Introduction to Medical Imaging and Analysis Softwares
Professor Debdoot Sheet
Department of Electrical Engineering
Indian Institute of Technology Kharagpur
Module 3
Lecture No 12
Random Forest for Segmentation and Classification

So welcome and so in continuation of our last topic which was on decision trees we would now be working on the subsequent supermodel called as random forest and this is where we take a lot of decision trees not just one single tree are created. So I understand that all of you have covered down decision trees and have write down basic experiments on how to use those decision trees based on the codes which we have already released on the extra materials.

(Refer Slide Time: 1:00)

Now as we go down to random forest, we would now be making use of all the concepts you had learned in the last lecture. So now the first concept which comes down is we knew how to create a decision tree given samples. But the question is at over here the point is that if I am using the same number of samples then, I am obviously going to end up with the high probability of generating the same decision tree, so there should be some way can I actually decorrelate them. One point where the decorrelation may happen is actually because I am randomly selecting the number of split points at particular feature.

So there is a slight chance that some different split points might get selected and you might have a different topology of the tree coming down. But to make it even much more precarious and from the prospective of actually looking into locality and globality together we actually

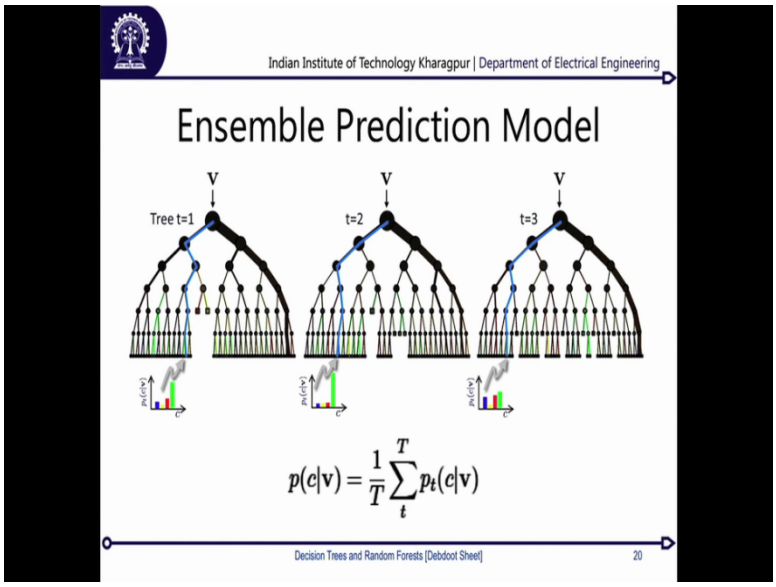
do randomize sub sampling or what is called as a bootstrap aggregation over the feature space itself.

So instead of growing each tree using the complete training feature space, we would be taking only a subset of these samples on that feature space. So it is not a subset of the features but it is a subset of the samples. So if you typically look over here in our example we have this training set where all the samples are marked down over there. So we did not associate any class label by a color over here instead of that what we have is we just put down different colors for the samples which were used to train three different trees.

So there is a red tree, there is a green tree, and there is a blue tree, ok. So our forest as of now is consisting of three different trees. So all the they look somewhat similar in a graphics over here, but the amount of decision associated at each leaf node for the same sample going in to them would be different.

Now this concept is called as bootstrap aggregation and it is not so hard to do, actually you can just do some sort of a randomized recalling sampler over there and you can choose down random number of samples, put them into small bags, you say that this is bag one for creating first tree, there is bag to for creating second tree and that is bag three for creating the third tree, ok.

(Refer Slide Time: 2:56)



So from there once we have the tree created we need to look into how it will do a prediction. So when doing a prediction you is not going to push your test sample through one of the trees, you would actually pushing your thread samples through all the trees. Now for each

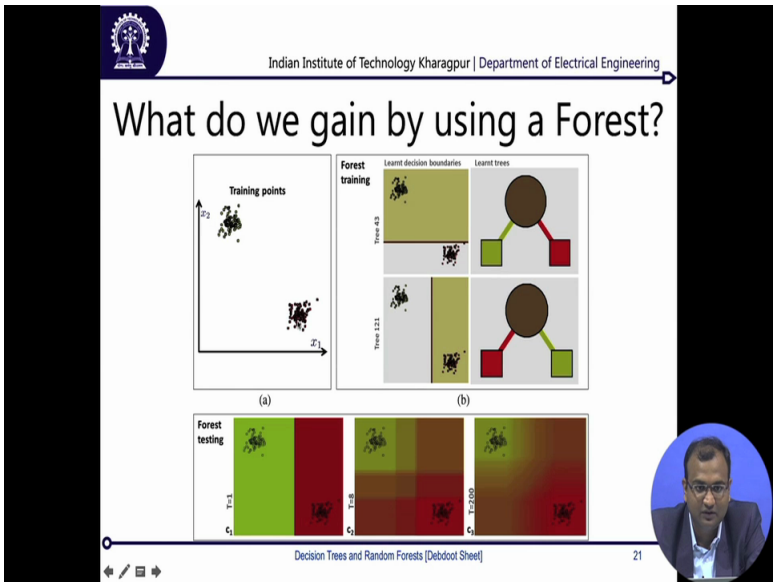
tree you would be having certain amount of decision say for this first tree I get down that my green class is very high and my blue and red class are somewhat substantially appearing over there, ok.

For the second tree I push down the same sample and I would be seeing that my green class is very high and all the other classes are much lower than in the first tree, ok. So this looks somewhat like a much more reliable decision to me, but we do not know as to what is reliable. On the third one when we push down, we see that the green class probability has gone down somehow; I mean this is saying that it is not so much on the green class.

Now in one way you can obviously say that let us go by the maximum voting, I take whichever is the maximum probability, chances are you would make a lot of errors by a that point. So instead of that we go by very democratic method, which is let us say that each tree has a same say each of them is equally wise, so just take an average over each of them.

So what we do is the posterior probability of getting a class C, given a sample V a sample feature vector V = the average over the number of trees which you have. So you have T number of trees for each of the trees probability. So Pt is the posterior probability of one tree for a class C given a feature vector V and I am going to do this for all the classes over all the trees.

(Refer Slide Time: 4:52)



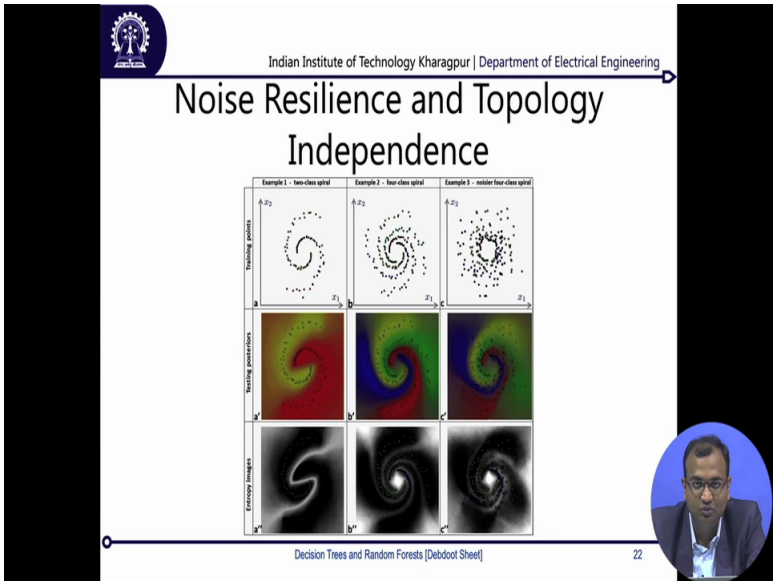
So this average value is what we are going to get as the posterior for one random forest itself, ok. Great, now what do we gain by doing that is a major question, right. So I have a few generic examples as to what we gain by using a random forest. Now say that we had just

these two types of training points over there. Now if I am doing a simple tree, then I can have this as one split, there can be one of this as a splits over here and there can be n number of such splits coming down over there.

Now if I am using only one single tree you would see that using only one split I would get down a very binary classification coming down over there. But the question is that over here and here I never encountered any sample during my training. So actually that is a grey zone, that is where we are when we are not very confident of which class it belongs to however, we are just telling over here that it is very confidently into one of these classes, right.

So instead of that if we just increase the tree. So this is an example with 4 numbers of trees and you would see that the probability gradually blends between these two surety zones. We increase the number of trees to 100 trees over here and then we see that the probability has very gradually blended. So with increase in the number of trees you would see you are going to mimic the actual uncertainty space which would be associated if there was a human taking this decision or which would be a much more causable solution for taking those decisions.

(Refer Slide Time: 6:03)



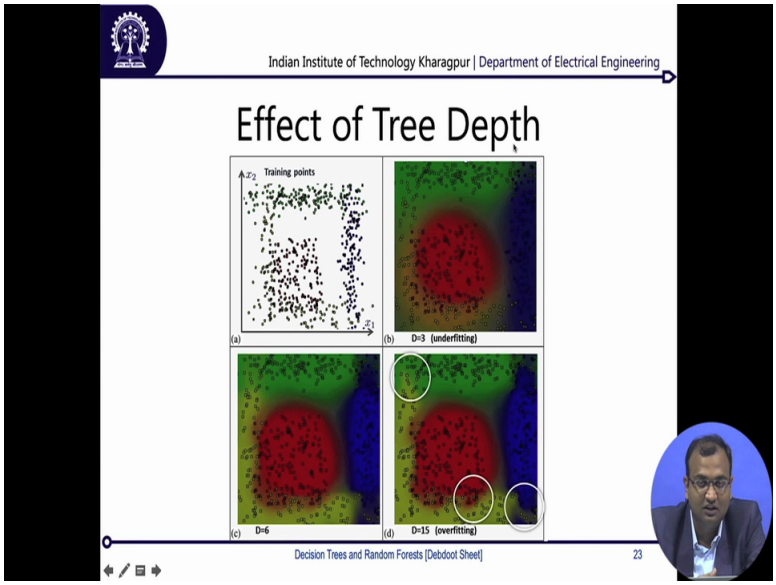
Now beyond that there is a concept called as noise resilience and topology independence. So one beautiful thing if you might have ever read about linear discriminant methods where you try to ferocious discriminant analysis or you might have a support vector machine, where there are just straight lines and there are very sharp boundaries between points is that you need to understand the data topology.

So if whether the data is linearly separable by a straight line or whether it is separable by some sought of a electrical curve this is what you need to know prior hand. But for trees you do not need to understand that, just one thing which makes a topology independent which is you can just have the data given down over there and it will figure out what kind of a topology the data is maintaining.

The next part is that it is also resilient to noise which means that if you take this particular example over here where we just have 1, 2, 3, 4, 5 classes and each of them is in a spiral, here is also the same example of 5 classes and in spiral except for that noise has been added. So it is basically this one is an example of noise added to this particular version over here.

So if you look at it the way both of them have learned is even the noise (())(7:09) has learned it much more easier in a much better way than we would ever have been humanly possible to decide for us to what this topology is existing on noise (()) (07:18). And with that we look at this entropy, which is just looking at how pure the decisions across each of them are spread over there, you would see that even on a noisy data it has very high entropy and there is a lot of purity existing on the decision split. So this is a beauty which you gain with using decision forests rather than using one single tree. The whole democratic voting plays a major role over here in order to prone down and clear of the whole kind of de-cluttering clause by all of this data.

(Refer Slide Time: 7:56)

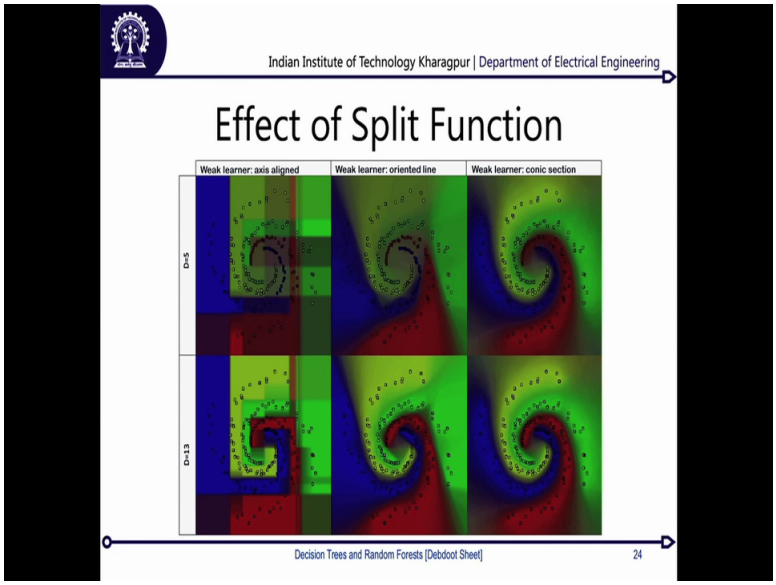


Now there is another concept of depth which is how deep should a tree be, so we did ask answer this question in the previous lecture which was I can keep on growing till I reach only

one single sample, which is not so good in idea because for a 1 megapixel I might end up having a millionaires such leaf nodes. Now here I would be asking this question if I am growing too large or I stop down at some point. So I can say that I will not grow beyond the depth of 30, or beyond the depth of 40 and what will be the affect. Now, this was a very classical example which was taken down as to residing the depth till 3 which was doing an under fitting very bad under fitting on the data.

To a depth of 6 it was doing a very good fit over there, when it was extended to a depth of 15, which was a forceful expansion so you grow all the nodes till the depth of 15. You would see that there is a lot of over fit happening over there which is not expected. So this is a tendency which needs to be taken care of and if you just do a normal stop criteria based on the number of samples coming at a node or some sort of a purity condition then you would most slightly not be doing an over fitting over there.

(Refer Slide Time: 9:04)



So that is a very important thing to be kept in mind when putting them into practical applications. Now from there let us look at the effect of split function, though we did read about that there are access aligned splits, then there are oblique splits or oriented lines and then there can be some sought of a polygonal split or conic sections. So this is an example of the same spiral data over here so which is also called as a Swiss roll data.

So on this one so Swiss roll comes on a fact that you have these pastry roles, Swiss pastry roles which is a layer of cake and there is an icing sugar and there is a cream and again a layer of cake and then everything is rolled together. So if you look at this particular example

you would see that each class is sort of spirally within the other class and it appears as for the cross section of a Swiss role.

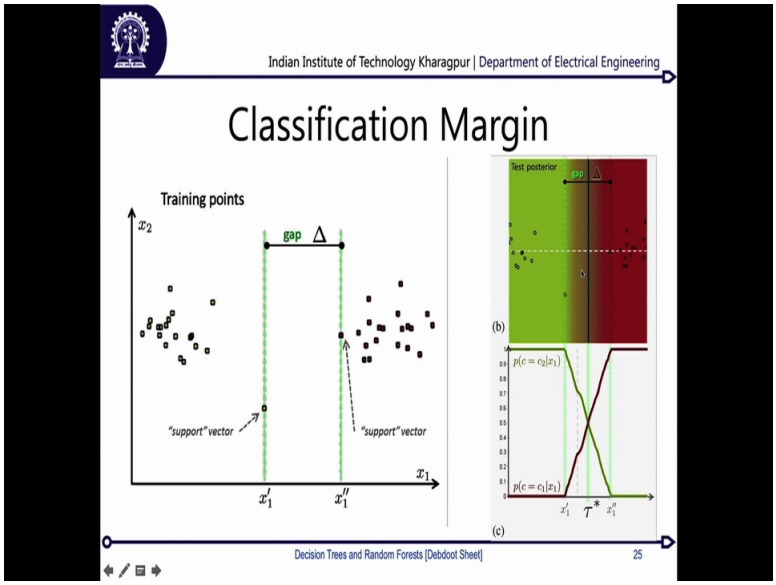
So on this particular Swiss role dataset people had write out this particular example and so if you put down oblique splits versus if we put down axis aligned splits versus if we put down polygonal splits. Now, as you see with increase in depth of tree and increase in number of trees, what would happen is that these orthogonal or axis aligned splits would eventually be able to produce as good a result as you would get down with this polygonal split, this polygonal split.

But with the polygonal split the number of free variables you have to define and optimize is much higher. So for a axis aligned split what happens is you can define an line equation something like this, Y = Y1, or Y = Y2, Y = Y3. For a oblique split you would be defining something like Y = MX + C, so you have two different factors M and C these two to be optimized over there.

So you can have multiple numbers of oblique lines but M and C has to be optimized over there. If I do some sort of a quadratic equation, I will have more number of factors over there. So I can have a quadratic equation like something like 1 = X square + BX + C, so I will have three factors to be optimized over there.

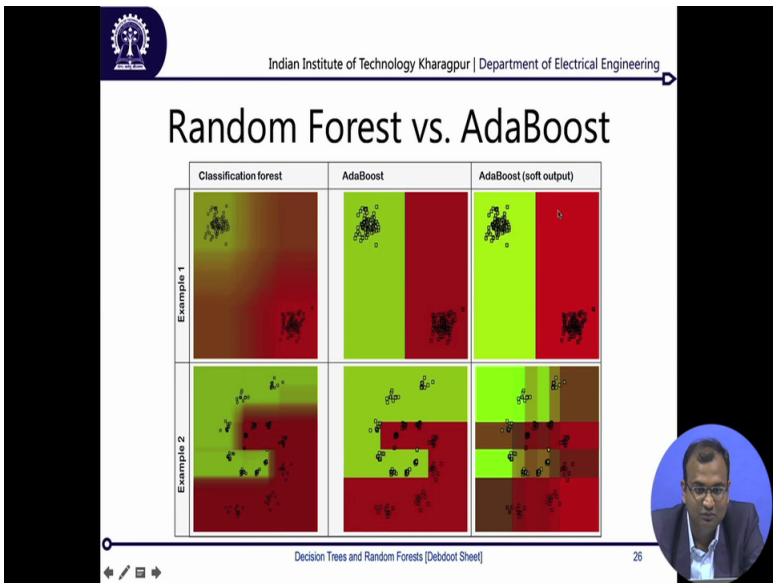
Similarly the number of factors which you keep on doing is what becomes complicated and that matrix for optimization on the objective and finding out which feature and which particular value of that feature that also keeps on compounding equivocally over there. Now from there interesting observation in terms of classification margin whether it is just a jump to the left or right, whether it is binary classification or it is doing a very smooth classification.

(Refer Slide Time: 11:18)



You would see that with decision trees if you have more number of them on a random forest you would not be doing a binary classification but you have a very smooth transition, which is the property which we had discussed and this is again relooking as to how will be the classification margin looking at . If you look very carefully then these margin points over here which are an equivalent to what in support vectors machines would be called as support vectors, so you would have very pure class representation and those margin points.

(Refer Slide Time: 12:00)

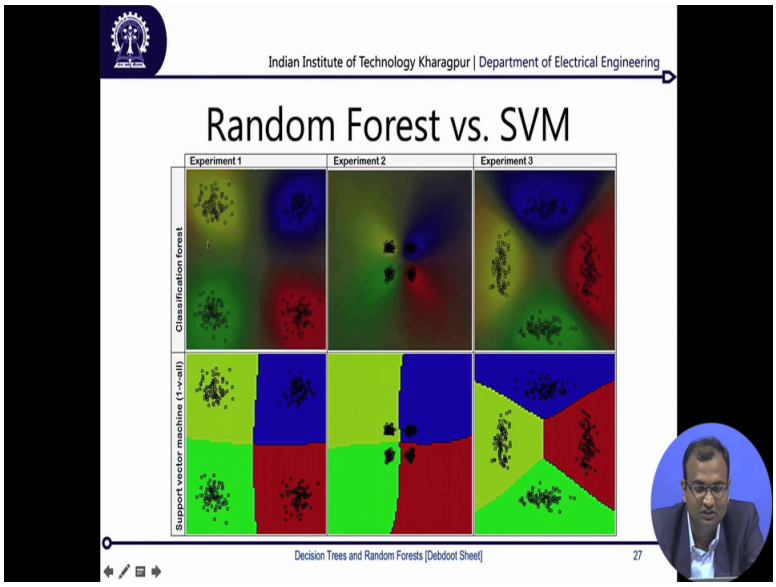


So something which is just between these gaps of the support vectors is where you have a gradual transition in terms of your probability. Now, there is a famous comparison between Random Forest and AdaBoost, now if you look at these examples over there. Now AdaBoost

does not fair that good in terms of doing a gradual transition in the unknown territory as compared to a Random Forest.

And severely is lacked when you are trying to do something on the Swiss role dataset or these kinds of spirals and because this does want you to have information about the topology and the kind of splits you are using over there. So compare to that, Random Forest is obviously much better in terms of topology resilience.

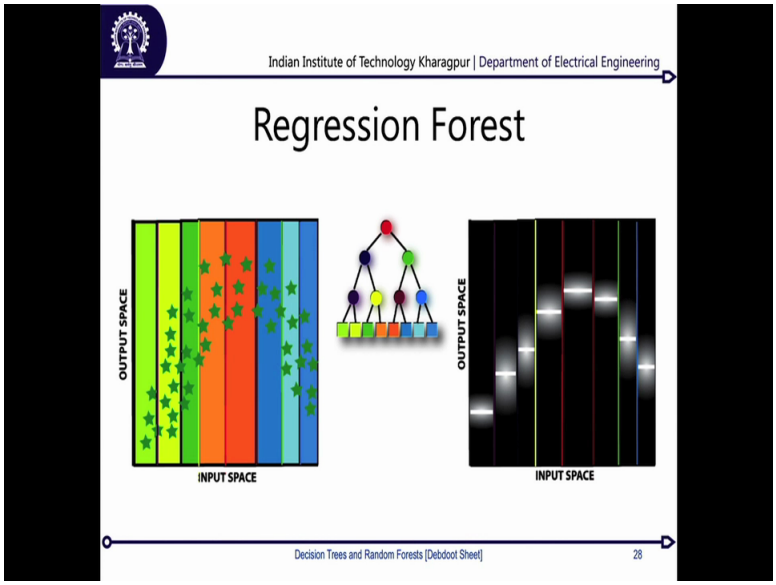
(Refer Slide Time: 12:29)



A classical example of trying to compare them with support vector machines on multi class you can just look at these 4 points spread out data. Now if you look at this particular example, where they are equivocally spread and you would like to have equal margins now. And the support vector it often comes down in this sort of shattering criteria. Now the problem is that nobody knows that whether this is a definitive criterion, there might be a condition where I might have something like this and straight line joining in between them.

It might be quite orthogonal to what would this is at 90 degree rotated version, so they are some curious observation which people have found out in doing experiments around. So the question which comes is that like we are speaking a lot about classification using this Random Forest model. The question is there is obviously another problem which we touched at starting of the decision tree which was about predicting the age of an ant by looking at the picture of an ant which was regression problem in itself.

(Refer Slide Time: 13:28)

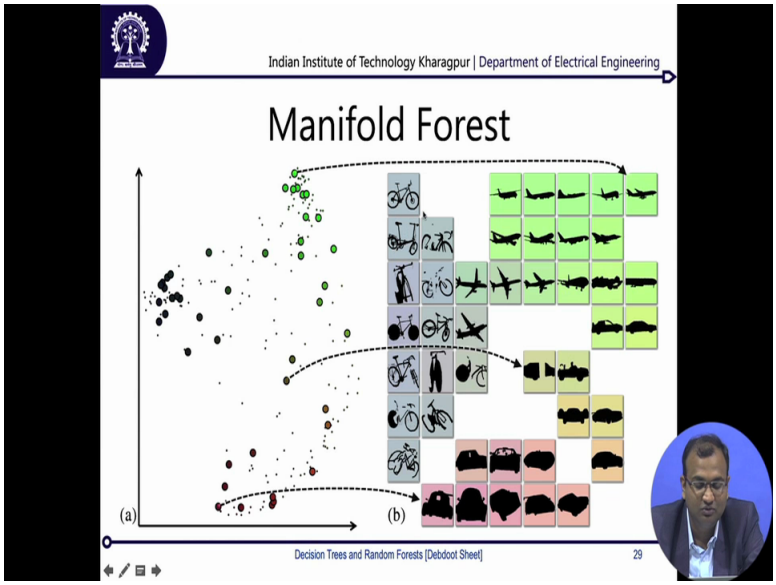


Now regression problem in terms of a Random Forest is quite easier to solve, what you can do is since there would be a variable on which you are going to regress. Now you can actually split it down into piece wise linear function. So say age is a variable over there on which you are going to regress. You can divide an age into gap of intervals of one year, so the chance that you are in an error is basically half a year of a problem.

So over here the question which comes down is that if somebody is say 6 years and 11 months, you might predict them to be 6 years where as they are very close to 7 years. Now this is the only erroneous condition which might happen over there, but you can choose these boundaries affectively and the number of step functions over there.

Now once we do what we assigned is each of these categorical variables which is my pieces of my linear function, they are assigned as one class. And now you actually end up doing a classification problem and just finding out which is the label which I am predicting out of my final node. So that is how we solve a regression within a Forest itself and it is a very straight forward model, you do not need major changes throughout the whole computation process as such.

(Refer Slide Time: 14:38)



So from there an interesting fact is that they are very useful for manifold learning where manifold is basically, fold within fold size it is defined. So Swiss role is one sort of a manifold, the pastry kind of manifold, you can have similar kind of different manifold, so this was an example where people trying to look into appearances of different objects and they were mapped into different points.

So if you are trying to do say a nearest neighbour classification it would not be so because if you look at these bicycles, there can be so many types of bicycles and there would be bicycles over here till here. And this is a point where it is very close to car, so if I have a bicycle point over here, it would for a nearest neighbour it might actually end up getting here then going down over there. So these are some interesting facts which happen within a Random Forest and it can learn these kinds of manifolds because it is independent to the topology. You are no more restricting yourself to the topology in which you are learning.

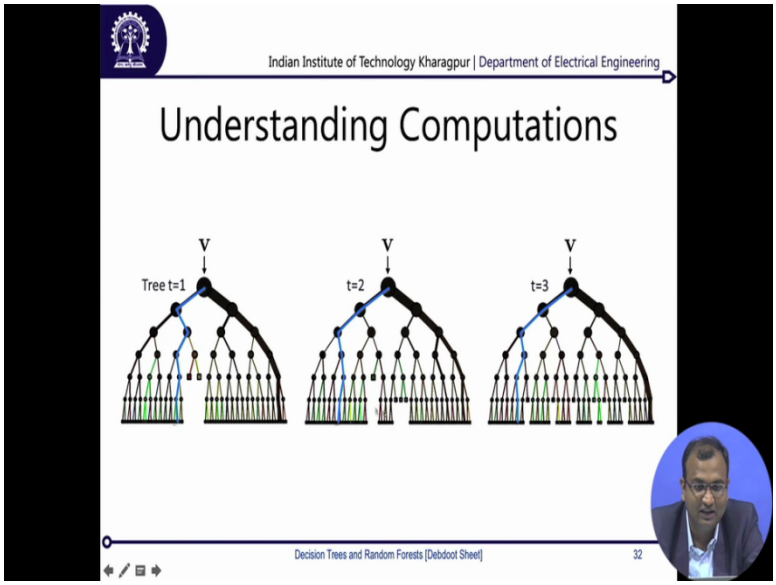
(Refer Slide Time: 15:38)

So and coming back to this point which I was telling about how this again goes into power was because Kinect for Xbox 360 uses this body part classification, have a look into this interesting paper by Shotton from Microsoft and this is their particular paper which says about how a consumer grade device is actually using them and is the main brain power between your entertainment to a very practical scenario.

So these are some stuffs which we have I am discussing in general, I am not very specifically focusing on medical image analysis because in application scenarios is where we will be drawing all of these again up and taking them as to where they go into our use. So from there there are critical challenges where we will have to really look into it and they are what I call as Engineering Design Perspective.

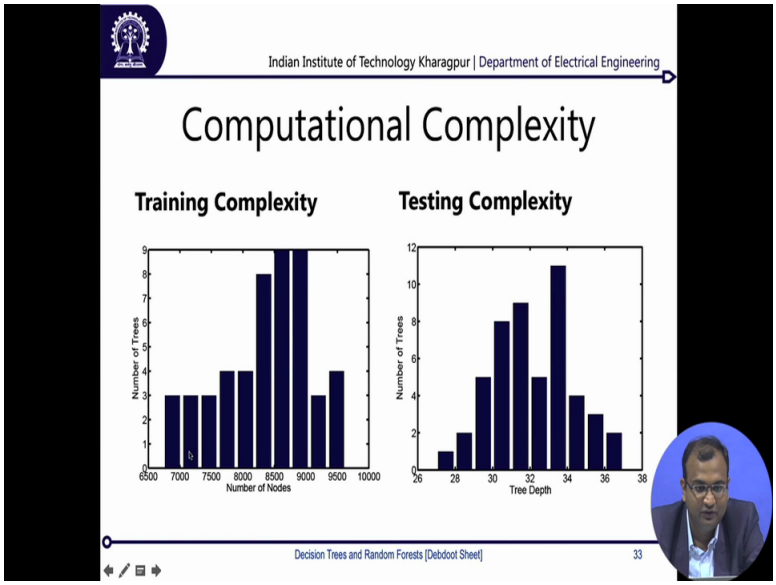
So one point is you need to understand computations within these trees in a much better way. Because if you look at a tree, so if I have a tree up to the depth of D, then the possible number of nodes which I will be having is 2 power of D that is a maximum number of nodes, which I can have at any point of time. So if I am going to create a tree up to a depth of 10, so it is basically 2 power of 10 which is almost 1024, that is 1000 number of nodes in a tree if it is just for a depth of 10.

(Refer Slide Time: 17:15)



If I just make it depth of 11, it is going to be to power of 11, which makes it 2048. Now the way it is escalating is really enormous it is an exponential scale escalation in the complexity of the tree and this is what we need to understand. So see that I have created down three different trees it is not necessary that every time I will be having all of these filled up. So there will be some points when I will need node purity at some particular point which is much above the depth of the tree and I would just be stopping over there because I do not have any criteria to split and go down further. So in a typical scenario you would most likely observe that the total number of nodes is always lesser than 2 power of D.

(Refer Slide Time: 17:41)



And I take this very classical example from one of the problems we were solving for tissue characterization where we did observed the training complexity which was defined as the number of nodes the number of trees with the different number of nodes. So it ranges something between from say what 7000 till 9500. So it was basically the number of nodes which ranges over here and we by just counting how many trees did have that one, so there were total 50 trees which were grown over there. I would see majority of the trees had the number of nodes over here. Now on the other side of is the testing complexity.

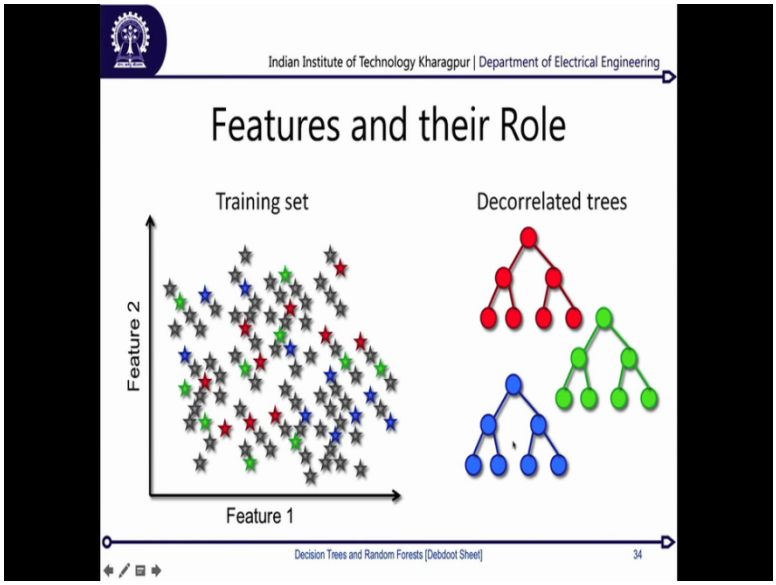
So in testing complexity in testing time what happens is you do not traverse through each and every node of the tree. Given one sample, it will just be traversing through the maximum number of nodes, which = the depth in the worst case because you are going to traverse through one node then go down to the left or right which at the next level. Then you again go down to again a left or right which is again at the next level.

So in the worst case you will start at the top node and go down to the last bottom most nodes at the last depth over there. So the total number of traverse was which it will do during testing is actually proportional to the depth of the tree. So this is an example where we found out the total depth of the tree and the number of trees which we are having. So if you look very closely I mean the largest tree was somewhere around 37 and the smallest tree was somewhere around like the average was around 32, so 32 is the typical depth you would be finding for this tree.

But then if you look at the total number of nodes which they are supposed to have, the maximum number of nodes these trees can have is 2 power of 32. Now this peak over here of 8000, 8500, 9500 is actually much lower than 2 to the power of 32 over there. So this clearly brings us to a point that we are building up trees which are really sparse, they are not completely dense and obviously they have not yet traverse.

So we did encapsulate the feature space in a very good way, but we did not need the completely dense tree in order to encapsulate the feature space. So over here you would see that training complexity which is obviously dependent on the number of nodes I am going to grow is always much larger than the testing complexity. So when you are training a tree you would require more time more CPU time to go into it, when you are predicting out of a tree it is much lesser because the number of node traverses. The number of decisions you would typically be taking is in the order of 32. But here the number of decisions you take is in the order of 8500, which is a factor larger over there.

(Refer Slide Time: 20:13)

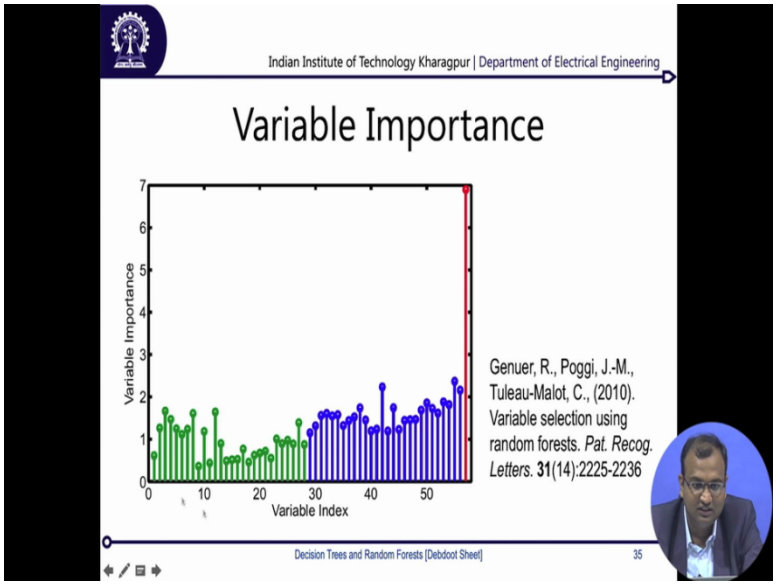


Now from there we come down to the point about features and their roles so say that I have got this decorrelated tree so from my first slide we have been able to create them down. Now what I want to look at is, say I am looking at this red tree, I have taken only the samples which are marked in red, and all the other samples are called out of bag samples. So there is a bag and the samples which are not in the bag are out bag. Now once I have created this tree, I can actually take a lot of benefit out of these samples, which are not used for creating the tree, and that is sort of my validation sets validation samples. Now what I do is I use this out of bag samples and I can shuffle down one feature at a time.

So say this has 11 features in it, so I take it has 11 features which are 11 different columns arranged over there, each row is 1-1 sample number. So it says take the first feature and I just randomly shuffle its location. So I am not touching for the same sample, I am not touching all the other features. So feature 2 to feature 10 I am just keeping it as it is in the same ordinal form.

Feature 1 is what I am shuffling along the sample space over there. So with this reshuffled version, I am just going to push it through each of this tree. Now when I push it through each of these trees, I will be getting down certain kind of a probability and certain decisions for each of the samples. Now I will be measuring out what is the amount of error I am getting in the total classification for shuffling each of these feature spaces over here.

(Refer Slide Time: 21:48)



Now, as I do that for certain features I would be getting much higher error, for certain features I would getting a much lower error. Now that would typically give me graph somewhere like this. So here we have something around 56 features and for each of them, this Y axis is basically the error which I am showing. So the lowest error is somewhere around 0.3 and the highest error which goes down is about 0.7.

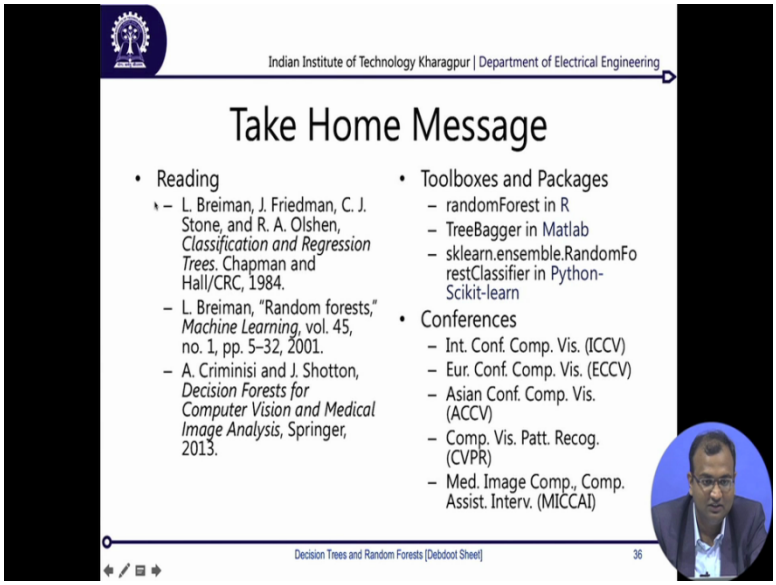
Now, can we use this for certain inference and that is actually yes. So if you look at this particular graph this is called as variable importance and you can go through this particular paper which speaks about variable importance and variable selection using Random Forest and this is where I draw you back to the concepts which we were telling as to what can we infer about Random Forests when we know about the samples present over there.

So what comes is that if a certain feature was very important over there and then shuffling that feature will actually introduce a lot more of error. So this particular feature which is the 56 feature was actually very much important in the whole decision process and this particular feature, which gets a very low value is the one which was not much playing a major role in the decision process that is why even if you shuffle it down which is just randomize our this feature the chances of getting an error is actually very low, so this was not impacting.

So this is what makes this particular breed of learning (()) 23:04) auto feature selected. So you can give down features which can also have garbage features or non-discriminated features itself within there. The learner itself is going to select which ever feature is really

important for it and discard all the ones which are not at all important going to play any significant role over there.

(Refer Slide Time: 23:45)



So this is a very important property, so in the subsequent one when we do the lab session I would be showing you how we can get each of these samples out from the learner itself. So from there, this is where I come to a conclusion and on the take home message I can basically point you to the particular books which you can read so there are three books which I would suggest for classification and regression trees is the seminal book by Breiman, Friedman, Olshen and Stone, you can also read about Random Forest from the first seminal paper which appeared in the journal called as Machine Learning in 2001.

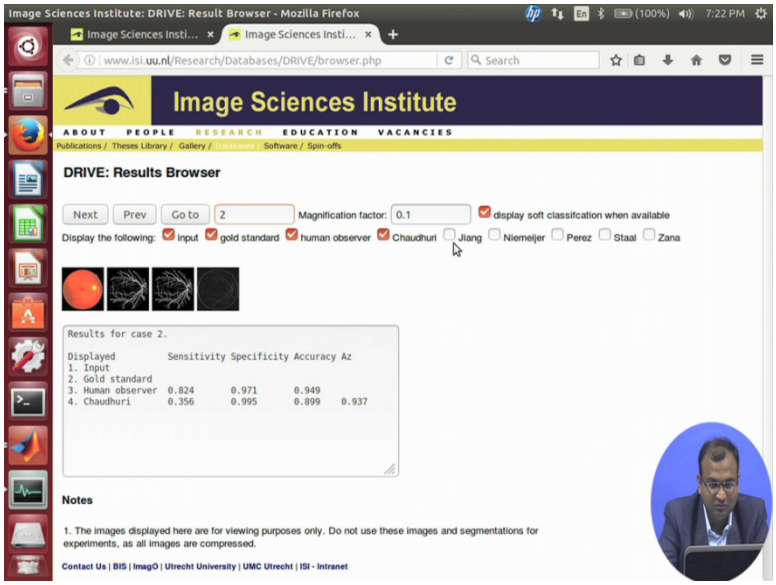
And very precise narrative and sort of from where we have borrowed most of the contents for this particular presentation and teaching today is from this particular book by Criminisi and Shotton and from Springer, so this is about decision forest for Computer Vision and Medical Image Analysis. So you get lot of very practical applications, where this has been going on Medical Image Analysis, so just have a pointer to these ones. You can implement Random Forest very easily on R, there is the command is just Random Forest in Matlab it is called TreeBagger, in Python you need to have Scikit-learn, so you will need to have access to Sipi and from Scikit-learn you can have this Random Forest Transable Classifier module created and for your in interesting use.

And please go through the papers from these important conferences, where Random Forest has something in the hype, so you can go for papers around in the year 2010 to 2012 and

2014 and this is when the (()) (23:03) was really going on them. So you would be finding a lot of these interesting topics from here over there. So with this I would end down our topic on decision forests and on a conclusion from decision trees and Random Forest together.

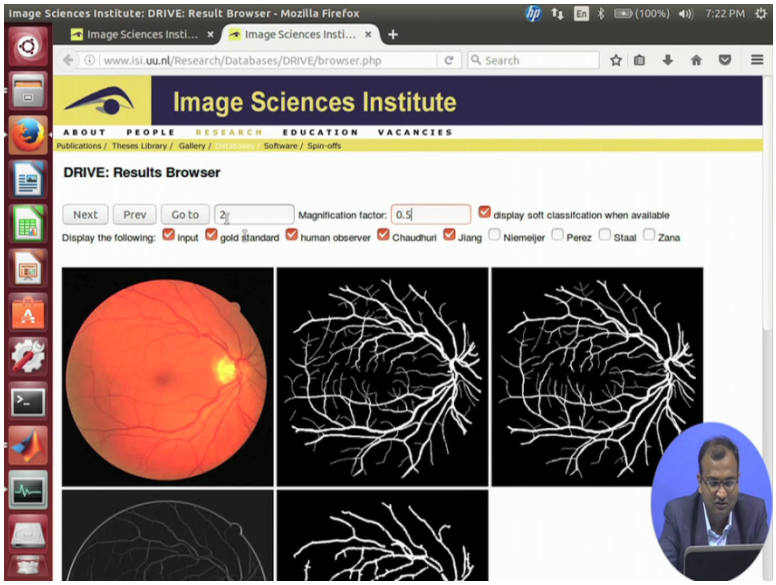
So now is one of our life lab demonstrations and I will show you a very simple example on retinal vessel segmentation problem. So we have a full-fledged lecture on detailing every different methods of how this vessel segmentation problem works. But I will just show you a very simple basic demonstration of how to use a Random Forest in order to do it.

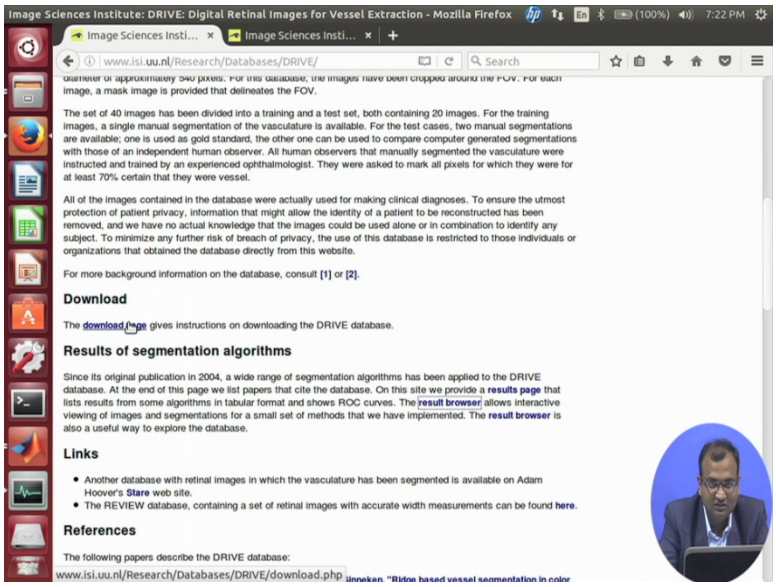
(Refer Slide Time: 25:55)



And we are going to be very-very sort of precarious over here, so that we do not waste a lot of time on training over large dataset. I will just use one single image from a very publicly known dataset. So you can actually download the whole data from the image senses institute drive reposit. So this is a repository of 40 retinal images and they have been annotated by two different observers. So you can go down onto this results browser and you can select the particular image say image number 1, image number 2 and you can check mark over here different observers and see what their findings were in different algorithms, you can chose the magnification factor make them bigger or smaller and see.

(Refer Slide Time: 26:22)



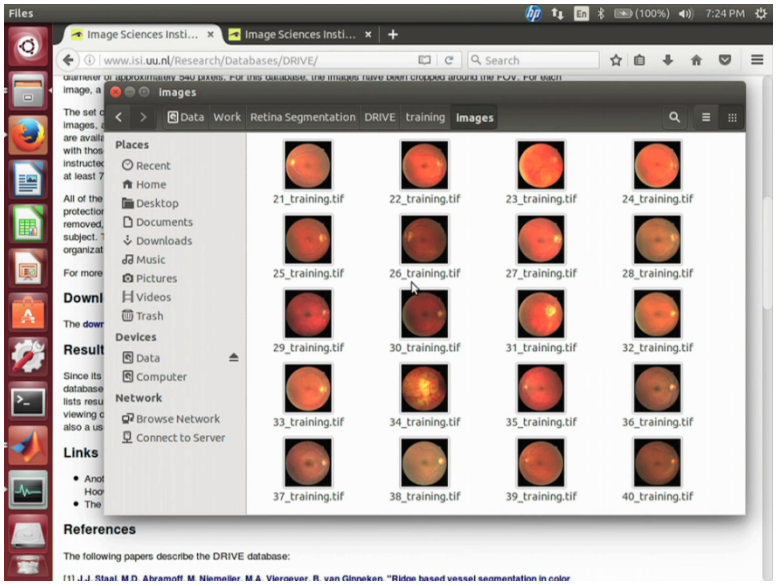


This is the interactive browser, so let us go down to the main page from where you can an access to so you need to go down over here and click on this downloads page and fill up the instructions over there completely and you would be able to download the complete stuff. If you want to look at the results you can click on the results browser and open it. So in order to save time what I have done is, I already have the results, I already have the whole dataset with me as a zip folder based on drive and there is my unzipped version.

So when you open it you get two folders one is the test, another is a train and so there are some other worker I was doing so these results are kept over there. What I would be doing is I go down on the training dataset and you look over here what you would get down is 1st

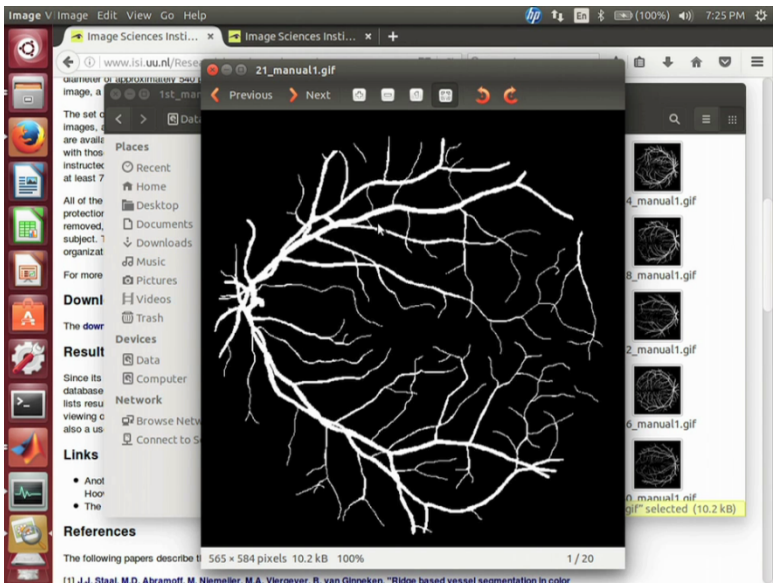
manual, images and mask. These two are two different folders, which I created during my work.

(Refer Slide Time: 27:20)



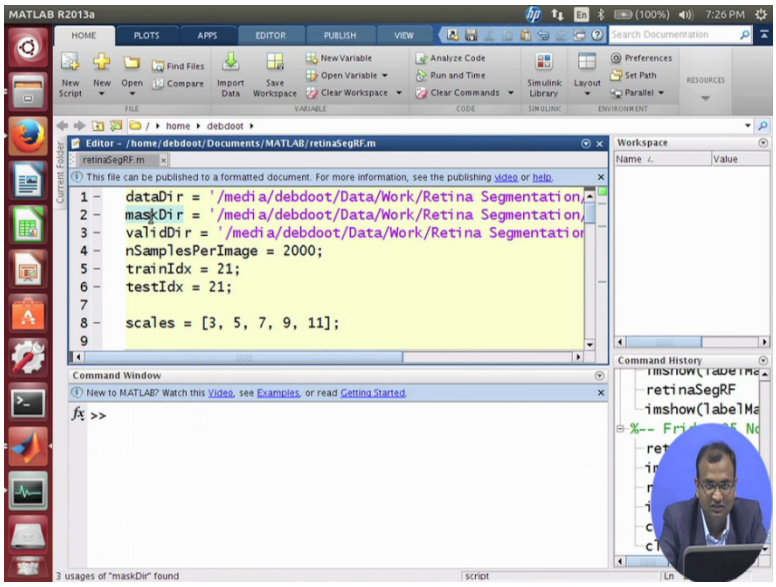
So you can look into this images folder and you can see different images over there and they are numbered from 21 till 40, which you see and you can open up each of these images and for the fun of seeing how they look at and so each of them is about 565 cross 584 pixels and they are full color RGB images. So what we are going to do over here is quite simple thing, you have these images, you have 1st manual is basically one manual annotator for labeling each of them.

(Refer Slide Time: 27:49)



So each single pixel is labeled as black or white, white belongs to the vessel region and black belongs to the background over there. Now along with that what they give is thing called as mask and this is basically the white region is which belongs to the retinal region and everything else is which is not within the retinal region because you do not want to somehow take a lot of pixels from this background and just keep on piling them within the learning framework over there.

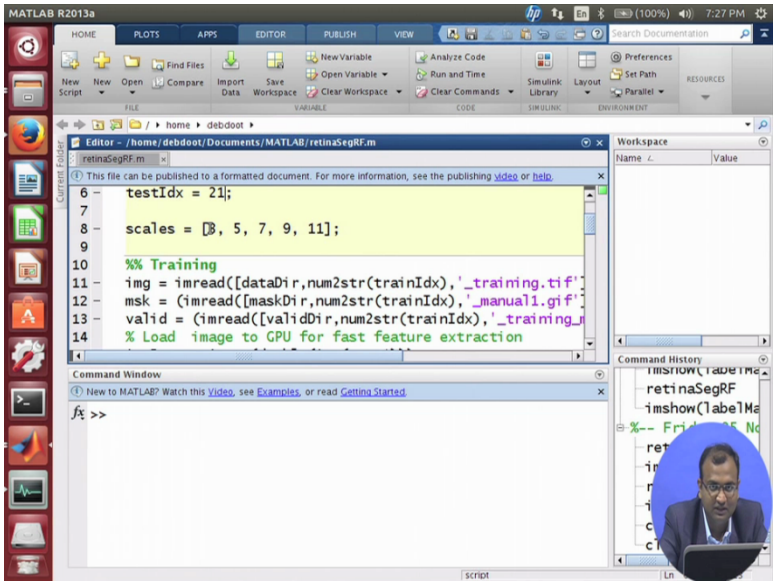
(Refer Slide Time: 28:28)



So let us get started, I have a small code written down over here will do a brief walk through and what it is doing, so the first part there is I just make down directories over there. I call as data directory, mask directory and valid. So this mask is basically the retinal vessel mask which was over there which was called as 1st manual and the valid directory is basically I am just looking at the valid pixels, I am not going to take down the other ones which are not belonging to retinal region. I was initially thinking of taking down some 2000 samples, but I would prefer taking down all of the samples within one and do it. Now I can set down a training and testing index, so just for the fun of it I am putting I am going use the same image for training and testing.

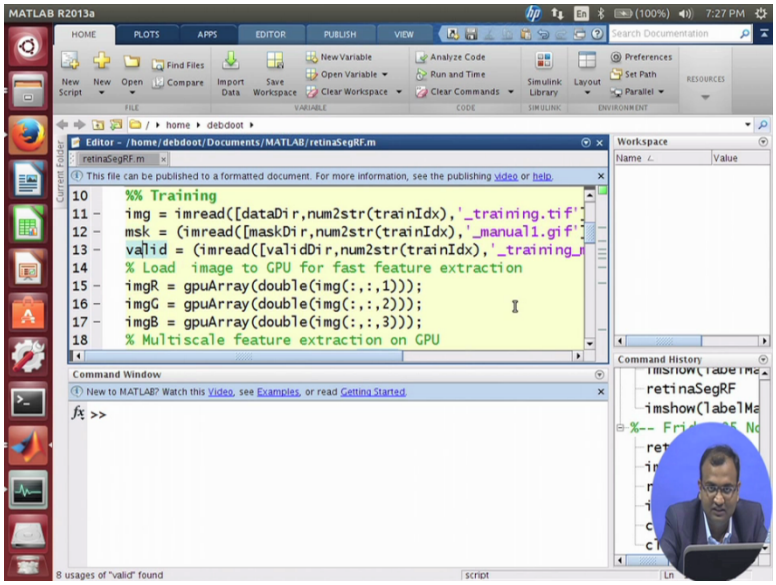
You can loop over and do all of these, which we will detail in the later on exercise. But let us say that I am going to use only one image in order to learn and I will see what will be the prediction on the same image if I am testing it on. I am going to use, so these are certain scale definitions over there, which comes down from the feature extract as which we had learnt in the earlier classes, which was about you can use variances or mean in order to find out different features.

(Refer Slide Time: 29:31)



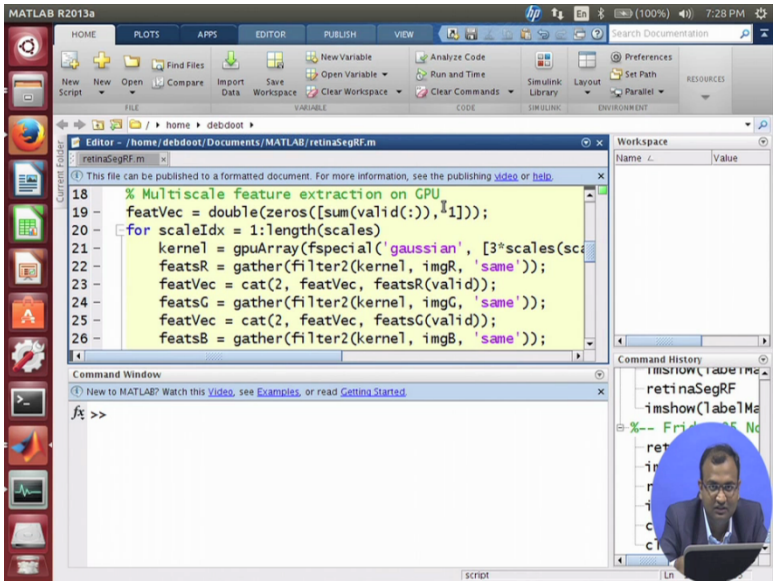
So I am just going to use mean as a feature, so I just use patches here, 3 + 3 patch or 5 + 5 patch and for purpose, I am not going to use very symmetric patches I am going to use a Gaussian weighted patch over here. So I will initially take a Gaussian with variance of 3 then variance of 5, then variance of 7 and eventually till 11, and at each of these scales I will be just (()) (29:47) out the mean and that is my feature vector. So typically you can look that there are 1, 2, 3, 4, 5 different features, which I am going to find out and each of them is getting computed at each color channel. So there is red, green, blue, 3 channels on the color image, I am going to have 3 into 5 of them, so there are 15 features which I get down. Now along with that what I would do is now I start my work so I am going to read down my image.

(Refer Slide Time: 30:18)



I am going to read down the labels over there in mask and the valid regions of the image as well. So what I do is after I read them these since these are binary ones, now that image over there would typically be in a 0 and 255 range, I just want to convert into trues and false, so I use this operator. Now, I will be making heavy use of GPUs which I have so if you do not have a GPU do not panic around within you do not need to do this part of it, so you can basically (()) (30:43) and teach of these. Otherwise, what I am doing over here is basically load down each channel as a GPU array and I am transferring it from a CPU memory on the GPU memory.

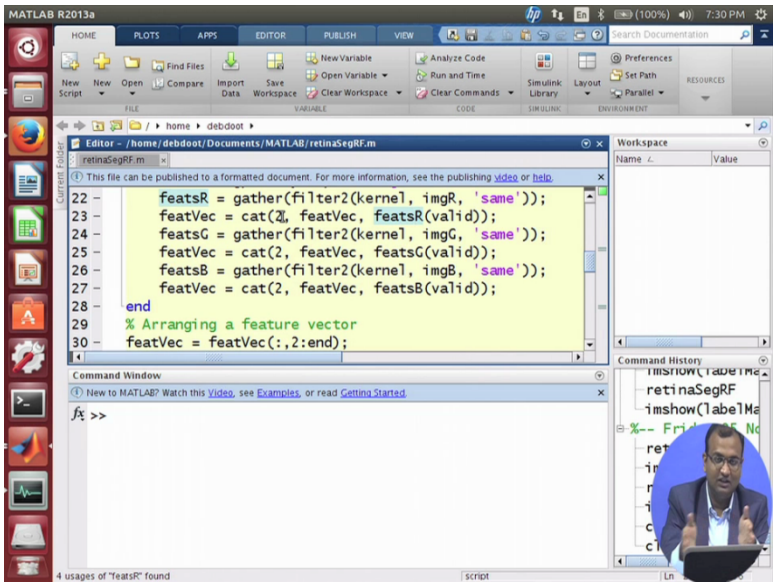
(Refer Slide Time: 31:08)



Now, I do a multiscale feature extraction on the GPU. So I start with something called as a feature Vec, this is a buffer which I am creating over here and this is of the same size as the total number of valid pixels. And this is just a column vector, so this is something like all valid pixels cross one that is the column vector size I am using. Now what I do is, I generate a kernel which is for me using a Gaussian kernel of the size of 3 times the number the scales I am choosing and the scales and this is going to iterate over each scale. So I choose a scale index which moves from 1 till length of the scales. So I basically have 5 scales, so 1, 2, 3, 4, 5 and when I do a scales of scale Idx over here, I would be able to take that particular value whether it is a 3 or 5, or 7 and we had seen above. Now, what I do is I use this kernel and a filter 2 function which is we overloaded for GPU executions.

And I generate basically a feature map of the same size as that of the image. And since it is on the GPU memory I need to collect it on the CPU memory and my (()) (31:59) vocal command is gathered. Now if you do not have a GPU, do not need to worry a lot you can just remove this part over here and get rid of this one and as well you can remove this GPU array part over there, so it solves your problem your CPU is also well enough good to run on them, t might take a bit longer that bit longer than that. Now, after that I just linearize so what I do is I take these features from here on the CPU memory, transform them onto valid so this valid is just a binary mask. So wherever these features are present it will give me a linear array, which is the same length of valid cross 1.

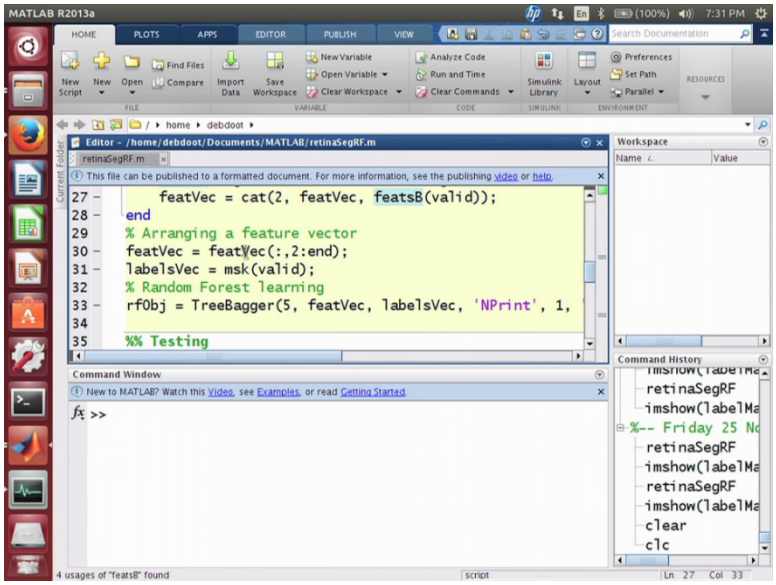
(Refer Slide Time: 32:37)



Now I am going to append that concatenate on the second dimension. So I am now basically collating each column one behind the other over here behind this buffer which is called as

featVec and I am going to do the same thing for the G channel, B channel and everything. So basically for each scale, I am going to get down 1 buffer and I keep on collating them so for the first scale which is a 3 + 3 scale, I have green, red, green and blue collated, then for the second scale which is at a 5 + 5 I again do the same thing and eventually I keep on doing subsequently for all of them.

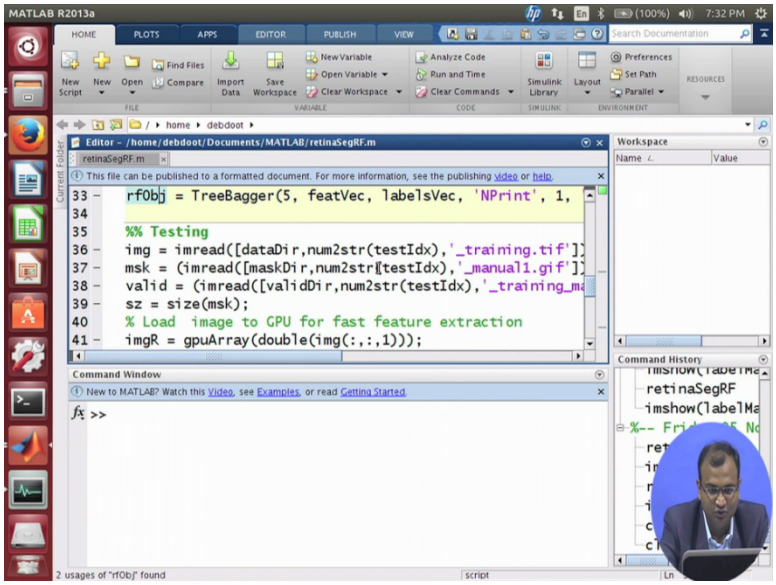
(Refer Slide Time: 33:08)



So this is just a here what I am doing is a basic book keeping which is get rid of the first column which is a garbage of all 0s using over there, so this will give me all the 15 feature vectors over there. And labelsVec is again just a vector linear vector of the size of the same as the valid number of pixels, which will give me whether it is a vec it is it is basically vessel or not a vessel. From there I enter into my TreeBagger, which is how I am going to create my Random Forest so this first argument is basically number of trees I gave the over there. This is the features which I am going to give, these are the labels over there and this NPrint is basically a,nd to print after each tree has been grown; I just want to see whether my forest is growing over there. And I have a minimum leaf exit criteria I set at 500, so if the number of samples at a leaf goes below 500, then you are just going to stop splitting.

So the same thing you can actually look into much more detail if you if you read through the documentation of TreeBagger itself which is very well documented for Matlab so I am not keeping it former. Next one we do is once we have this object created next is my testing phase.

(Refer Slide Time: 34:19)



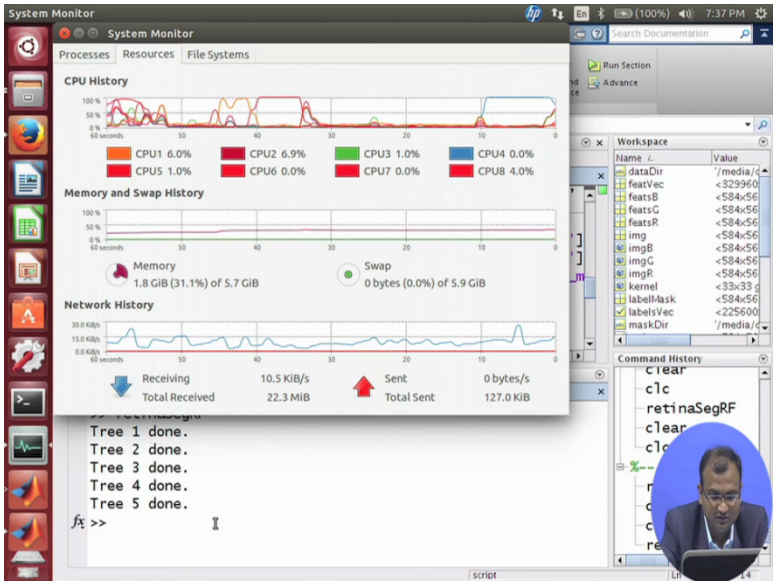
So in testing also I will be using all of them I will be using this mask as well in order to find out the accuracy, but I am not going to use it for the time being, now you will see eventually what were where it goes to use and then I create this buffers in order to get my image data transferred on to the GPU. And then I do the same amount of processing over here to get down my features then I get my features over here. Now look over here that I am just getting my features and linearizing them onto to the feature vector, I am not using anything from the labels for this particular one. And then what I do is I just put it through the predictive module and I call my rf object which I had learned in the training part over there and push the feature vector through it and what I would be getting is prediction.

So how it gives out is basically a cell, which consists of the class labels and the predictions per class. So I am I just do not want to look at the class labels, but I am more interesting looking at predictions per class. Now what I am going to do is this prediction per class this is like two column array, so basically you have two classes so it is going to give you posterior for each pixel for each class that is how you get two columns for each of these classes.

I am going to take the second class which is my class for the vessels because that was class one when I was training it down. And then I reshape that into the size of the matrix which I had taken up earlier using that mask. And then what I do is, I do a dot product with the valid region, so I just want to get rid of the rest of the thing which was not from the retina so all of them mall predictions and then I will be getting some sort of a probability in 2D mask. Now

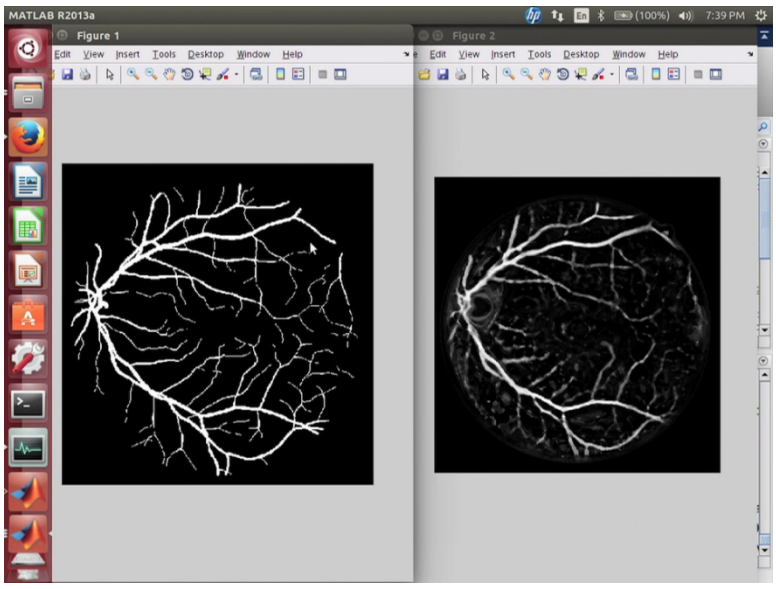
let us run this whole code and see it, so what was goes on. So now let us do a run over this one and see how it works out.

(Refer Slide Time: 36:18)



So if I just click on this run on my editor I see that the features have been extracted and I am actually training on my trees over there, I can see look at by performance monitor, ok good. So now my whole training as well as testing everything is done, this is actually done whole acceleration which you would be getting down getting down through a GPU. Now let us try to look into what the prediction was for the label mask, so I just do and I am sure of within a full range and this is what I am predicting it on.

(Refer Slide Time: 36:57)



And let us see how this actual mask was looking, so that is my variable which is called as MSK and so ok. If I want to just keep one on top of the other I would do a figure show it, so this is what I am predicting for my vessels and this is what the vessel actually is on the ground through. So if you look at them they are pretty close to each other not that bad I would say it is for the fact that we are just keeping in mind that we have just train used only one sample. And if we can use the more the number of data, the better it is always and the more the number of trees you can always play around with those numbers and so you can just keep on trying and there is a very simple example.

Eventually when we covered down much more details we would be going through it. The code would be made available for your use as well. So with that you can keep on playing around with Decision Trees and Random Forest and thanks.