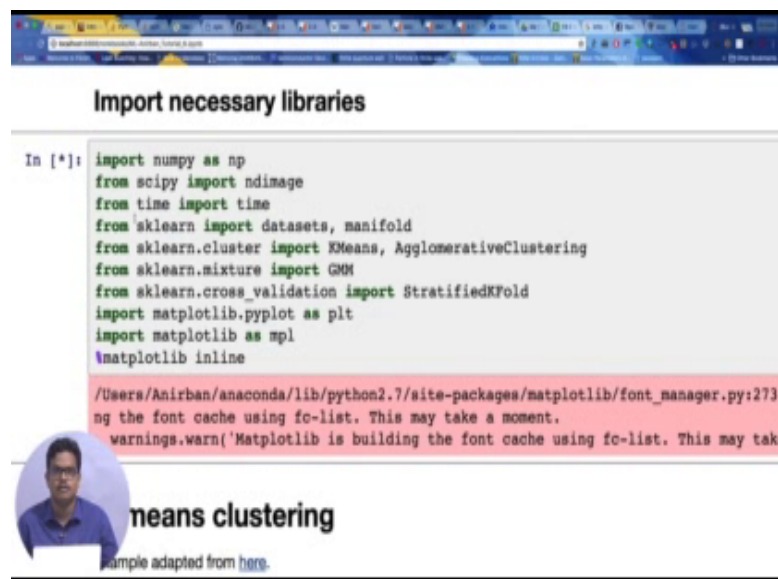**Introduction to Machine Learning**
Prof. Mr. Anirban Santara
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 40
Python Exercise on kmeans clustering

Hello friends, Anirban here, welcome to the python programming session of the last week of this course. Today's topics are different kinds of clustering algorithms. Now we are going to take up kmeans clustering algorithm, then look at Gaussian mixture models, and finally, we are going to look at different hierarchical clustering algorithms. So, I have all the code ready and I am going to implement them cell by a cell after explaining what they do, and then we are going to see the results. So, without further I do, let us jump write in the coding.

(Refer Slide Time: 01:41)

So, first we are going to import the necessary libraries and as you can see that this today we are going to use the kmeans algorithms. So, these are already implemented under scikit learn dot cluster and agglomerative clustering, and these two are from scikit learn dot cluster agglomerative clustering is actually the kind of hierarchical clustering that we are going to take up.

The Gaussian mixtures models will be taken from scikit learn dot mixture and so we do not have really written the main code of this algorithm. So, as these are already available

in scikit learn. So, after doing the imports, so, I first execute this particular segment, and the imports happen, the imports are here.

(Refer Slide Time: 01:46)
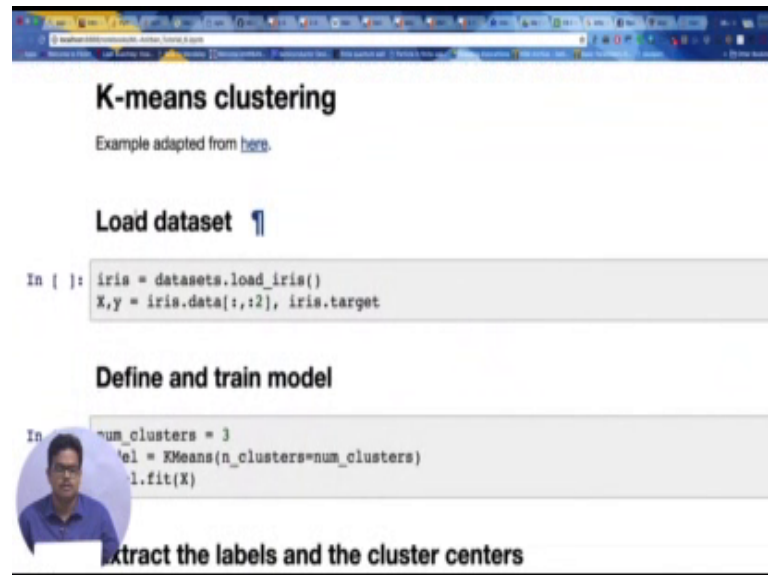
So, after that let us first look into k means clustering. And I have actually like this is not an original set of examples, these are right from the official documentation of scikit learned and I have given the link to that here. So, that is actually a bit more sophisticated version of the same next set of experiments. So, let us look at the experiment that we are going to do today.

First, we are going to load the dataset. So, the first section of the code, it invokes scikit learn dot load iris and it loads the iris dataset, the same iris dataset that we have used in the previous exercises. And the iris dataset just as a recap, the iris dataset is dataset that is meant for classification algorithms and this dataset consists of iris flowers described in terms of their sepal, length sepal, width petal length, and petal width. So, there are four features describing each instance of iris flowers, and it will be depending upon these features values, we have to predict which particular category that flower falls in. So, it could be in either of the three different species of iris flowers, and that is what we have to do.

So, the dataset dot load iris will directly load the iris dataset into this variable, and then we pick up the data. So, iris dot data contains the input data - input features, and iris dot target contains the target species value - the 0, 1 or 2 three different species which we are

trying to predict. And you can see here that we are picking up selecting the first two features of the dataset because and that is what we have been doing in most of our exercise till in this course. So, this actually keeps it gives the problem simple and it is easy to visualize what is happening. So, after we have loaded the dataset, let us go ahead and load dataset.

(Refer Slide Time: 03:56)
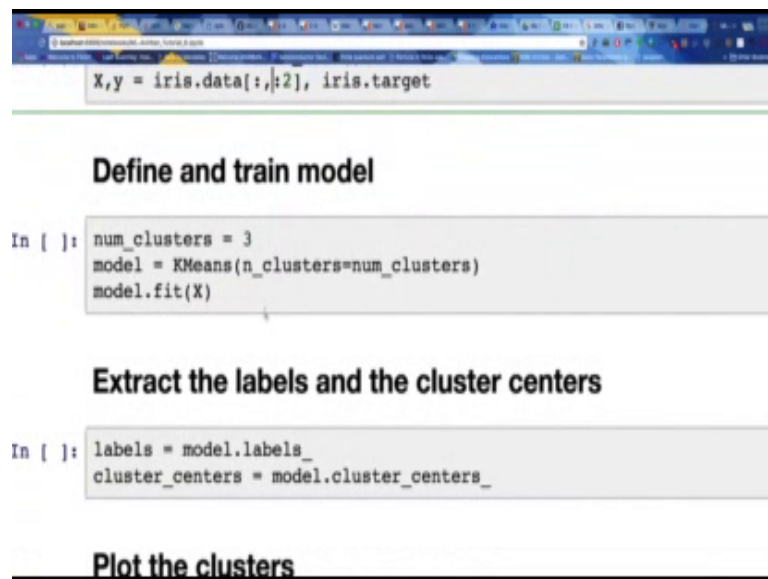
So, after we have loaded the dataset, we define and train our model. So, in the clustering algorithm, we have to first specify how many clusters we would want in the clustering algorithm. So, here is how the clustering of the kmeans clustering algorithm works. So, first, we are given kmean clustering algorithm is an unsupervised algorithm, unsupervised machine learning algorithm. So, we do not have class labels here, we just want to work with the input feature dimensions. So, once we have been given our dataset in kmeans clustering, we are going to make like first we have to make guesses about k centroids. So, the data will be divided into clusters, and each cluster will have a centroid. So, those centroids are first guessed.
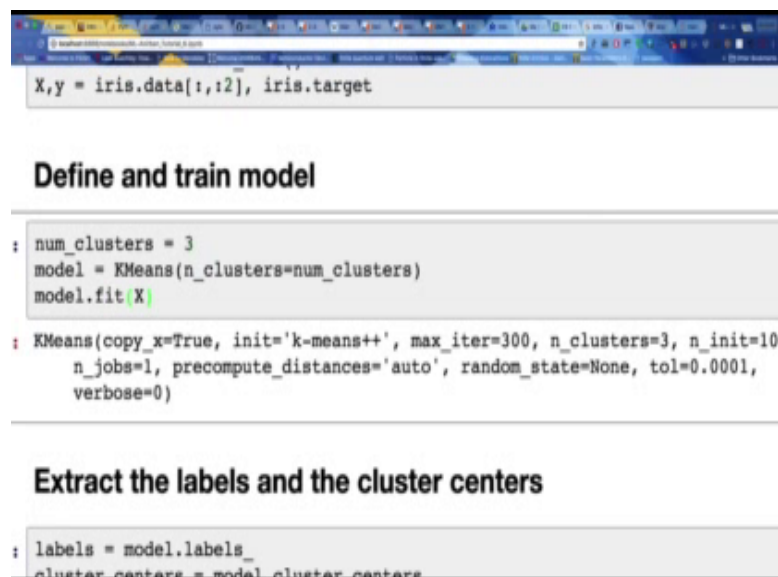
So, we can choose randomly like you have those training samples as the initial centroids; and then for every single centroid, for every single once the centroids are here, have with us, what we do is for every single of those points in the dataset, we are going to find which centroid is the closest to that particular point.

And each centroid actually corresponds to one cluster, so every single data point is

assigned the cluster which is corresponding to the centroid that is closest to the data point. So, after this step, the each point of the dataset has a cluster assigned to it, which has a corresponding centroid. So, again in the next iteration we are going to update the centroids. So, each centroid is now going to be shifted to the actual centroid of the cluster points that were assigned to it. So, then again we are going to like revise the cluster assignments of every single point in the training set. So, this is how the kmeans algorithm works. So, you have already been taught in great detail in the theory session and what I just said is a quick recap of the algorithm.

So, this is what we are doing here. The number of clusters is three here. This means the value of k of the k means algorithm is 3. Now we define the model. So, this beautiful api of scikit learn allows us to first as you know to clear the model with all the parameters in just one step. So, this k means object is they take as the input numbers of clusters in the attribute in clusters. So, this particular keyword is assigned the number of clusters 3. So, k means with n clusters is equal to 3 will do you know k means algorithm with on 3 different clusters. So, once this particular model has been cleared, we fit it on the data.
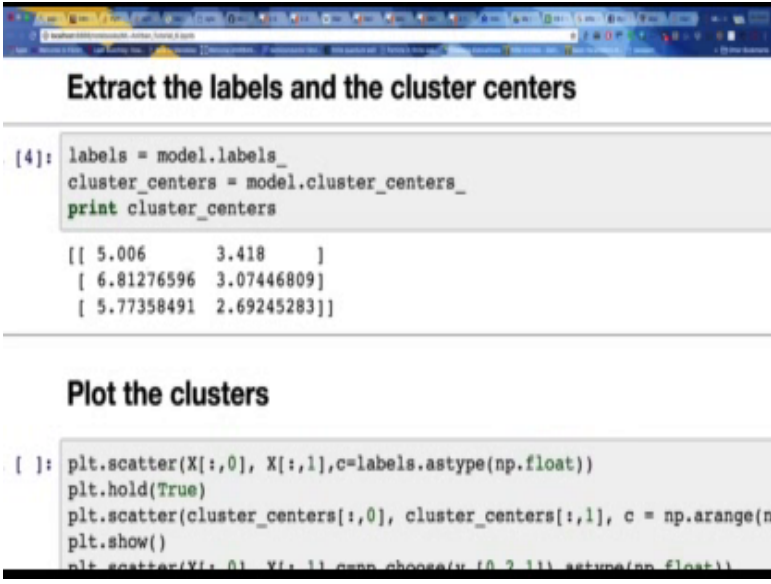
(Refer Slide Time: 07:08)

So, this actually does the training. And you can see what kind of object has been created here, so the number of, maximum number of iterations that were used for achieving the convergence. Before deciding the final clusters is given here, so it was like 300 different iterations, 300 iterations of the k means clustering algorithms and the number of

clustering was 3. And n init is a variable which specifies the number of you know the number of times you want to repeat this entire experiment.

So, what clusters you finally end up with actually depends upon the total number the kind of initialization you have on the data points. So, you know what initial points actually define the final cluster and that is why you need to restart the experiment again and again with newer and newer with new initializations with different kinds of initialization. So, say for each experiment of the k means algorithm you are doing 300 iterations; so n init equal to 10 means that we are doing the same experiment 10 times with different initializations.

And then we will choose the results of that particular experiment which had the best results. And this best results will be judged according to a particular metric which is internally specified as you know the squared distances, so the compactness of the clusters. So, how compact the clusters are, so that with that experiment which gives you the most compact clusters is going to be taken as the considered as the best one and the results corresponding to that that will be used.

(Refer Slide Time: 09:21)

Extract the labels and the cluster centers

```
[4]: labels = model.labels_
     cluster_centers = model.cluster_centers_
     print cluster_centers

     [[ 5.006        3.418      ]
      [ 6.81276596   3.07446809]
      [ 5.77358491   2.69245283]]
```

Plot the clusters

```
[ ]: plt.scatter(X[:,0], X[:,1],c=labels.astype(np.float))
     plt.hold(True)
     plt.scatter(cluster_centers[:,0], cluster_centers[:,1], c = np.arange(n
     plt.show()
     plt.scatter(X[:,0], X[:,1],cmap.choose(y,[0,2,1]).astype(np.float))
```

The next the other you know other variables can be like the description of these variables are available in the official scikit learn documentation of kmeans. So, after we have trained this algorithm, let us see what all you know attributes this particular object has. So, this particular model has this particular attribute called labels underscore.

So, this labels and labels underscore actually gives you the cluster index corresponding to every single training point. So, at the end of the clustering, every point in the training set will be assigned a particular cluster and labels underscore will contain those clusters, cluster you know those cluster numbers. So, I have the labels inside this and also you can find the cluster centers, say you wanted just like 3, you had 3 clusters over here, and you can find the cluster centers or centers centroids. So, let us print out the cluster centers, cluster centers yeah. Let us go ahead and execute this.

(Refer Slide Time: 10:17)
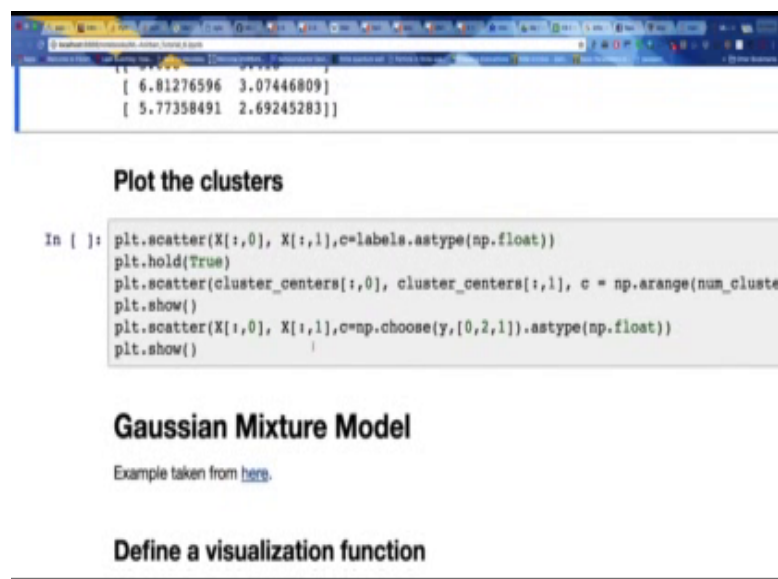
So, you see you have the label inside this and these are the cluster centers, these three, one along each row. Let us plot the clusters and this will be interesting. So, this is the simple like this map dot loop dot pipe dot plt, so this plt dot scatter all right. So, first we have doing scatter plot of all the points in the training set with the labels as were predicted by the kmeans algorithm, and then we are marking the cluster centers there, and then we are doing the same like we are plotting out what the ground truth labels looks like.

(Refer Slide Time: 10:52)


```
plt.scatter(X[:,0], X[:,1],c=np.choose(y,[0,2,1]).astype(np.float))
plt.show()
```
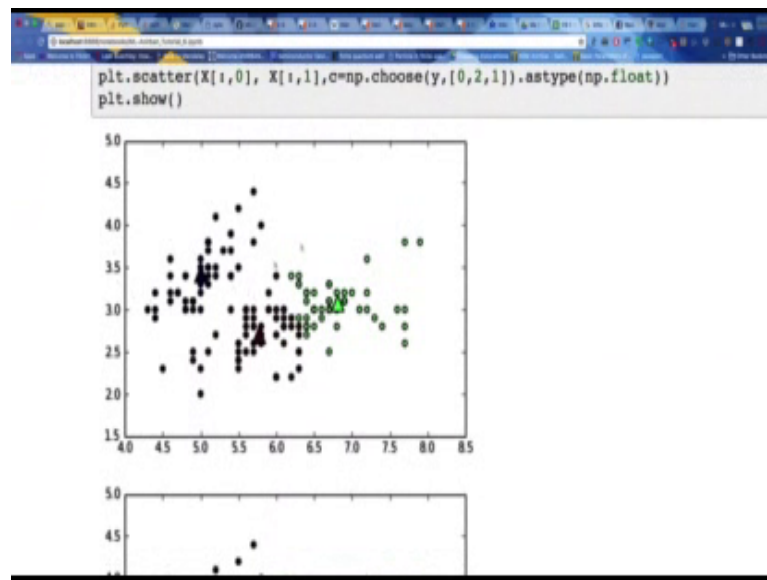
So, let us go ahead and see how this thing works. So, you see let us look at the first diagram. So, in this case, you can see that in this diagram you can see that the three are the data points were actually divided into three clusters, and these big triangles these are the clusters centroids that were obtained.

(Refer Slide Time: 11:13)
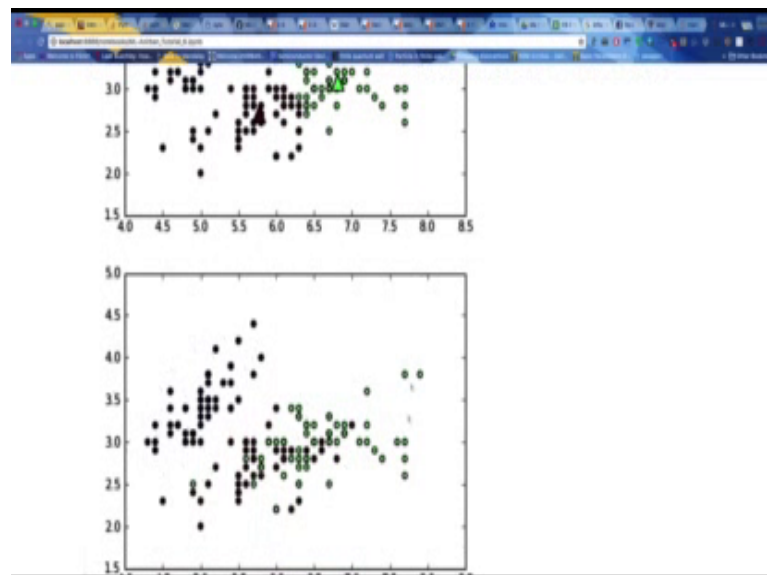
These are the original labels and you can see that the k means algorithm could successfully cluster out all the class zero points into one cluster as would typically to keep the case is; however, since class two and class like class one and class two they

were all mixed up right.

So, it could not you know properly detect both the classes due to the overlap and hence there is some inaccuracy over here. So, you can see that how the clustering performs right and this comparison actually reflects that the clustering is doing something meaningful. So, the objects or the samples of the same class are been clustered into one cluster you can see. So, all of these samples they belong to one particular class according to the ground truth and they were all put into one single cluster by the k means clustering algorithm. And these clusters are also like kind of homogenous, most of the green samples reside in this part and they have been marked properly right.

(Refer Slide Time: 13:01)
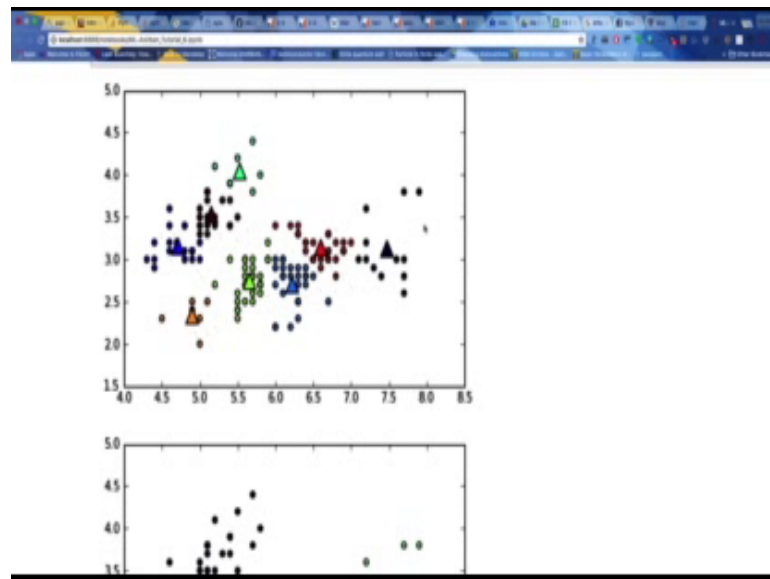
Let us go ahead to Gaussian mixture model let us go ahead and first likes you see what how things change when we change the number of clusters let us make it 8, and we retrain let us execute this part. So, now you can see that there are 8 clusters enter and let us plot the clusters again. See, in the same ground truth it is here, but now in the clusters are there were large number of clusters.

And you can see these points have been clustered together like these together these together, all of these are different clusters and the corresponding clusters are marked. So, this is how k means algorithm works; k means algorithm is one of the most you know widely used clustering algorithms in literature. And it happens to perform really good in most scenarios a pretty algorithm to works on to just begin with if you are clustering.
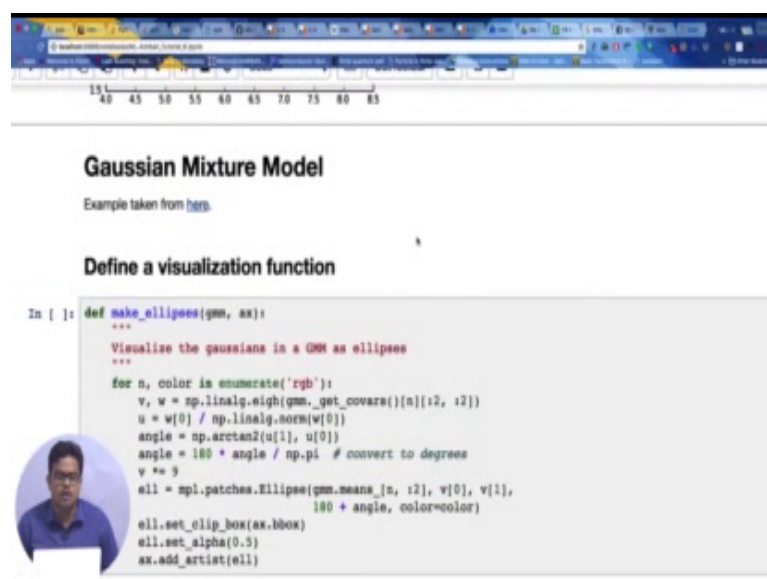
Let us look at Gaussian mixture models next and this is completely like copy pasted from another like standard tutorial from the official scikit learn website and with little modifications and comments here and there. So, you can actually look up this link and find the actual one. And I choose this because this gives a really wonderful visualization of how what things how things actually work while you working using with will work with gmms. So, first what we our motivation here is to see how Gaussian mixture models work.

So, let me give you a quick you know explanation of what Gaussian models, how Gaussian mixture models work. So, the idea is to the motivation is to like approximate a particular probability distribution. So, you have some probability distribution which goes this way and you want to fit curve to the distribution all right and say the distribution like looks a hilly terrain and that can be approximated as a you know a linear combination of several Gaussian distributions. It is better explained on paper, but I do not have on now. So, I will be doing in the tutorial session.

Gaussian mixture model is all about fitting a particular distribution using a mixture of Gaussians - linear combination of several Gaussian distributions. And the algorithm is trained by or the model is trained that is the parameters of the model are learned from data using a particular algorithm which is called the expectation maximization algorithm, and which is an efficient and an efficient way of doing approximate maximum likelihood

estimation. So, all of these theoretical details have already might already be covered in class or else you can look up the web and find out why these things are so.

So what we will be doing here is we will be like we have a dataset, the dataset is that of hand written characters, handwritten digits. And from the handwritten digits dataset, you have to detect in that handwritten digit dataset, you have to like run a clustering algorithm, and try to separate out the different kinds of digits. And we will see when it how Gaussian mixture model performs in that particular scenario.

So, what do it will do, it will like try to fit Gaussian's over different areas of the input space and then try to model the entire probability distribution as a linear combination of those Gaussian distributions. So, finally, we will have a nice visualization of how things work I am sorry we would not be working on digits in this particular section, we will be working with digits in the next section, here we will be working right now only on the iris data only.

(Refer Slide Time: 16:36)

Gaussian Mixture Model

Example taken from here.

Define a visualization function
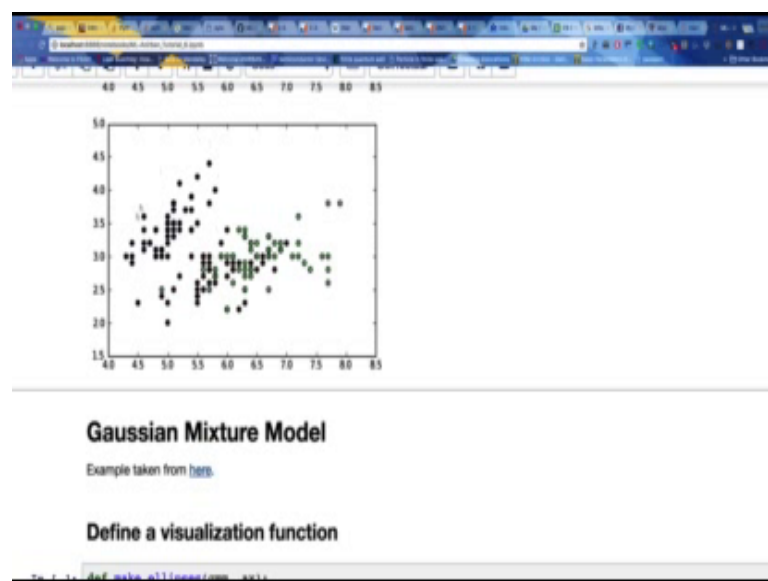
So, the same iris data we had in the previous case. And we will be fitting Gaussian's on each of these classes all right and checking how the system works. We will be working with digits in the hierarchical clustering because that makes things much more interesting all right let us go ahead.

(Refer Slide Time: 16:53)
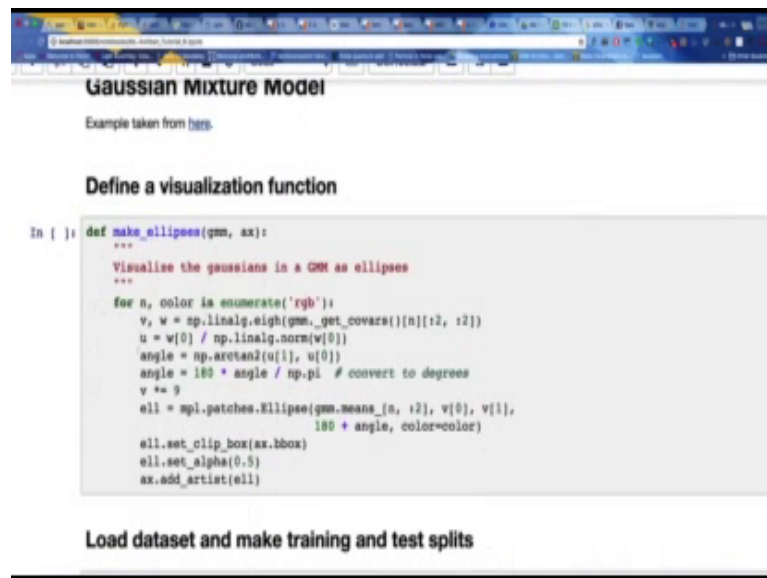
So, this is the function which makes this make ellipses function this gives a nice visualization of Gaussians. And let us first compile this function. You can go ahead and see how this function works, what this is not really related to this to the main objective of this lecture.

(Refer Slide Time: 17:13)
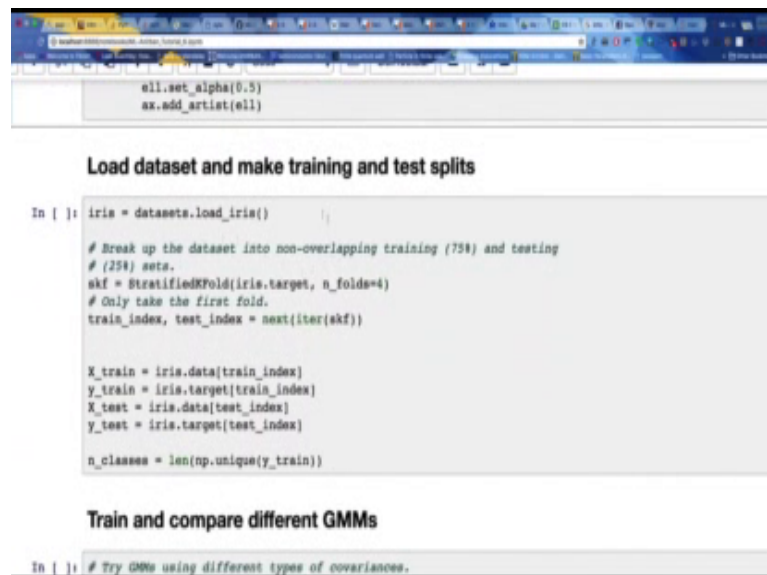
Let us go ahead and load the dataset and make training and test bits pretty straightforward.

(Refer Slide Time: 17:21)

Let us go ahead to the next section. In this, we are going to compare different kinds of Gaussian mixture models. And let us first execute the code and then I will talk about it, it takes some time to run.

(Refer Slide Time: 17:43)
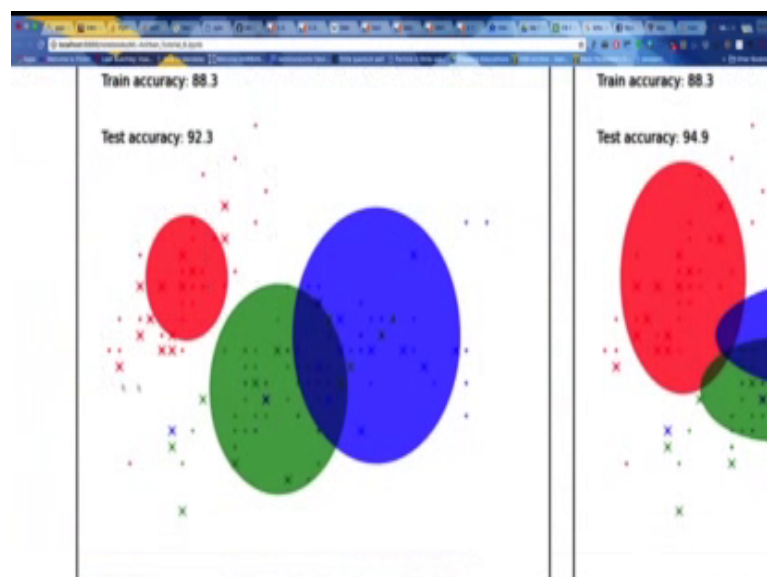
So, it is here right here. So, what you see here is the performance of different kinds of Gaussians. So, what do you see here, let us concentrate on one. So, these are the same iris dataset. And these are the 3 Gaussians that are being fit over here. And this circle are actually like kind of contours and gives the shape of the two-dimensional contour of the

Gaussian and the bigger process they are the test data. So, and the smaller ones are the training data. So, what we are doing here is we are trying to compare the effect of using different kinds of Gaussian's different kinds of like covariance matrices of those Gaussians. Spherical covariance metrics will give raise to these kinds of clusters.

(Refer Slide Time: 18:37)
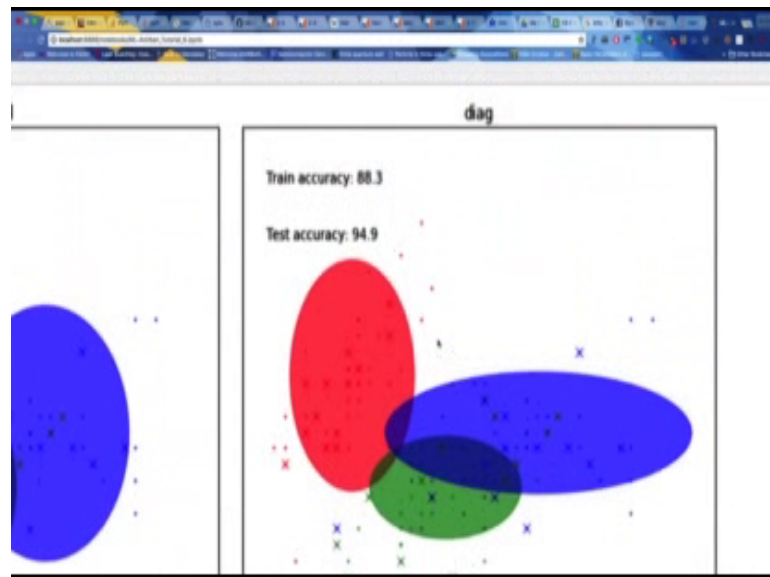
And this one is for a diagonal covariance matrix.

(Refer Slide Time: 18:41)
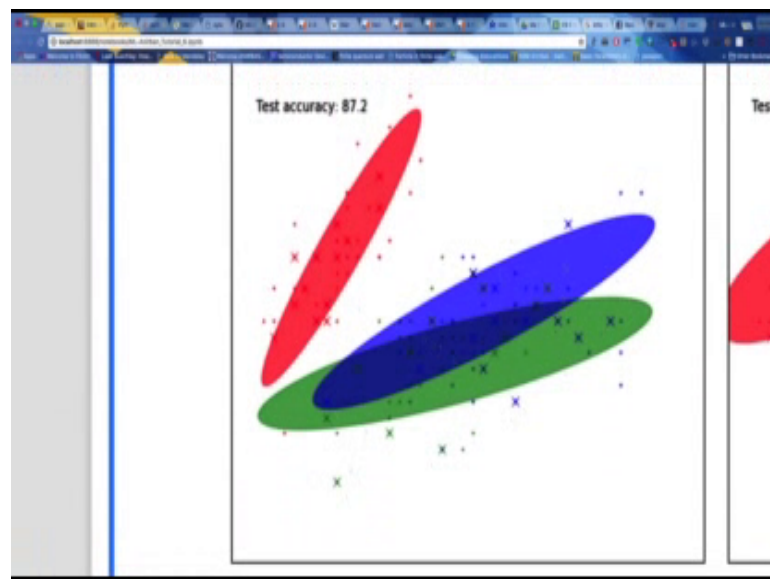
And if you have a full covariance matrix, it will be this way.
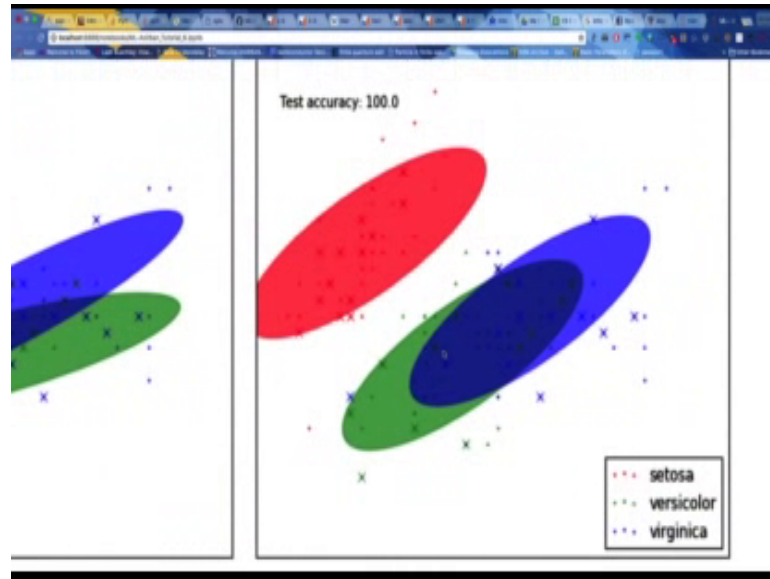
And if you have a tied covariance matrix then you will have these kinds of clusters. Now, let me tell you what these mean actually. So, a spherical covariance matrix means that the covariance matrix will be diagonal the covariance matrix of each Gaussian I mean. So, the 3 different mixtures will have three different Gaussian distributions and the covariance matrices of these Gaussian distributions will be different.

However, in each covariance matrix will be diagonal covariance matrix also the diagonal elements should be equal. So, this mean that the Gaussians will be have a kind of like this kind of circular or spherical cross section or which is also called contour. So, it assumes that the data has the same variance along both, along all the feature access.

So, this is an assumption and this is the result that you see. So, see that the Gaussians are almost circles. So, they are not they are actually their circles, but due to different scaling of these axis it appears that they are not circles, but they are actually circles, because the variance along both the future axis are equal in this case.
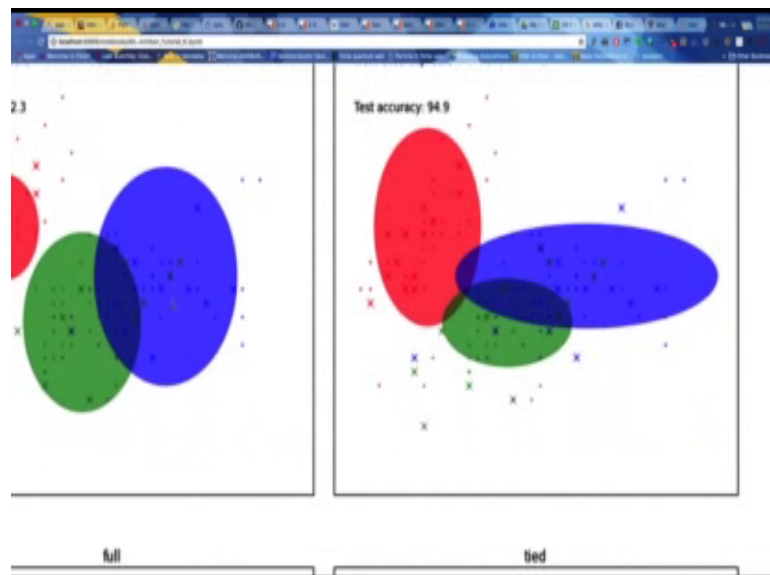
So, diagonal covariance matrices mean that the covariance matrix should be diagonal that; that means, that the Gaussian will be aligned along any one of the future axis all. So, you can see that the one this Gaussian is aligned along the y-axis. So, this one please

turn the camera towards my computer. So, the first Gaussian is along the vertical direction, and the others are horizontal. And the same is the case with, so what I was talking about is this one all right, and sorry for the technical problem.

So, the first Gaussian you can see that this is this is aligned along the vertical feature direction, the others are along the horizontal direction. The diagonal covariance matrix means that the Gaussian will be directed along any one of the feature axis, it would not be skewed. And the same is the case here, because the diagonal the Gaussian is aligned along one of the axis that is true because all of these are aligned along the vertical direction and also there is an another extra constraint that whether variances should be equal along both the feature axis. So, this is a much stricter you know constraint on the shapes of the Gaussians than this one; now what you see for the full, yeah this one.

(Refer Slide Time: 21:53)
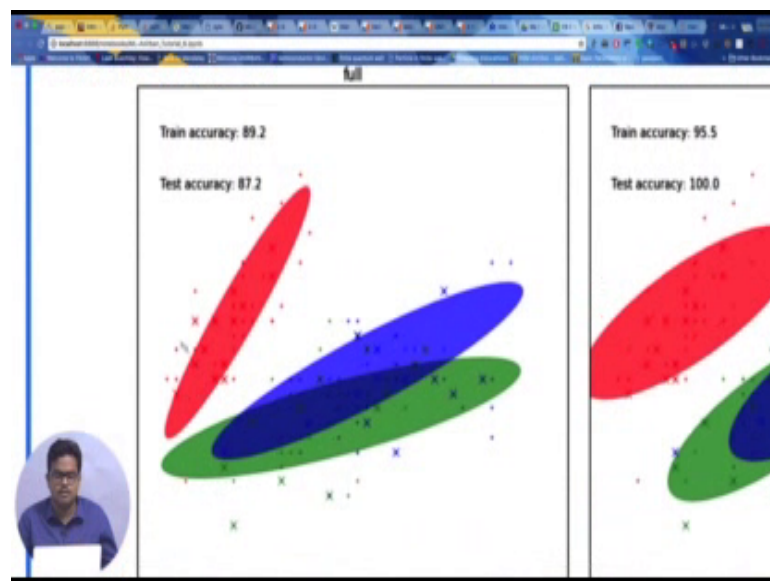
So, this means that full means that the diagonal that the covariance matrix is the full matrix, it does not need to be diagonal, and all the elements are available. So, you can have you know skews. So, you can see there that the Gaussians they are no longer aligned along one of these axis; as the off diagonal elements are nonzero allowed to be nonzero. So, the Gaussian can actually steer itself along the axis and can have a skew. So, this gives a much more flexibility in the modeling and this is reflected in the performances.

So, note that the training accuracy is 89 percent and 87 percent here. See the training

accuracy. So, the training accuracy means that how good the model can fit to the data. So, this is what we are doing, we are fitting the model to the data. So, we can talk about generalization later, but you see that the training accuracy is 89.2 percent here whereas it was 88.3 and 88.3 in the previous two cases. So, the training accuracy improves, just because the full covariance matrix Gaussian can fit on the data much better.

(Refer Slide Time: 23:11)
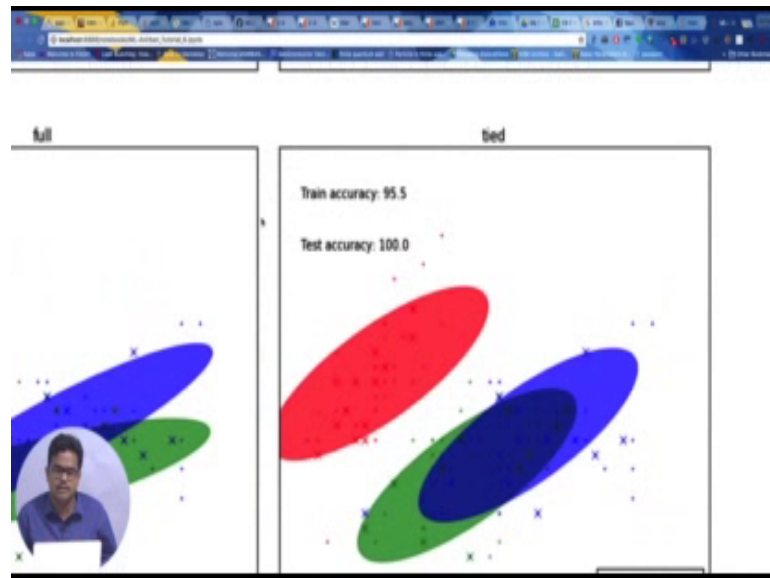
And we will talk about generalization later, but let me talk about the let me explain tied covariance matrix means. Tied covariance matrix means that covariance matrix of the Gaussian's can have off diagonal elements. So, the diagonal matrices are full; however, all the Gaussian's should have the same covariance matrix. So, it is imposed. So, you can see that all the ellipses are looking the same right, only the names are different.
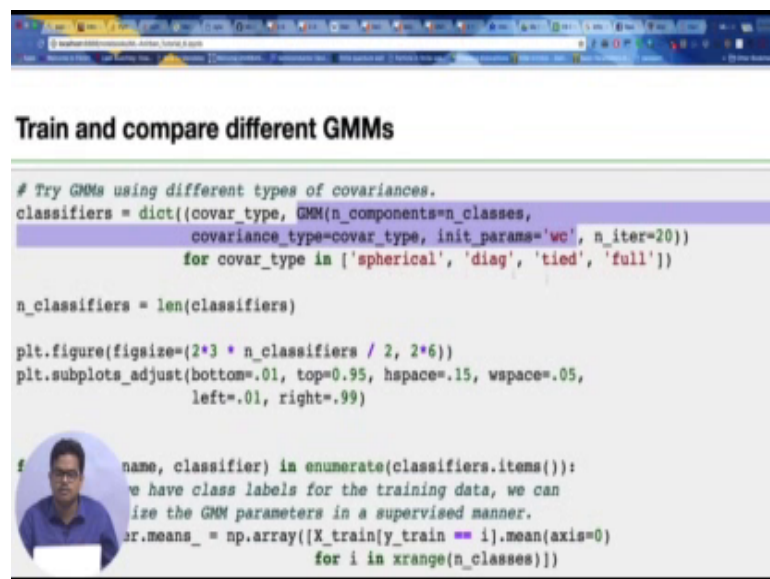
So, you are allowed to have a very you know high dimensional or you know full covariance matrix. So, you are allowed have a lot of parameters in a single a lot of free parameters within a within a single covariance matrix, but all the covariance matrix will be tied; that means, that all the Gaussian in the mixture will have the same covariance matrix which will be full. So, you can see that this particular constraint actually improves the training accuracy to 95.5 percent. So, you can actually fit on the data better and the test accuracy is also more hundred percent.

So, you can see that the test accuracy is better for the spherical covariance matrix and diagonal covariance matrix than the full covariance matrix this is because of you can

clearly see the over fitting over here. So, as the full covariance matrix has a larger number of parameters. So, it can fit on the training data more snuggly all right and as it is like it is hence it can actually like over fit on the training data and hence the generalization performance is poor and this is what is reflected in the test accuracy over here.

However, the for spherical and diagonal covariance matrices the matrices that the system modeled as have not have you know many parameters, and hence it cannot over fit on the data. And as the over fitting is less in this case in the generalization performance which is reflected in the test accuracy is better. And you can see that the test accuracy is best for the tied case in which it is imposed that all the different Gaussians in the mixture will have the exact same covariance matrix.

(Refer Slide Time: 25:43)
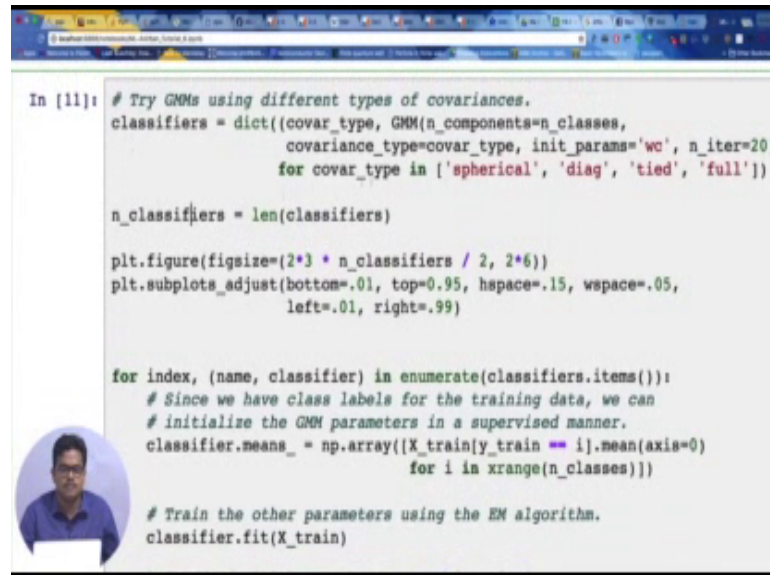
So, let me look at me let me show you the code. Now what we did is we will pretty simple. So, we first we find a dictionary, and the dictionary will be like some covariance type. So, whether specifying that we are going to deal with 4 different covariance matrix types, so spherical, diagonal tied and full, so this is something called a dictionary comprehension and the dictionary is going to have covariance type and GMM components. So, this is the particular dictionary and this is the particular GMM model that we are going to train in each case, where n classes will be the n components number of classes that we have.

Covariance type will come from either one of these elements in the list. And the number of iterations in the approximation in the estimation algorithm which is e m algorithm is going to be 20.

(Refer Slide Time: 26:31)

So, after this, what we do is I will say that n classifier is equal to length of classifiers. So, in this case, it is 4. And then we initialize the plots and sub plots and then for every single classifier we first initialize the means to the means that we are getting right from the training data. So, as the labels are available in this case. So, we know that which particular class as which particular mean all right. So, we extract the mean here and then we assign the means and then we train the rest of the parameters using classifier dot fit and then we plot the we initialize the sub plot and make ellipses for using a classifier, so all of these ellipses that that you are seeing here, they are generated at this step.

(Refer Slide Time: 27:23)

And then after these ellipses are have been made you plot in the data and you show the labels as well and then you do some more can see you know then you just like find out the training accuracy, and test accuracy and print them right on the figure this is what is done.

(Refer Slide Time: 27:51)
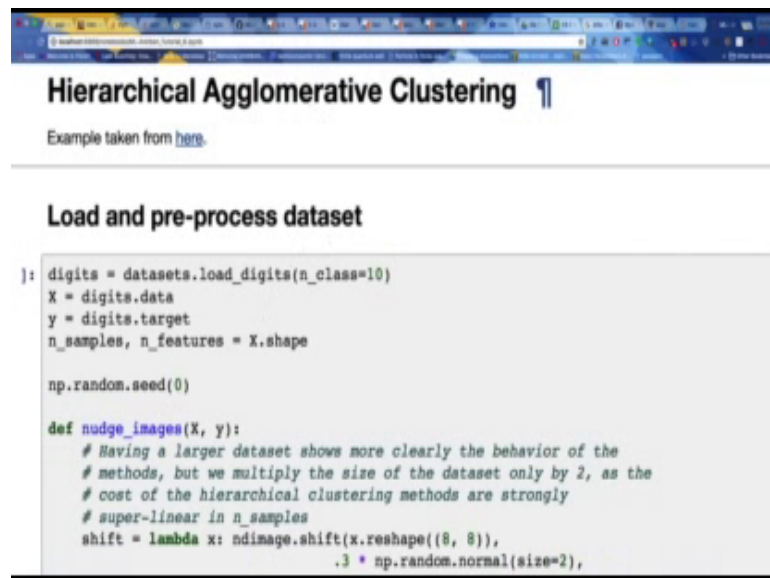
So, what did we see here we just studied how the GMMs, how Gaussian mixture models can be you know implemented in scikit learn, and using the inbuilt GMM model under mixture. And then saw the effect of different kinds of Gaussians different kinds of

covariance matrices on the performance of the estimation algorithm.

And we saw that as the number of parameters is increased by increasing the degree of freedom in choosing the covariance matrix, the performance on the training set increases which means that it is starts over fit on the data on the training data. And performance in the test set you know deteriorates. Also we saw that we studied what different kinds of covariance matrix mean.

(Refer Slide Time: 28:54)
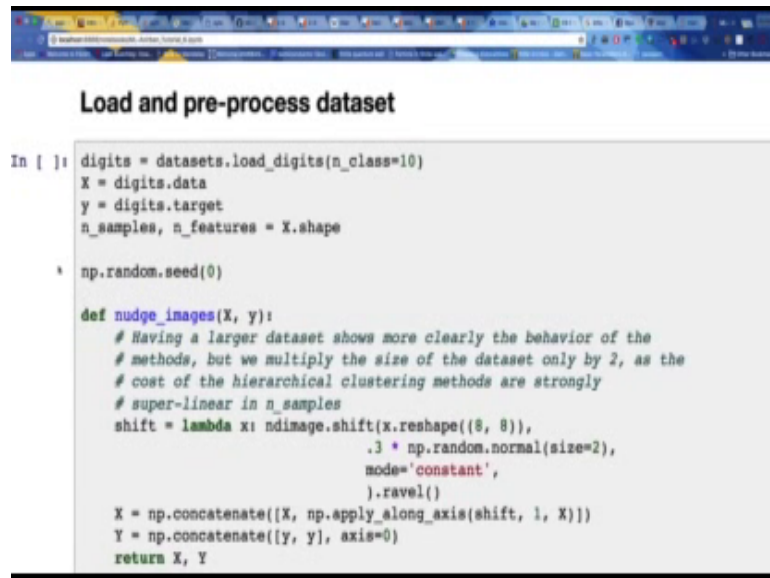
So, let us go ahead and study hierarchical agglomerative clustering. So, hierarchical clustering methods actually try to do the clustering in a hierarchical fashion, either from again a top down approach or in a bottom approach and agglomerative clustering is a bottom up approach in which initial clusters are every single point. So, initialize the clusters as the points in the data, and then you try to like you know merge those smaller clusters into larger clusters by bringing together similar looking points in one cluster, where did you seen the some kind of statistical distance between the cluster points.

(Refer Slide Time: 29:38)

So, we are going to study different kinds of agglomerative clustering. Let us in this particular experiment as I was talking about just wrongly mentioned in the previous earlier in the video that we are going to use the digit data set here. So, we load the digit digits and then you know you find out the take the input and the targets, and then you do something and then you write a function.

(Refer Slide Time: 30:04)
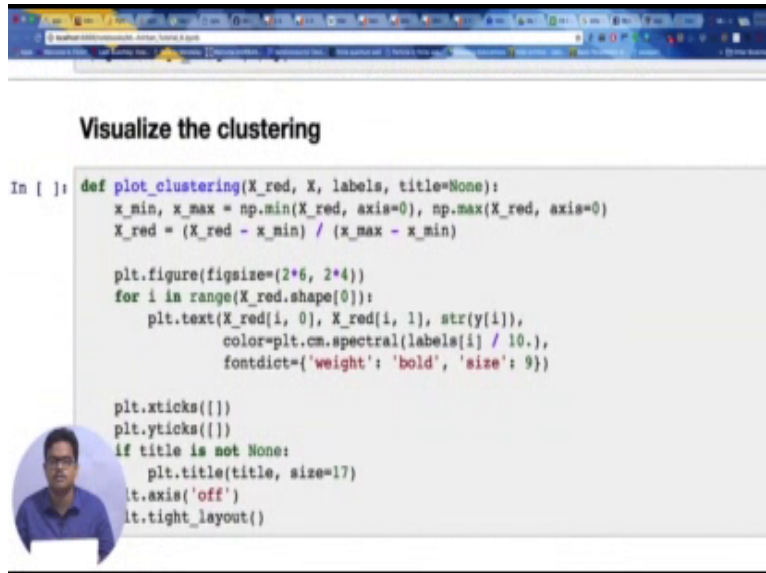
So, this is a function which extends the database a little bit in case of data set by doing some you know some shifts random distortions in the images. And this is likes it is helps

in the clustering algorithm because it get to see more samples. So, this is in first run this one.

(Refer Slide Time: 30:30)
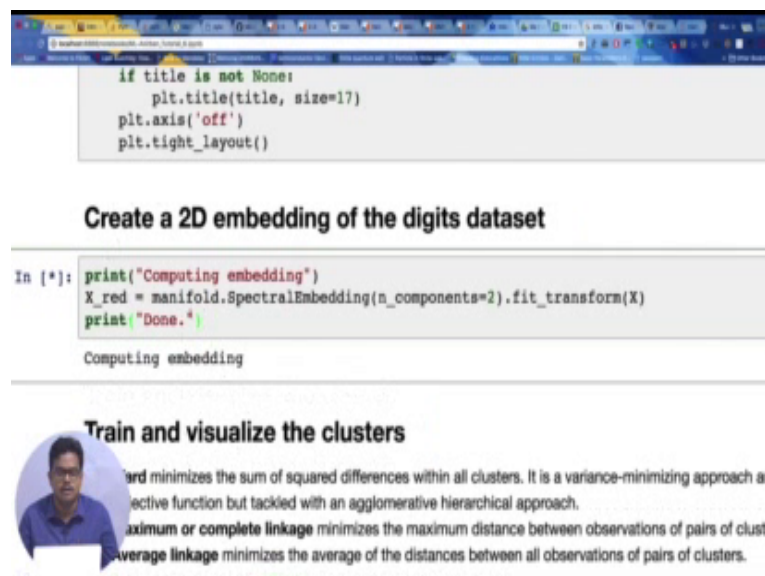
So, this actually loads data and do such some initial modification on to the dataset to make it mid algorithm more robust in estimating the parameters. And then we let us go ahead and visualize the clustering. So, this is actually a function which does the visualization for the clustering.

(Refer Slide Time: 31:46)

Next what we do is we do a two-dimensional embedding of the dataset. So, the dataset is

actually one of 8 cross 8 images. So, the images of handwritten digits are 8 cross 8, so they are 64-dimensional images. And what we are going to do is we are going to find out we are going to reduce the dimensionality to 2, so that we can express that you can you can visualize that on the screen. So, we use scikit learn dot manifold dot spectral embedding for this, and we say that the number of components is equal to 2, and then we fit the model and transform the data and we get the reduced dimensional data. So let us do the dimensionality reduction. So, it is like it first it computes the embedding and then it transforms.

(Refer Slide Time: 32:12)

And next, we are going to train and visualize the clusters on this reduced dimensional data. And we are going to talk about three different kinds of you know linkages that we can specify. So, this is done right we have the data we are going to talk about three different kinds of linkages in the algorithm. So, for example, this actually corresponds to the way in which we are going to compute you know how closed two clusters are in the statistical space in the feature space, so the idea of agglomerative clustering is to hierarchically combine the similar looking clusters together and like go up to till the end.
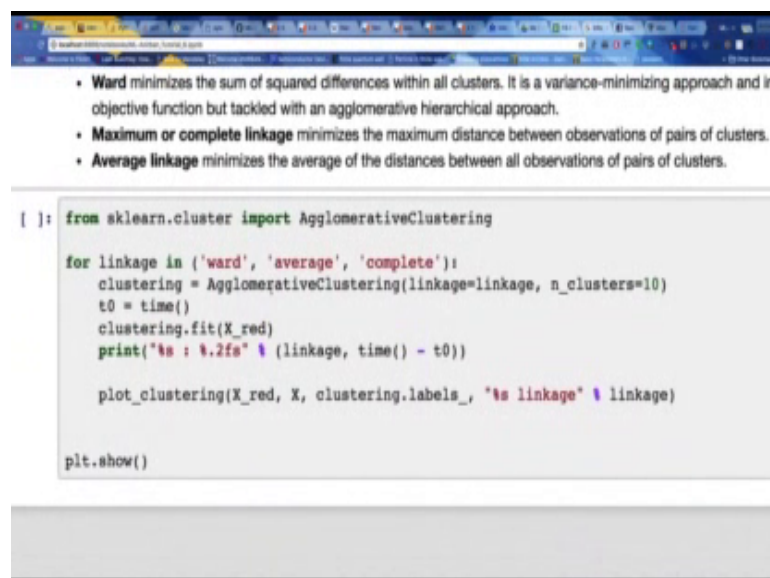
So, how do you find out how related or how close how similar two clusters are, so this is specified by these algorithms. So, the words algorithm actually tries to reduce the variance in the clusters all right. So, it is going to take kind of like you know squared exactly. So, it is like squared differences between the sum of square differences in all

clusters. So, it tries to minimize that. So, you can actually look up the paper and study more about this method. So, this was not taught in the course, but the other two were.

So, the maximum or complete linkage this algorithm this specific, this says that the similarity of two clusters is it should be judged on the basis of the maximum of the pair wise distances between the points of those two clusters. So, you have cluster a and cluster b. So, you are going to compute pairwise distances of every point in cluster from every other every point in cluster b. So, you have the set of all of these distances say there are n a points. So, there are say there are n cluster a and n 2 points in cluster b. So, there are there will be n 1 into n 2 n 1 times n 2 distances right and then you are going to say that these two clusters are.

So, you are going to the distances between these two clusters is going to be take chosen as the maximum of these distances. And you are going to judge how similar these two clusters are on the basis of the maximum of these distances. So, this is a complete linkage algorithm average linkage, what it does what it also takes the average of the pair wise distances. So, you compute the pair wise distances of the points of two clusters and then you take the average value of the distance all right the distance between two clusters this is the average of the pair wise distances of the points in those two clusters.

(Refer Slide Time: 34:56)

Examine these three kinds of in this three variants here. So, let me check yeah done. So, we import agglomerative clustering from scikit learn dot cluster. And for linkage in word

average and complete, we are going to do the clustering, clustering equal to so, this particular a line it initializes the clustering algorithm. The initializes the model and we are started to note time, and then we fit on the dataset the dataset is the reduced dimensional image. And there we print how much time was taken right then we plot the clusters.

(Refer Slide Time: 36:27)

Let us go ahead and execute this segment; it takes some time. Let me shrink this screen little bit, let us see how it works. So, you will see the effect of choosing different kinds of criteria and these were actually like kind of we are not well proven that this particular criterion will lead to this kind of this says wait a minute, so let us wait and talk. So, do one there no proofs that this will exactly lead to this kind of clusters, but still you have some you can have some you know some guess about the nature of clusters that are being produced by these different choices. You can see that different algorithms are completing and the times are being noted, the third one is remaining finish soon.

(Refer Slide Time: 36:34)

So, in this fancy plotting function, it plots with respect to like it, it you know it writes it plots with those numbers.

(Refer Slide Time: 36:49)
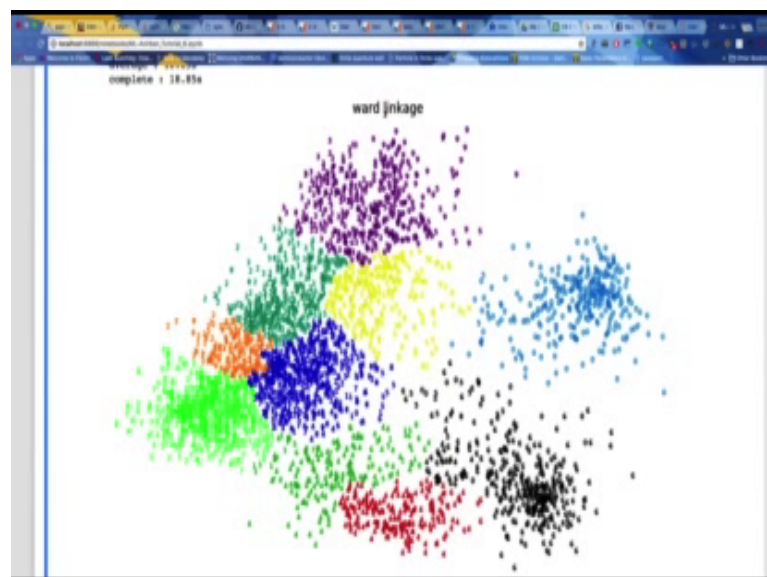
You can see you will see that yeah how they are been plotted. So, this is a very (Refer Time: 36:47) plotting style in which you instead of putting markers or likes like circles, squares in the plot you put the exact digit. So, you can use that function in a quite handy. So, this is the result of choosing words linkage. You can see that all the threes and nines they have been like kind of all the threes they stay together, all the nines stay together,

but they were all combined into a single cluster.

And these clusters what you try to notice is that these clusters are more or less homogeneous that this particular clusters consists mostly of sevens right this one is mostly of fours and fives, and this one you see. The similar looking characters they try to like being clustered together, this is mostly of zero it is all most entirely of zeroes. Let see how what is the case with the others. So, they are a bit different right you see that the words linkage we have classified these two groups of three's are separately; some threes came here some threes came there.

(Refer Slide Time: 38:01)

But complete link average linkage it to call the three's in one cluster, but there are certain cases which is called there under rich gets reach you see. So, the bigger cluster they continue to get more and more points. So, the bigger clusters in the case in the link course of that hierarchical agglomeration the bigger clusters will get more and more points and tend to get bigger.

So, the rich get richer whereas the poor remains the poor. So, you see that there is one cluster, this red one with one single point here, over here. So, this is one of the limitations of the average linkage algorithm, and in fact there is another point you see over here. So, this 4 is lonely, this is one single cluster, and this is just because of the nature of this particular algorithm.

(Refer Slide Time: 38:49)
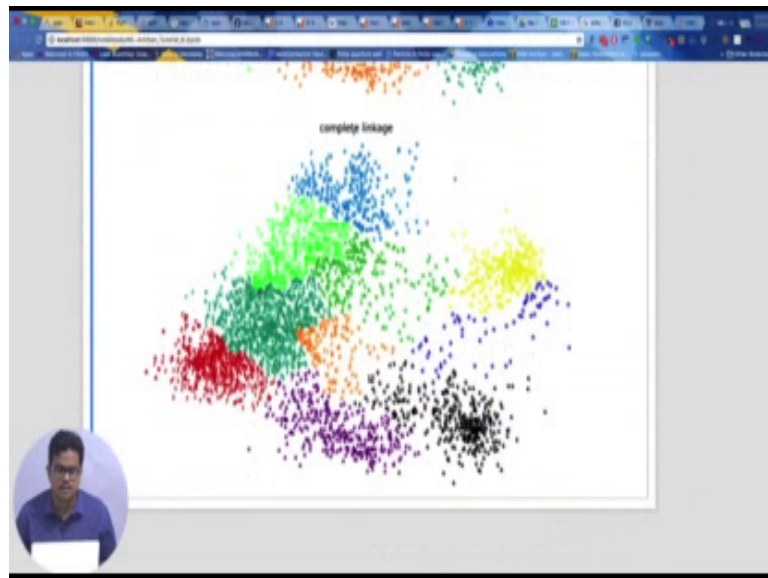
And this is the result of complete linkage right and you can see certain degree of homogeneity in the labels over here. And there are the number of ways in which you can evaluate these clusters I have not included those codes in this particular exercise, but you can definitely look up the official scikit learn documentation, and tutorials on different kinds of hierarchical clustering algorithm. And each of them I strongly recommend you to do so because they are quite you know enlightening and really good really good examples really interesting cool ones.

So, just go ahead look at the official documentation of these functions get your hands on them you know bring your own data, you just try to implement them on your and see the results, so that is all for today. See, you in the next video and it will be tutorial session on this on this same content right and that will be the end and this in fact, is the final the last hands on coding session of this course. And I really enjoyed spending these tutorial sessions these like hands on coding sessions; these are the first of my career. And thank you for being with me and giving this wonderful experience. And I am really thankful to you for giving me this opportunity.

So, bye-bye, see you in the tutorial class, bye-bye.