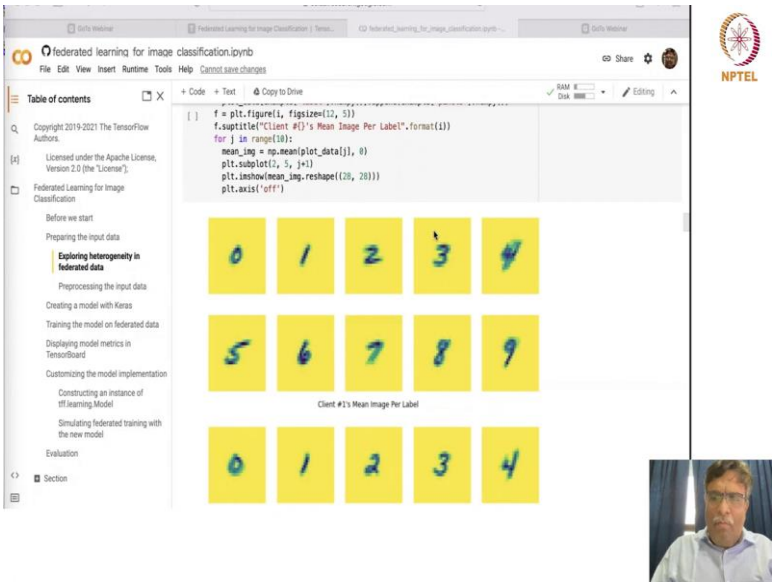
(Refer Slide Time: 12:13)



So, now here you will get client 0, client 1, client 2, client 3, client 4 and up to the client 5. So, each of these clients will have different numbers ok for 0, 1, 2, 3 up till 9 digits ok. So, this basically (Refer Time: 13:06) approximately 11 values ok for digit 9, this client 0 has got about 8 values or 8 types of digit 9 right something like this.

So, now, once we get this, we can visualize the mean image per client for each MNIST label right. So; that means, that we can find out the mean of each pixel values for all of the user's examples for one label right.

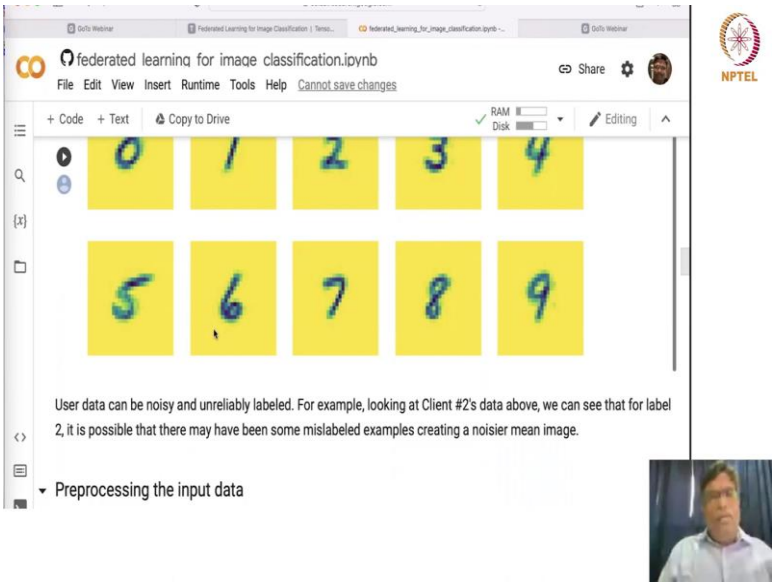
(Refer Slide Time: 13:38)



So, each client is going to have different mean images, meaning each clients will be nudging the model in their own directions locally. Because when your mean image changes right and when you are talking of something like a model and assume that you are using CNN.

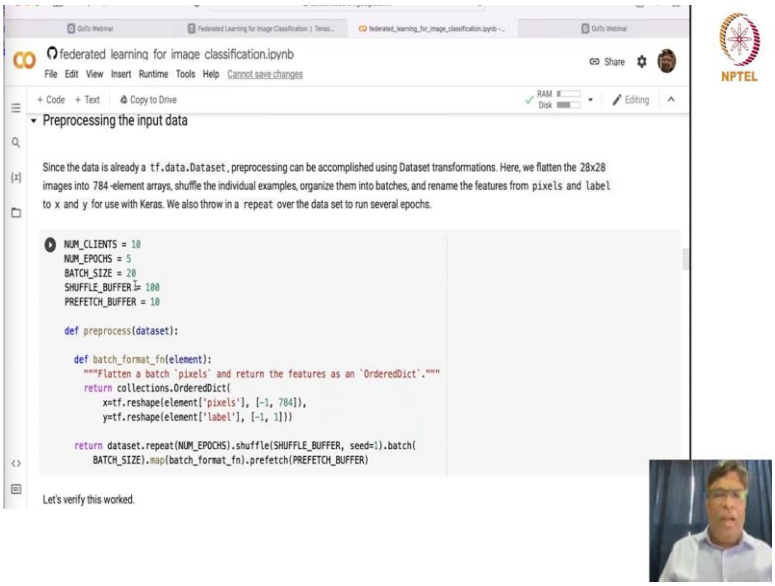
So, the mean image and the actual image is which is going to tell you the direction right of I suppose I need to zoom a bit if possible ok. So, let me close this so that you know, yeah. So see here that when you are talking of client 0's mean image per label this is how it is, client 1's this is how it is client 2 this is how it is. So, if you see this these are all not same mean images right ok.

(Refer Slide Time: 14:34)



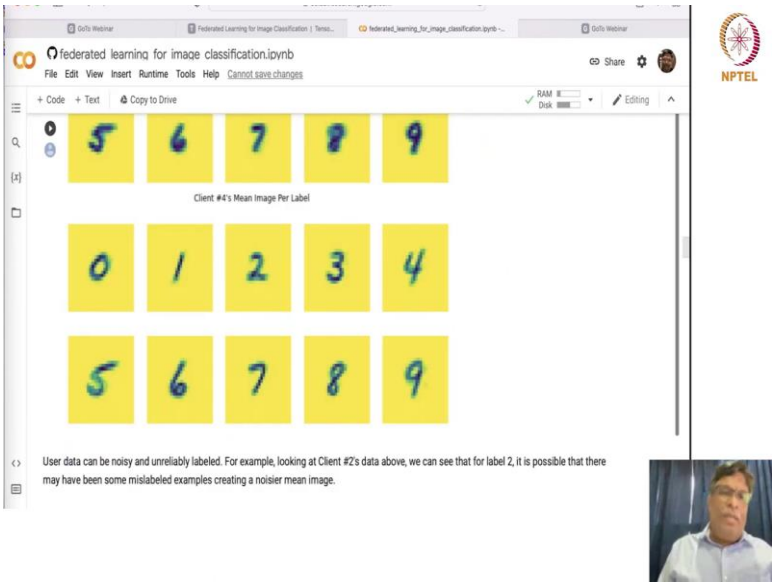
So, this basically means your data will be noisy, it will be unreliably labelled ok.

(Refer Slide Time: 14:38)



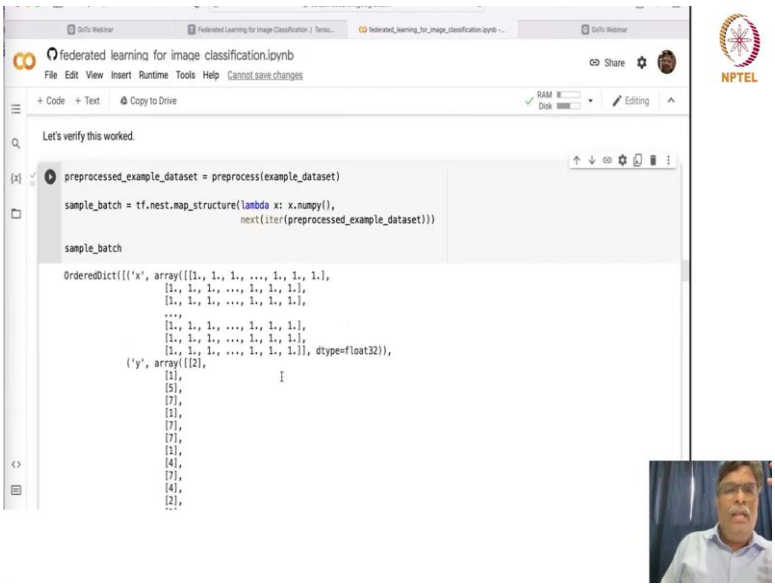
ah And then the mean image right is going to be affected. So, this is how it is going to happen ok.

(Refer Slide Time: 14:57)



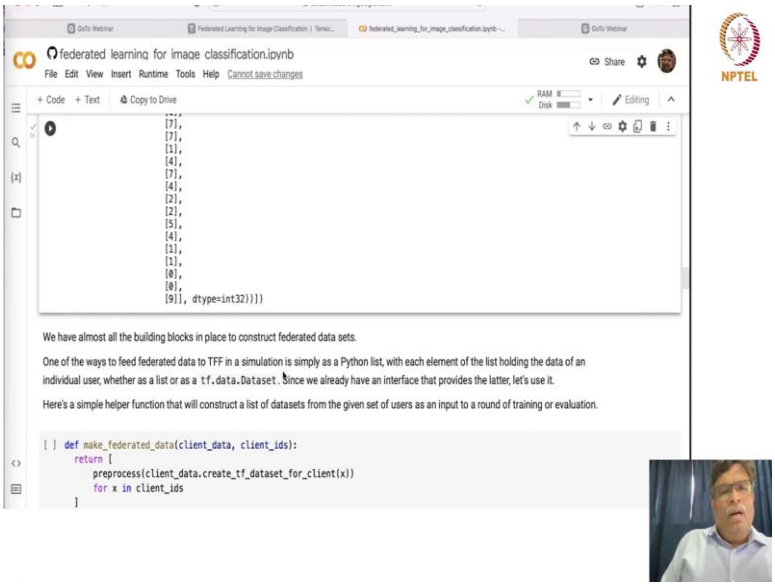
So, now so the idea is that you need to pre process that image. So, let us say that you have got 10 number of clients and your epoch size is 5, batch size is 20 you have some shuffle buffer right and all of that. So, let us try to understand that you can pre-process ok for data set transformation ok. So, that you do and then once you have pre-processed it ok.

(Refer Slide Time: 15:28)



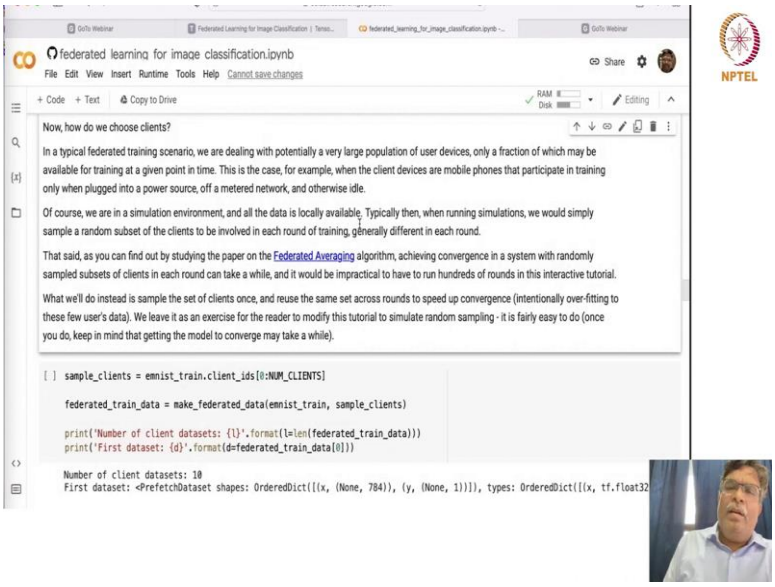
So, you basically are in a position, ok.

(Refer Slide Time: 15:37)



To get a initiation of how do you construct the federated data sets, right.

(Refer Slide Time: 15:54)

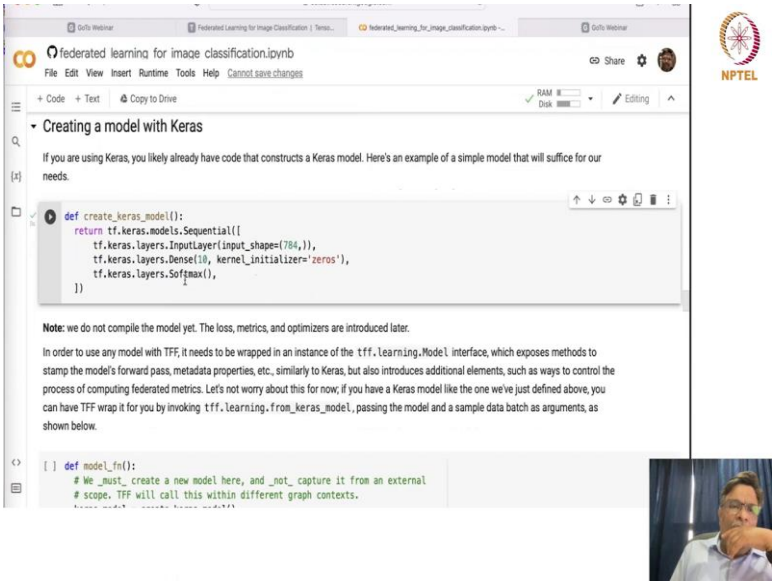


In the sense these are all federated data sets right, each of these clients have their own data and it has to have its own mean, it has to have everything ok. And then there is some specific simple Colab TFF the function which gives you a federated data thing right. So, this function is going to help you construct a list of data sets from the given set of users as an input to a round of training on evaluation.

So, the idea here is, see you have got now this data which is split among various clients right. But out of all of these you are supposed to get the data ok, from the given set of users as a input to a round of training of evaluation right. Now how do we choose the clients? Ok.

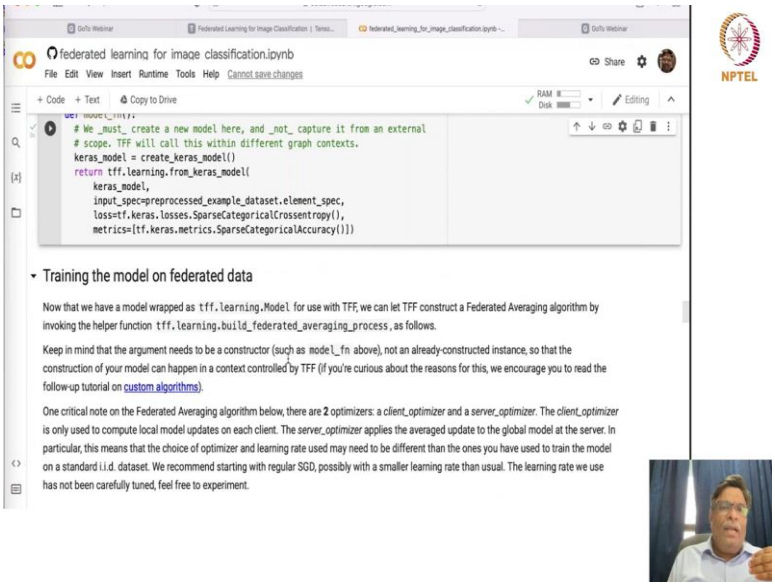
So, generally in a typical federated training scenario we are dealing with a potentially large population this, this, this only fraction of which may be available for training at a particular time, right. So, the idea is when the client devices are mobile phones that participate in training only when plugged into a power source of a metered network and otherwise this. So, the idea is we are in a simulation environment and all the data is locally available here, right.

(Refer Slide Time: 17:09)



So, the idea is you can talk of something called as federated averaging algorithm in this paper, we are not going into this. But the basic idea is that you are going to now generate ok, the client data set right. So, there are 10 clients and each of these clients will have certain data with you ok. Now, we will have to develop our own model right and this is a KERAS model.

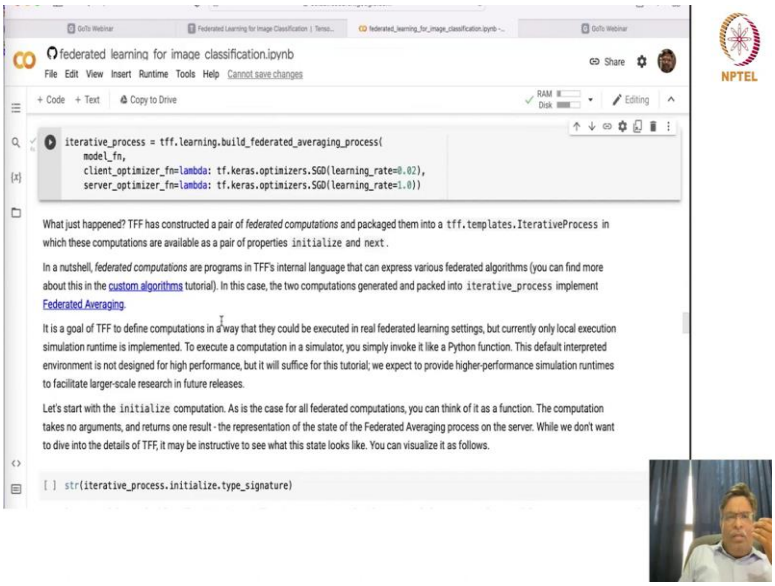
(Refer Slide Time: 17:48)



So, I am not going into the details, I am just telling you that we are creating some model ok. This is a some training model ok and now when you develop some model here ok you have to use that model with the federated data right. So, you are supposed to interface this model ok in such a way that it is going to actually give you ok certain matrices or matrix which are federated in terms right.

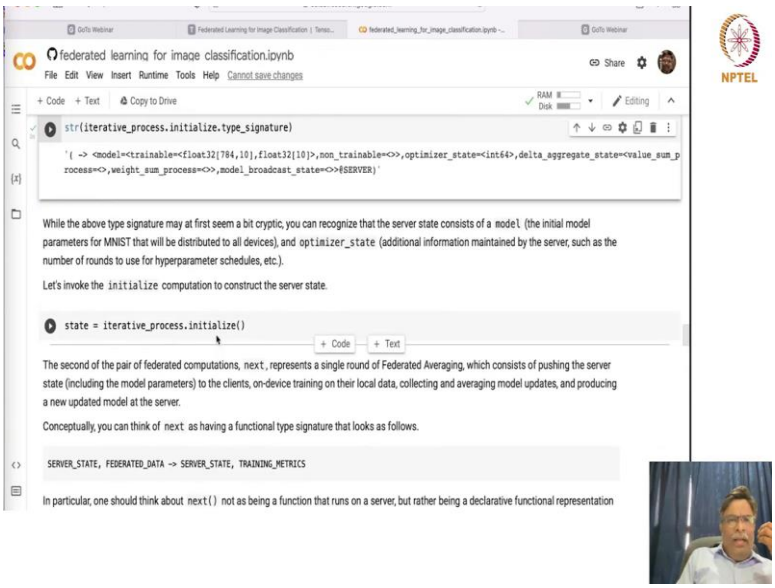
So, we are going to actually use this TFF learning from this model passing the model and a sample data batches as argument. So, the basic idea is you are now trying to actually develop a model, interface it in such a manner that you can use that federated data which you have got ok for training your model ok.

(Refer Slide Time: 18:46)

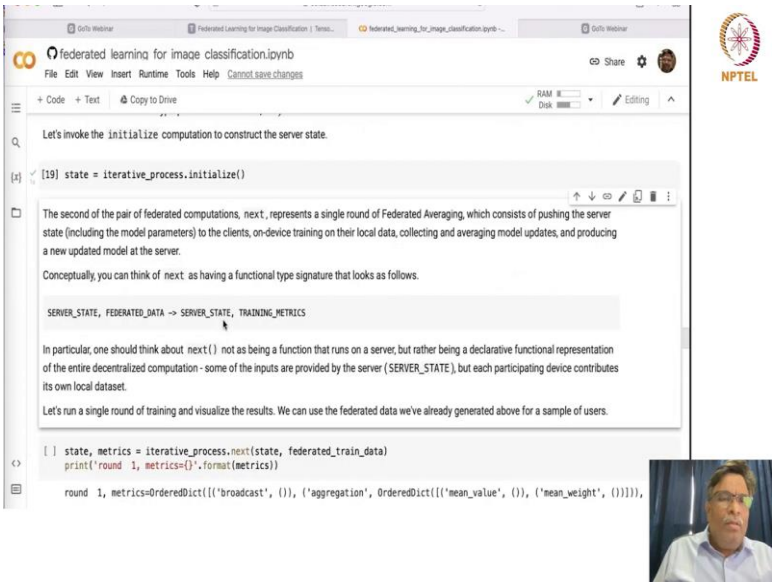


So, now since you have got this model wrapped as a TFF learning model ok for the use with TFF we can let TFF construct a federated averaging algorithms by invoking the helper function ok which basically does the averaging right. So, you are going to actually develop a model now ok, which is a iterative process do this and then ok.

(Refer Slide Time: 19:28)



(Refer Slide Time: 19:50)



Now, the idea here is we should understand here that this type of a signature what we are trying to see here. Because you are trying to actually generate iterative process, you are trying to develop a model which is trainable ok and you are trying to see that how many of this portion ok what data you are using for training, what data you are not going to use for training and then you will have to aggregate this information ok.

And then broadcast it to the server, what does it mean is that you will have a server state ok which basically is the initial model right. You would have a centralized trained MNIST model ok. And that model will be distributed to all the devices that is called as a server state initial model right so that is a model.

And then optimizer state which basically means this additional information which is maintained by the server which will include various parameters, hyper parameters and all of that ok is something which basically is the state of a server.

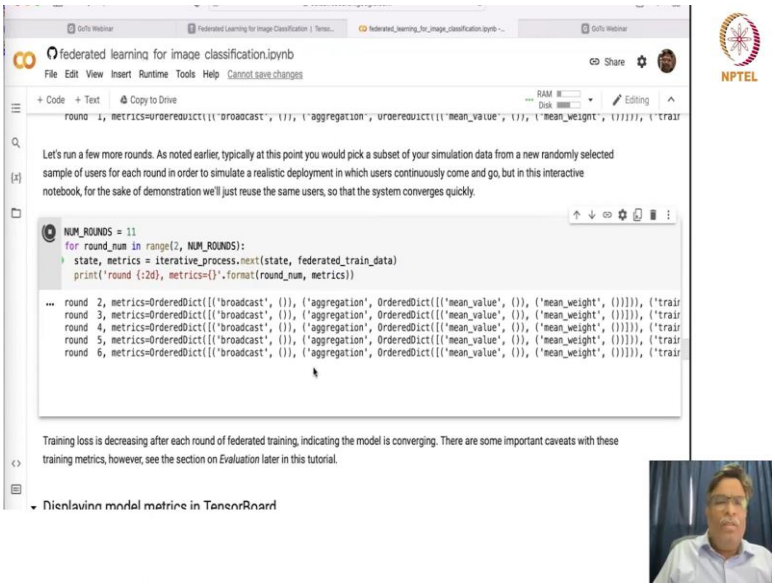
So, you are having a server state which talks about the basic model and then there is a optimizer state for the server which tells about like what are the various hyper parameters being used, how many epochs, how many rounds, how many iterations and so on and so forth.

So now, the idea is you need to initialize this process of training. Now once you are able to do it, once you have done a single iteration ok which is single round of iteration then

you will go in for a single round of federated averaging ok. So, when you do this averaging what effectively happens is you are pushing the server state to the client. Then you do this on device training on the local data, collect it back and then do averaging ok on that model which you updated the server right.

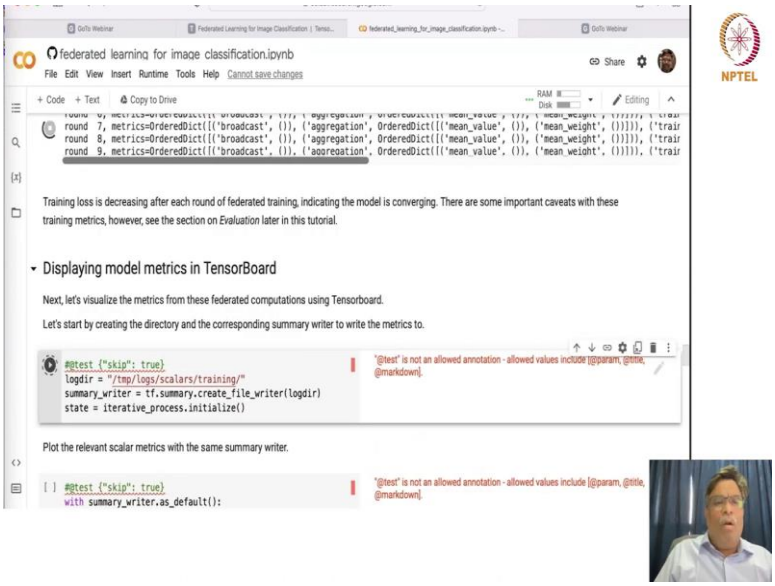
So, your server state is going to change with the data which it has used. So, your server state comma federated data changes to server state with training metrics. Because your federated data is not going to be shared, it is only the training matrix which is going to be shared right. So, this is how it is going to happen.

(Refer Slide Time: 22:31)



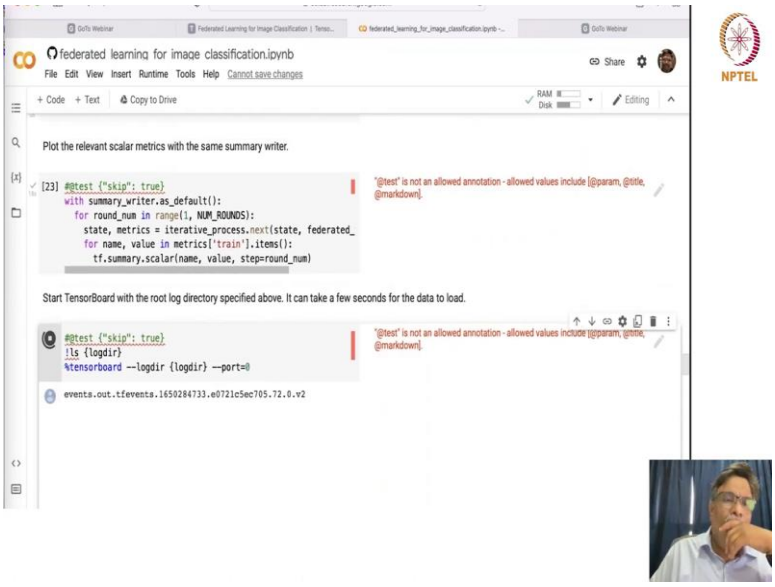
So, let us try to understand this in a bit maybe once you do a hands on and read through this we will be able to understand it better, but at least a gist of how basically it happens ok is what we are trying to tell you right. So, now, you do so many rounds of it. So, many iterations ok, then this is how the matrix is going to change ok. So, mean weight, this, that, aggregation you are doing broadcasting all of this.

(Refer Slide Time: 23:02)



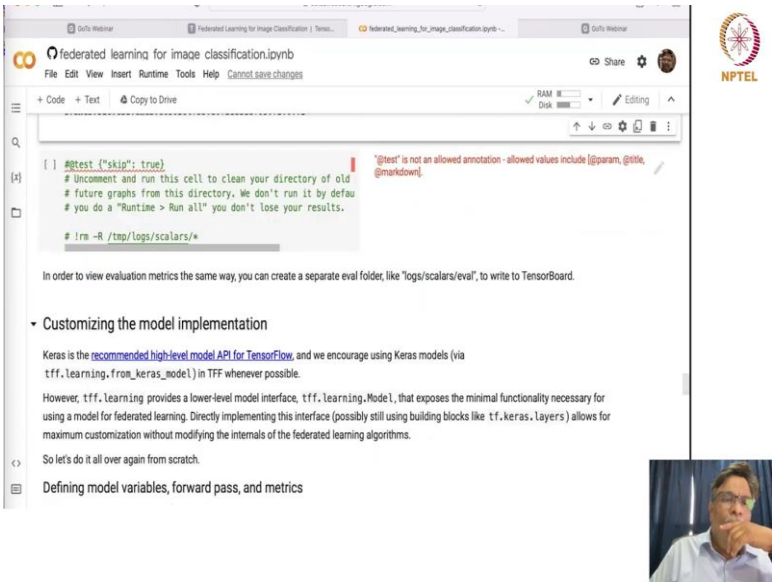
And then you can display this model matrix on the tensor board ok.

(Refer Slide Time: 23:07)

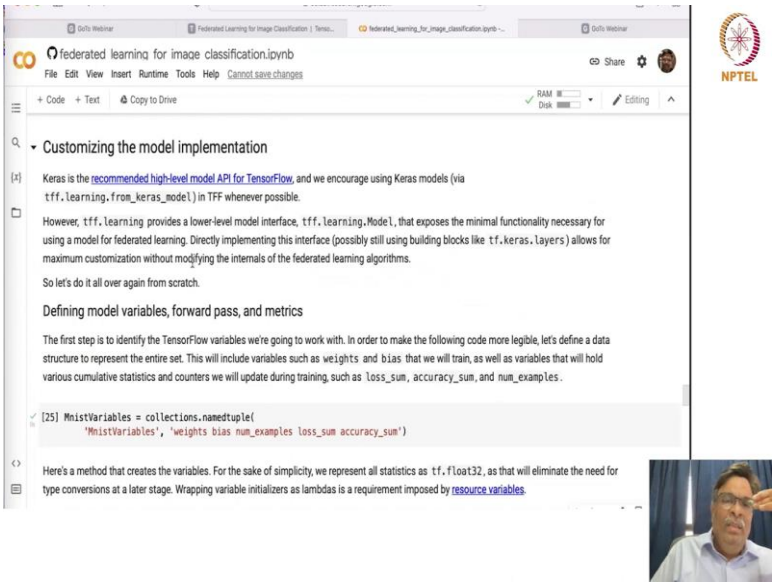


So, here the tensor board we will see in the end right, because since you are working on Colab. So, again I do not want to go now into tensor board log in and everything, but some thing here you will be able to see ok.

(Refer Slide Time: 23:36)

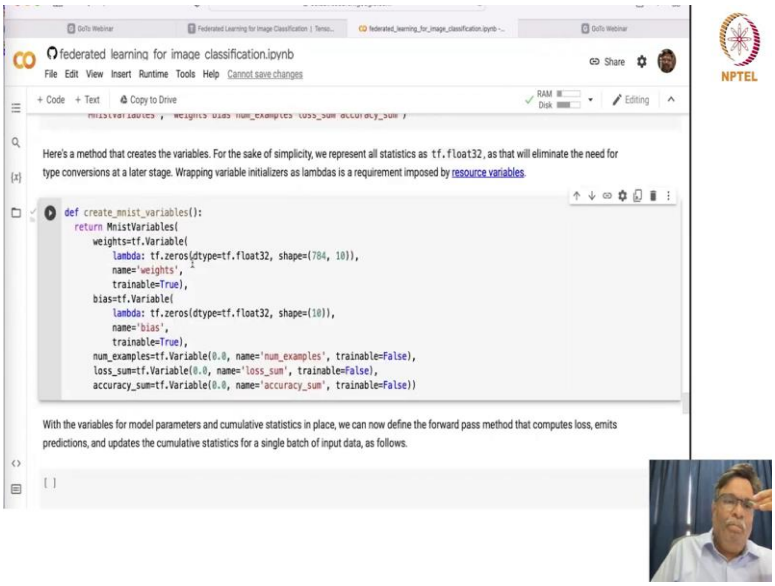


(Refer Slide Time: 23:37)

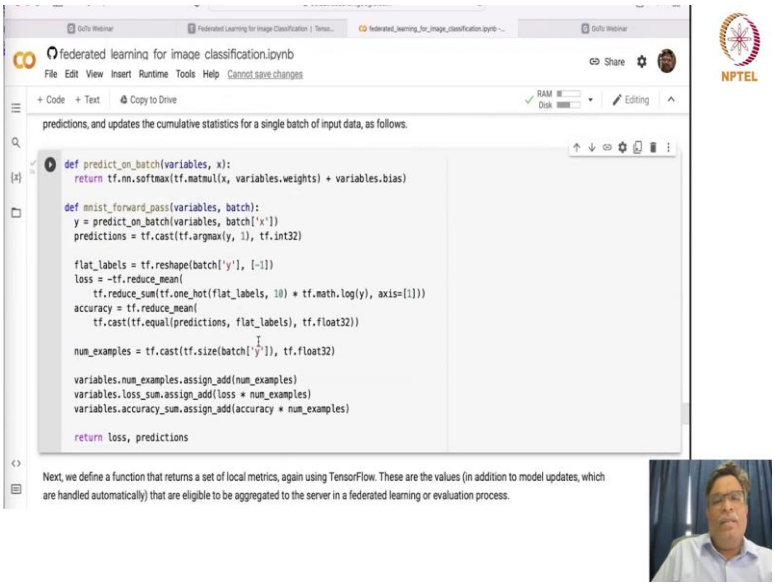


So, you actually can go deep into it, a lower level model interface wherein you are trying to actually go at the variables level ok. That you can create variables that basically means you can actually customize your model implementation.

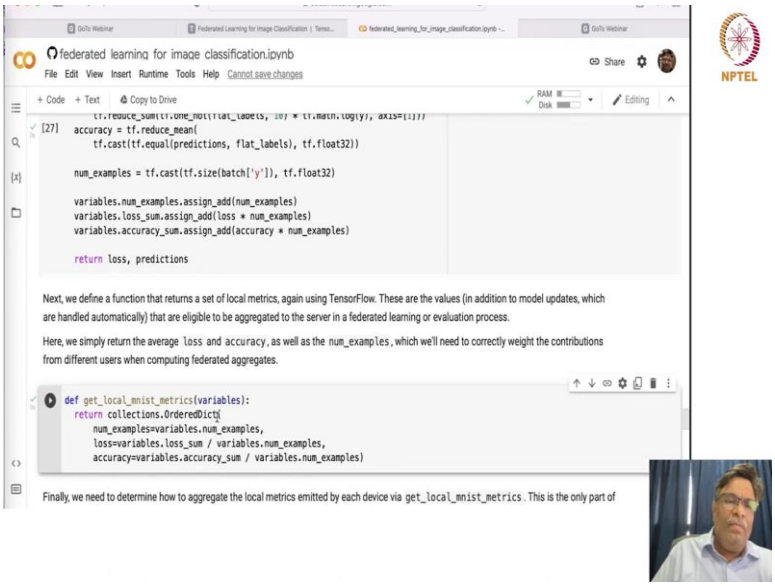
(Refer Slide Time: 24:00)



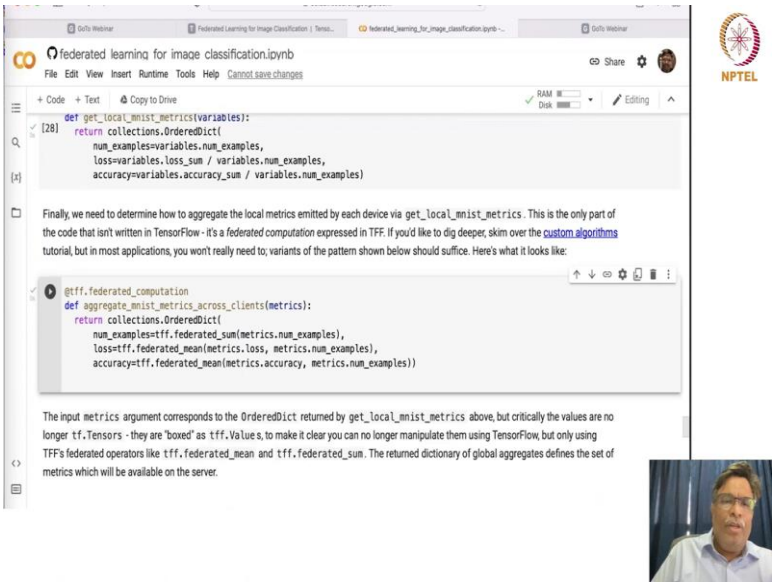
(Refer Slide Time: 24:00)



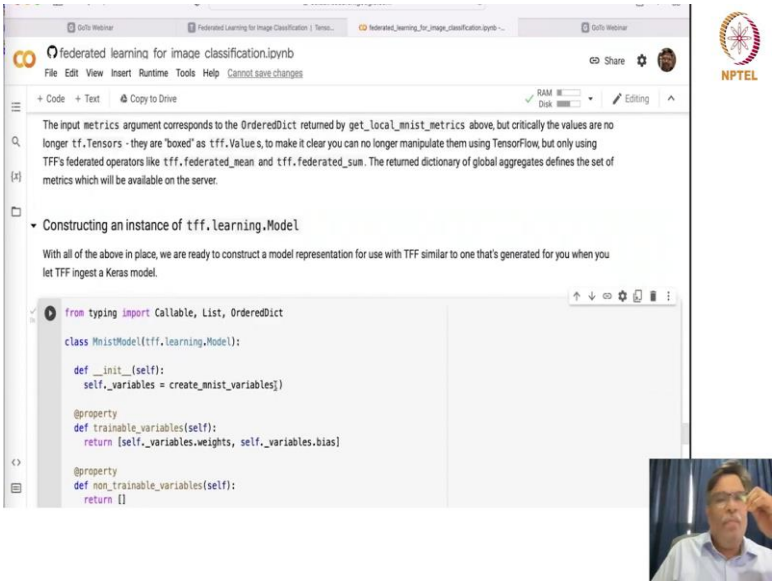
(Refer Slide Time: 24:05)



(Refer Slide Time: 24:09)

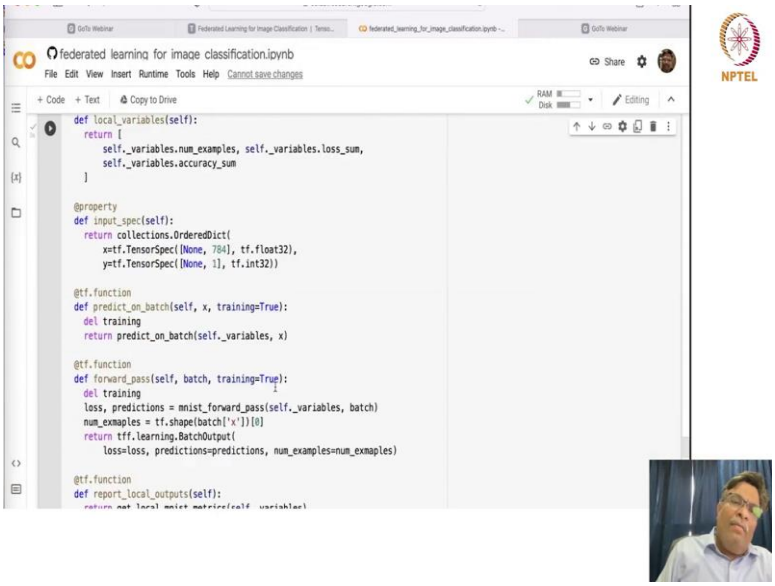


(Refer Slide Time: 24:14)



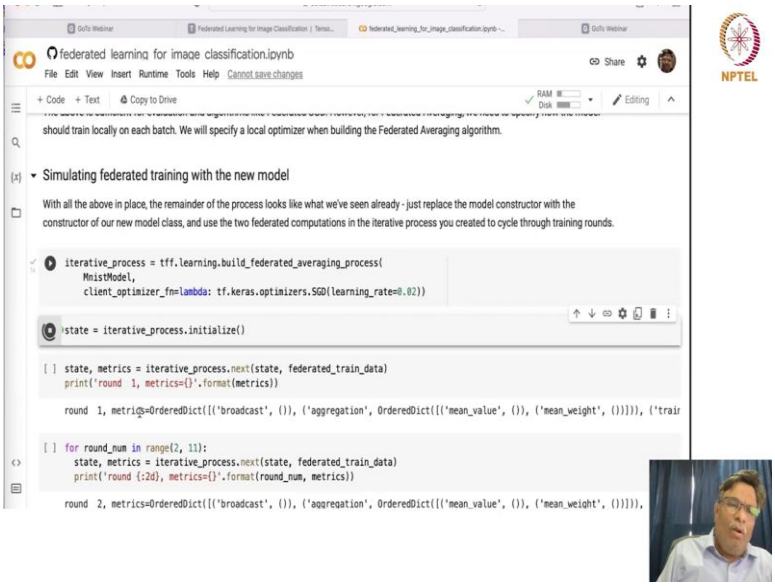
And then you try to collect the statistics, local metrics, do the federated computations and then you can use ok again another model representation, ok. Instead of the general KERAS model which we did.

(Refer Slide Time: 24:39)



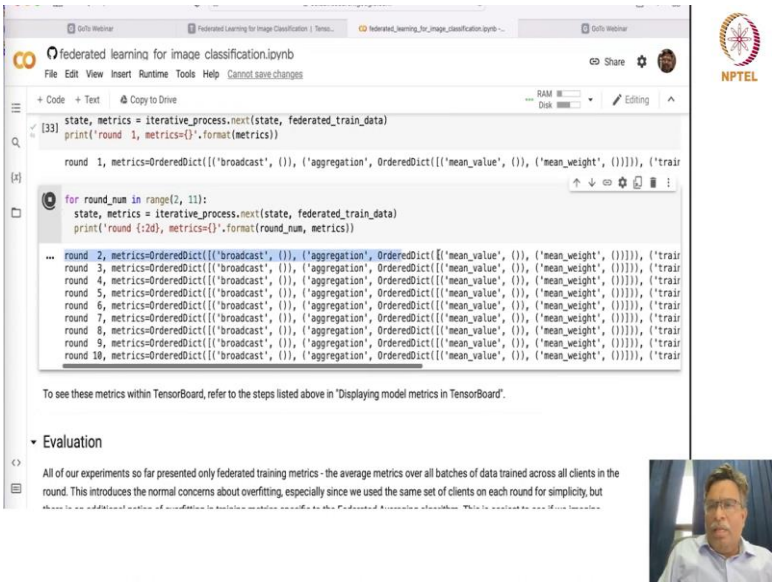
So, what we are trying to do is you can work at different levels of complexities right.

(Refer Slide Time: 24:46)



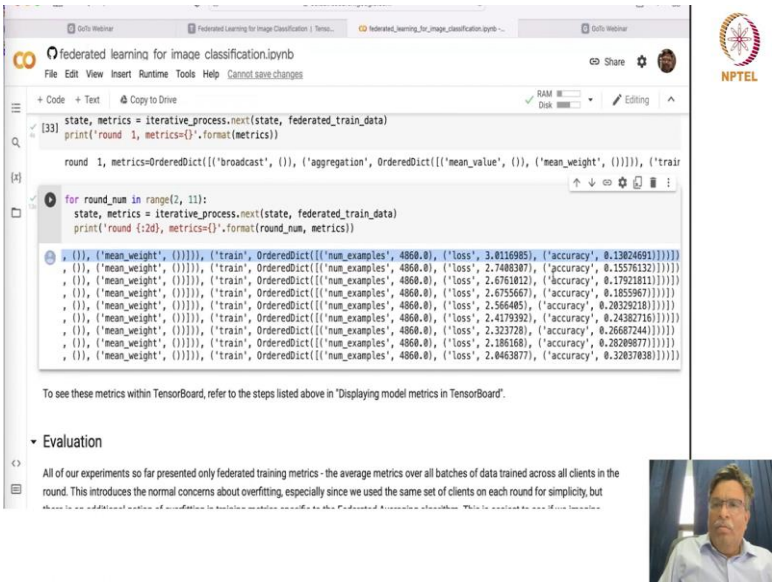
And you can make your own model instead of using a standard TFF KERAS model ok.

(Refer Slide Time: 24:50)



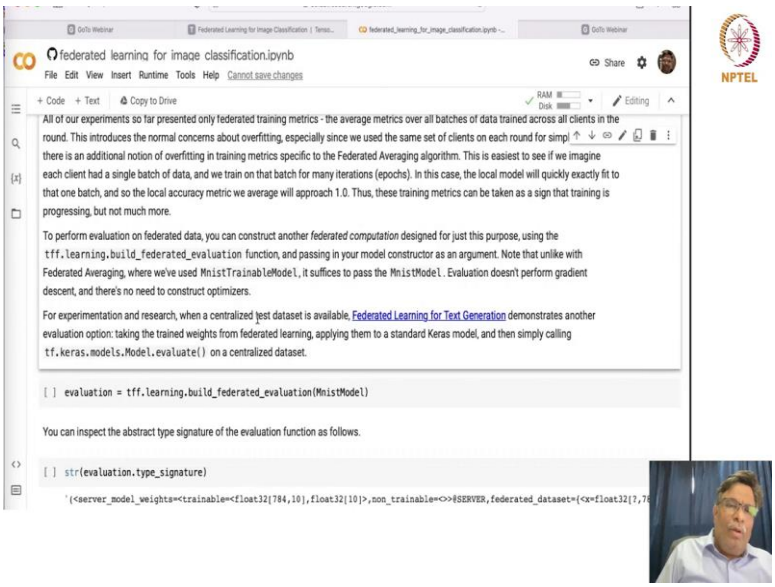
So, after training right here right, you are going to actually get after each round these matrices ok, these are metrics right.

(Refer Slide Time: 25:13)



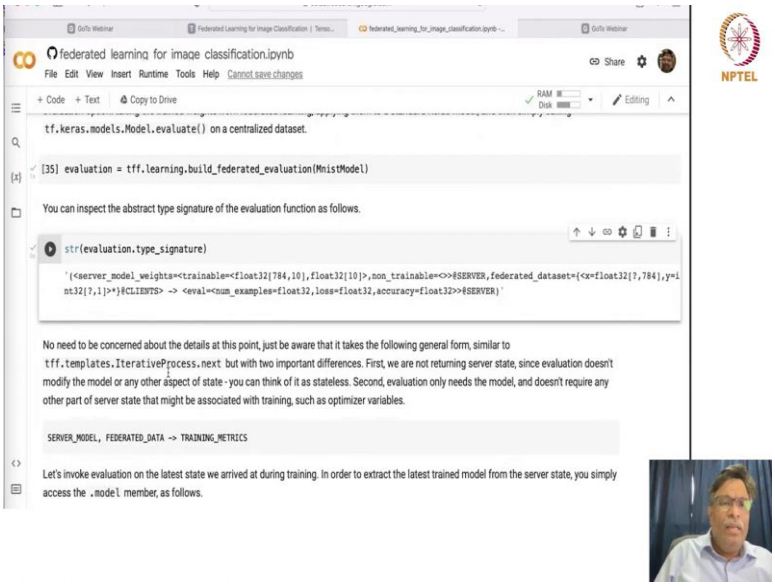
So, the details about this you can refer or we will tell you what it is, but the basic idea is this is how these metrics would be shared, right.

(Refer Slide Time: 25:22)



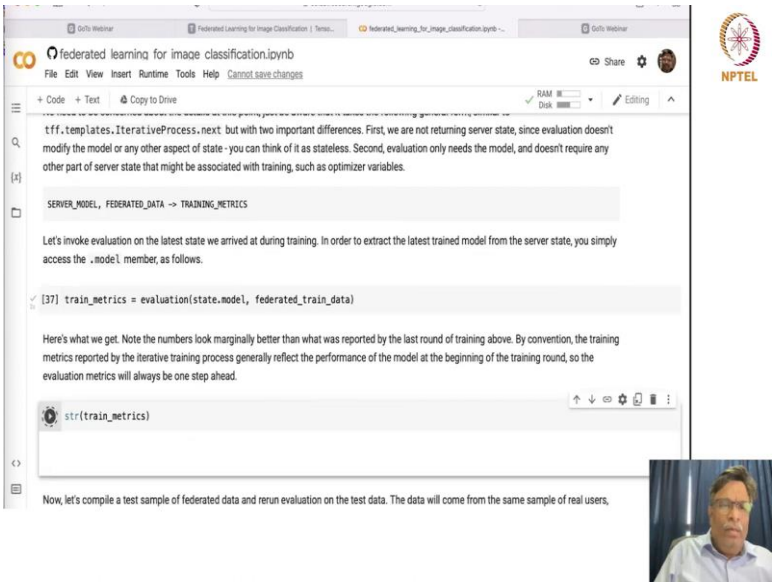
So, the idea is that once you are able to train, get those metrics back then you will have to evaluate right which is basically evaluation. Then you will have to come up with TFF learning build federated evaluation function ok.

(Refer Slide Time: 25:39)



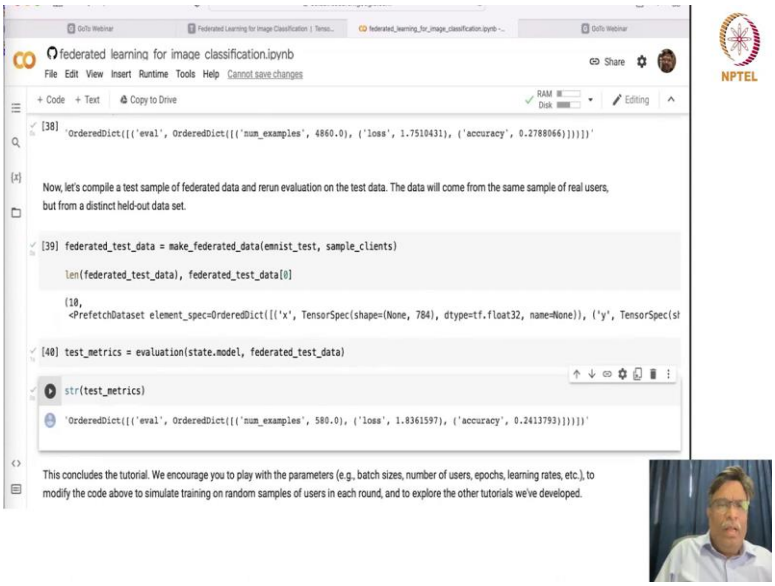
So, you can evaluate these evaluation function ok on the central data set.

(Refer Slide Time: 25:50)



And then evaluation you do it.

(Refer Slide Time: 25:56)



So, you can test the metrics on the data and then you can see that number of examples, the loss and the accuracy. So, the idea is these are a sum of and average federated learning right algorithm which you use will be combination of all of this accuracies here all of the losses here all of the number of examples here ok.

And that would be using some model right which is there in the paper you can just refer to that right then you will be able to average it and then you can change these parameters as it has been it is there in this notebook of changing batch sizes number of users epoch learning rate all those hyper parameters you can change.

The basic idea is that so one of the evaluations would require the model one of the evaluations will not require the model ok and it will not require any server state which will not require any optimizer variable. So, you will understand this that here this particular type of what to say server state information what to say exchange right is just for training metrics. Whereas, there is another type wherein here you are going to actually get ok so many of these things.

Wherein your server state and this changes and you get server state along with variables as well right apart from metrics right. So, these are certain things which is there. So, I had actually three more examples. So, maybe once I cover tomorrow better approach using something called as NVIDIA FLARE maybe we will do this or we will continue this and extend that maybe by another 15-20 minutes to cover that as well right.

So, that is what I am thinking and yeah that is it. So, any questions if you have you can ask me or you can try other three of the labs which are given here tonight and maybe tomorrow maybe we can do it a bit fast right so that you know we can cover other things as well yeah. So, that is it from my side.