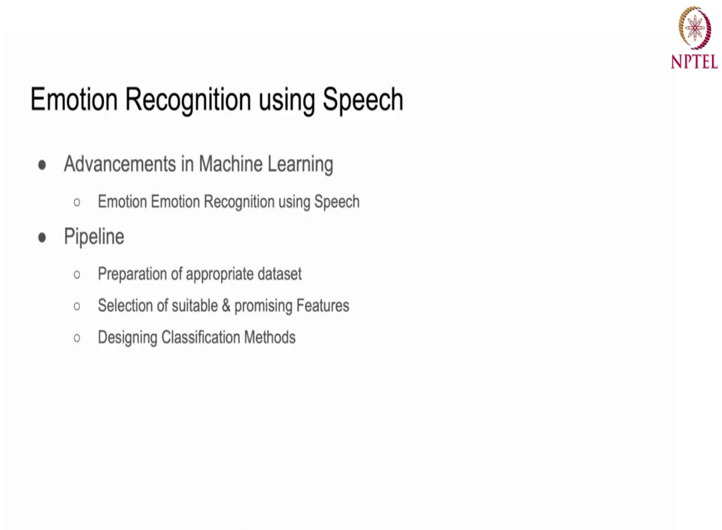
Affective Computing
Prof. Gulshan Sharma
Department of Computer Science and Engineering
Indraprastha Institute of Information Technology, Delhi

Lecture - 15
Tutorial: Emotion Recognition by Speech Signal

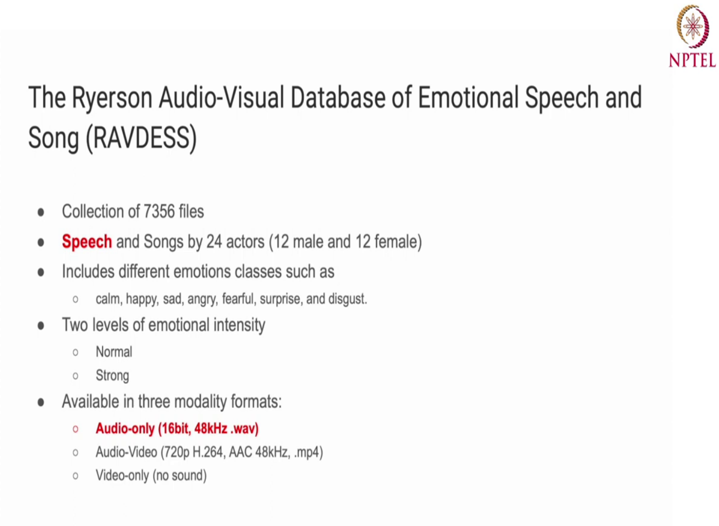
Hello everyone. My name is Gulshan Sharma and I am the Teaching Assistant for this NPTEL Affective Computing course. First and foremost I would like to welcome everyone on this very first tutorial of this course. We will attempt to learn emotion recognition through speech in this tutorial.

(Refer Slide Time: 00:40)



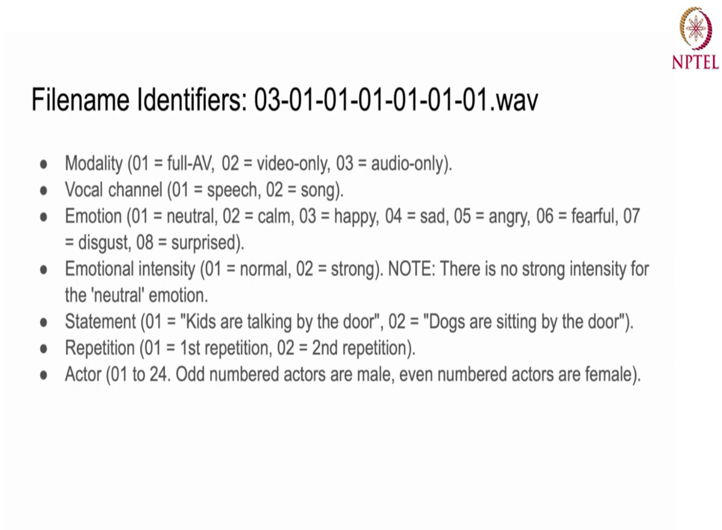
With recent advances in the field of machine learning, emotion recognition via speech signal has dramatically increased. Various theoretical and experimental study have been conducted in order to identify a person's emotional state by examining their speech signals. The speech emotion system pipeline includes repression of an appropriate dataset, selection of promising feature and design of an appropriate classification method.

(Refer Slide Time: 01:13)



So, in this tutorial, we will be utilizing a publicly available dataset known as RAVDESS. The RAVDESS dataset consists of 7356 files. The database includes speeches and songs from 24 actors, 12 male and 12 female. Emotion classes include calm, happiness, sadness, anger, fear, surprise and disgust. And this dataset is available in 3 formats audio only, video only, and audio and video. So, for our task, we will be used only speech power that is our audio only files.

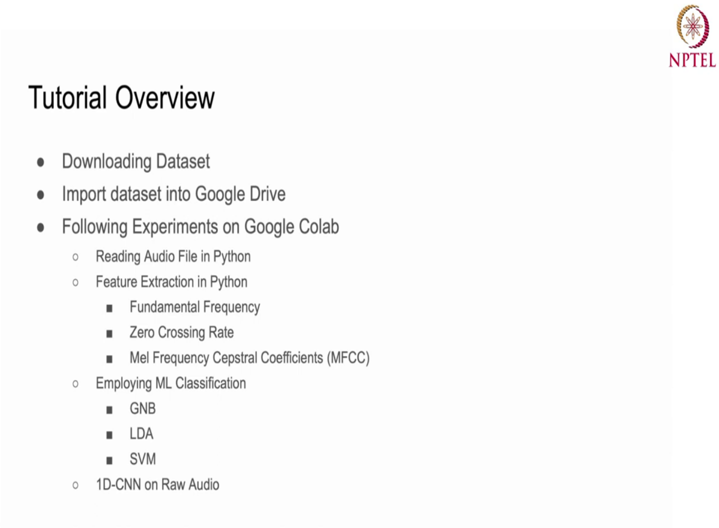
(Refer Slide Time: 01:57)



Now, moving towards the file name convention in this dataset, the file name in this dataset consists of 7 identifiers where the first identifier tell us about the modality either it is a full audio video file, video only file or audio only file. The second identifier will tell about the vocal channel, it is either a speech file or a song file. The third and the most important identifier is our emotion identifier, which will tell about the class of the emotion, neutral, calm, happy, sad, angry, fearful, disgust or surprised.

The fourth one is the emotional intensity either the emotion is of normal intensity or the strong intensity. Later on, fifth identifier will tell about the statement. And sixth will tell about the repetition of that statement. And the seventh identifier will tell about the actor. Odd number actors are male and even number actors are female.

(Refer Slide Time: 02:55)

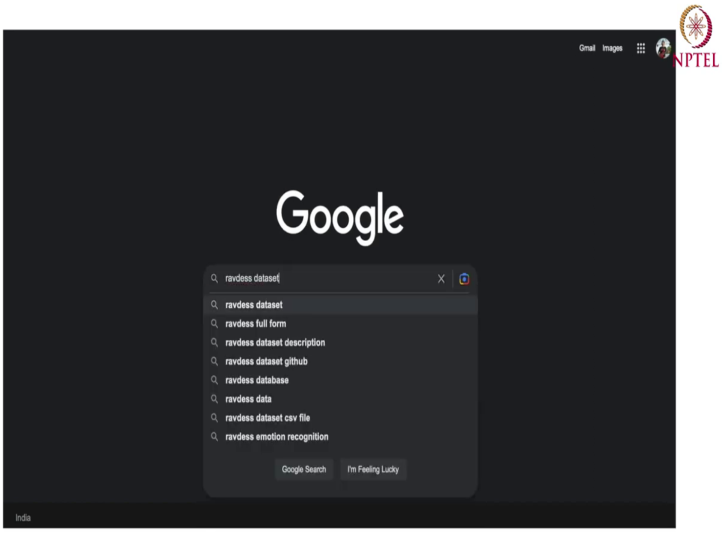


So, before starting the coding part, let me first give you the complete overview of this tutorial. We will start with downloading the dataset. After downloading the dataset we will import that dataset into a Google Drive. The reason behind importing the dataset into Google Drive is that we will be using Google Colab for our experimentations.

And our experimentation will start with reading audio file in Python. Then, we will extract these fundamental frequency zero cross rates and Mel Frequency Cepstral Coefficient as our features from the audio files. And after that we will be employing some of the classification algorithms like Gaussian Naive Bayes, Linear Discriminant Analysis and Support Vector Machine.

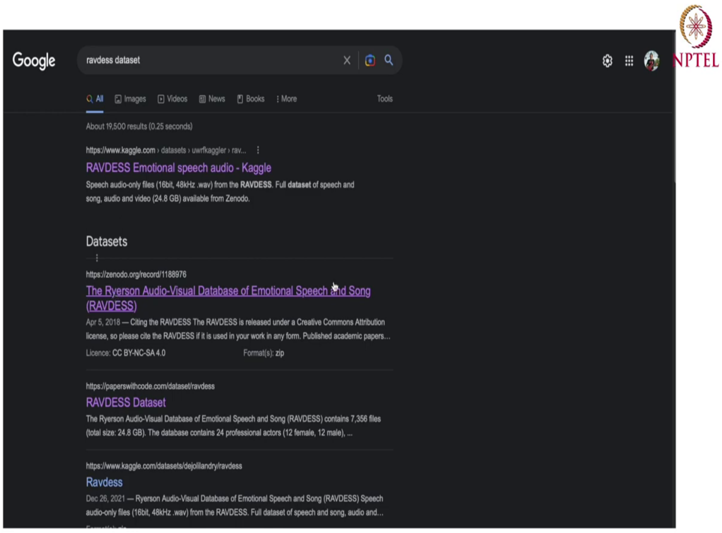
And in the end, we will also try to create a 1-dimensional convolution neural network over the raw audio for the emotion classification.

(Refer Slide Time: 04:06)



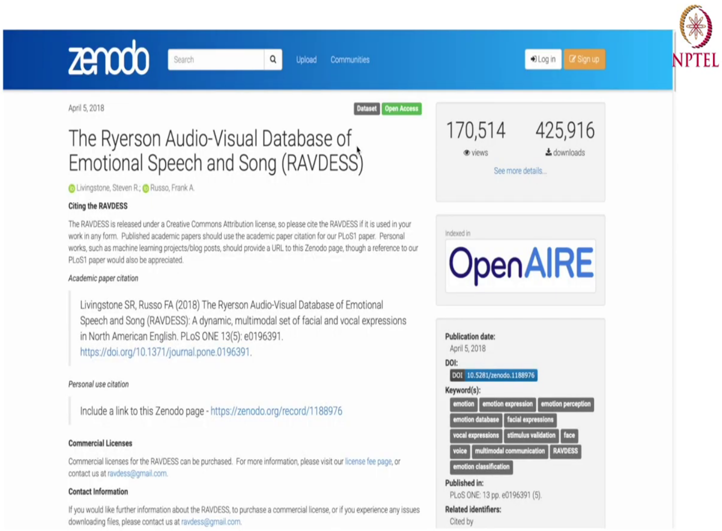
So, starting with our very first exercise which is dataset download, we can download this dataset by simply searching the RAVDESS on Google.

(Refer Slide Time: 04:10)



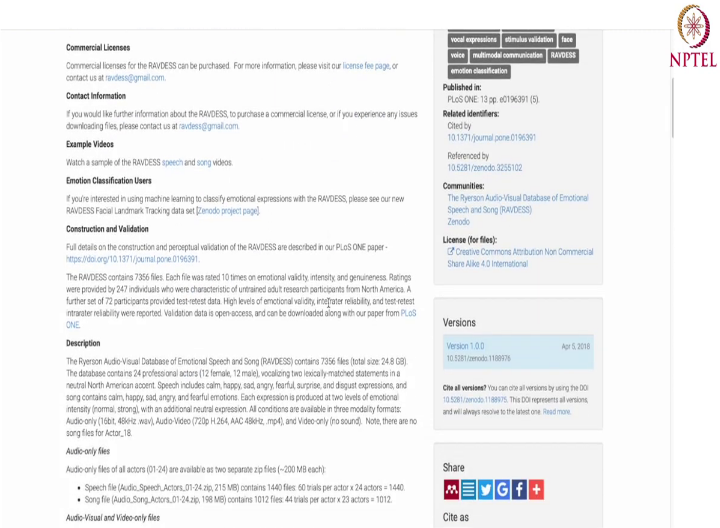
After putting this RAVDESS keyword on Google, you will find this zenodo link over here. So, basically, zenodo is a general purpose open access repository. Here we can store our data up to 50 GBs.

(Refer Slide Time: 04:29)



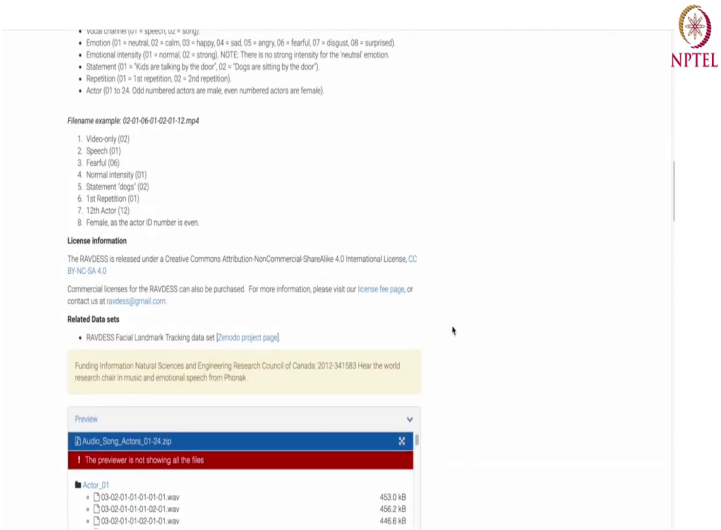
So, we can simply click on this link, and find the dataset.

(Refer Slide Time: 04:32)



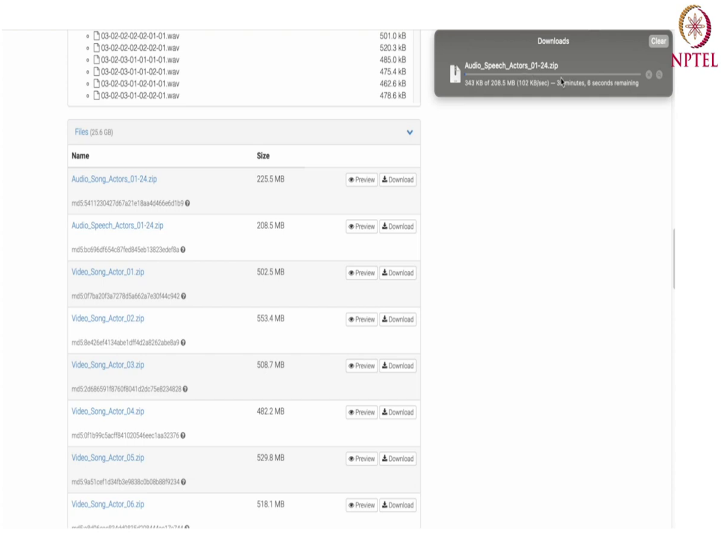
So, this dataset is basically released under creative common attribute license. So, one can openly use it for the publications.

(Refer Slide Time: 04:42)



And to download the dataset, and to download the our exact part which is audio speech actor dataset we can simply click on this link.

(Refer Slide Time: 04:47)



It will take some time to download, but in my case I have already downloaded this dataset. And I can show you.

(Refer Slide Time: 05:05)

After unzipping the downloaded file, this dataset will look something like this. So, there will be 24 folders each belonging to one actor.

(Refer Slide Time: 05:19)

And after going through one folder, we will have a couple of files over here. Or maybe I can just play a couple of files just for your reference.

Dogs are sitting by the door. Dogs are sitting by the door. Dogs are sitting by the door. Dogs are sitting by the door. Dogs are sitting by the door.

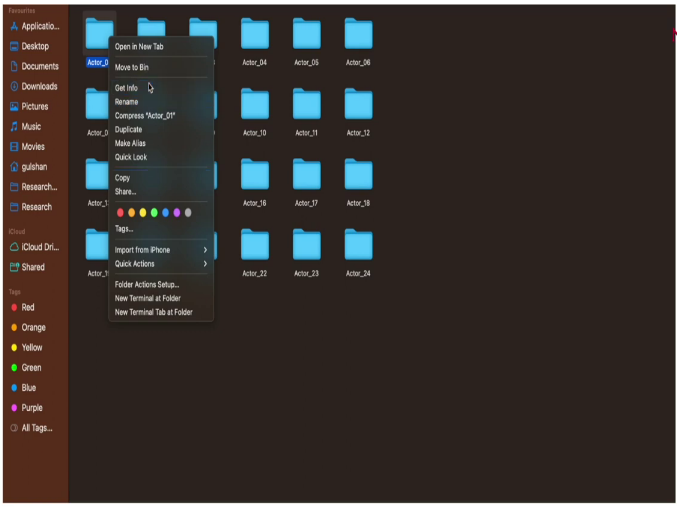
So, as you can see there are multiple emotions saying this line, dogs are sitting on the door. And if I move to some other folder, let us say actor 2 folder and play a couple of files.

Kids are talking by the door. Kids are talking by the door. Kids are talking by the door. Kids are talking by the door.

So, as we can see like there are a couple of variations in this speaking style representing different different type of emotions. So, as we have downloaded this dataset, now our next task is to upload it on Google Drive, so that we can easily access it through a Google Colab. I believe most of us can easily upload a folder on Google Drive. So, I will be skipping that part.

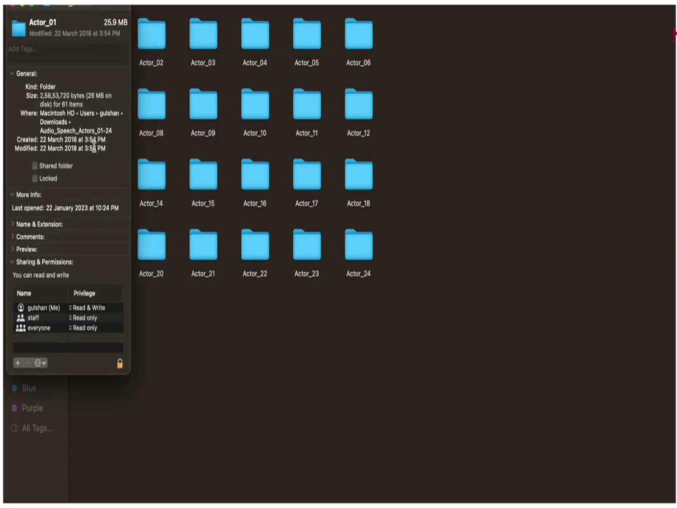
But with some of the participants it could be a situation that they are having a low bandwidth internet connection, these participants can take any of the folder and upload it on a Google Drive. So, let us suppose you are taking folder number 1.

(Refer Slide Time: 07:10)



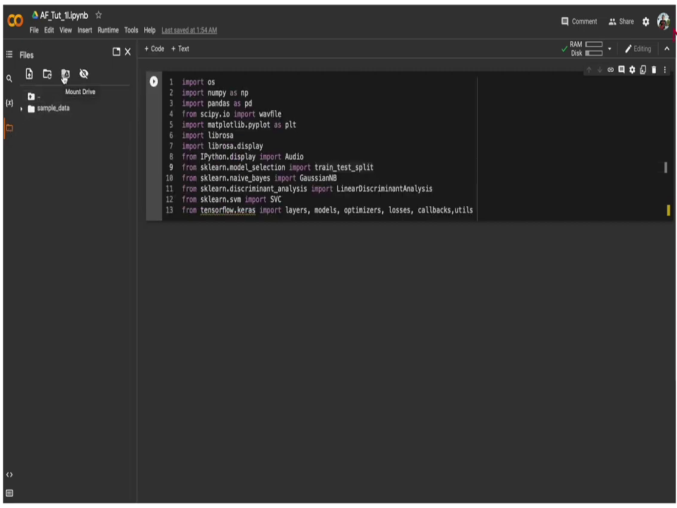
So, folder number 1, I believe is of 25.9 MB. So, it will not be a very big file to upload on a Google Drive.

(Refer Slide Time: 07:14)



So, now, I will shift on the Google Colab and we will start writing a program for emotion recognition.

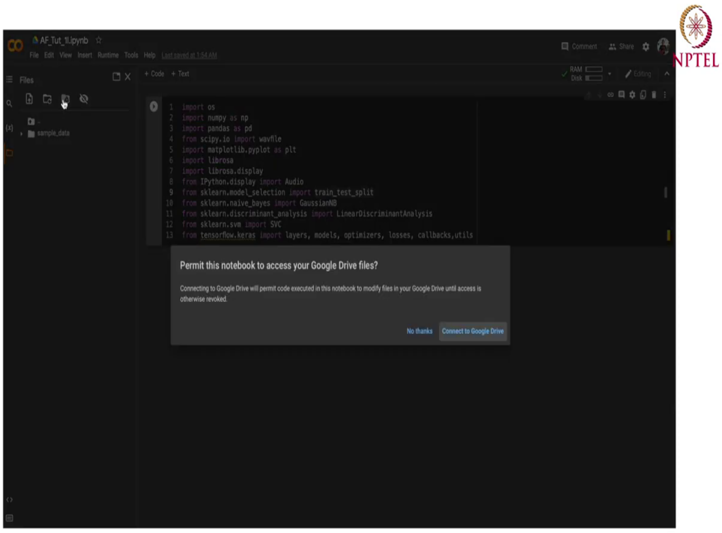
(Refer Slide Time: 07:33)



So, before starting our programming exercise, so before starting our programming exercise I assume that everyone has some sort of experience with Python programming language and everyone is aware of Google Colab Interface. We will start this exercise with importing couple of libraries and the helping functions. To save some time I have already copied required import code. So, everyone who is programming along with me can pause this video and write this code in their own environment.

So, after importing the required libraries and helping functions, we will start with reading the audio files using Python, but before that we need to mount Google Drive with our Colab Interface. To do so, we will first click on this files icon over here and then select mount drive option.

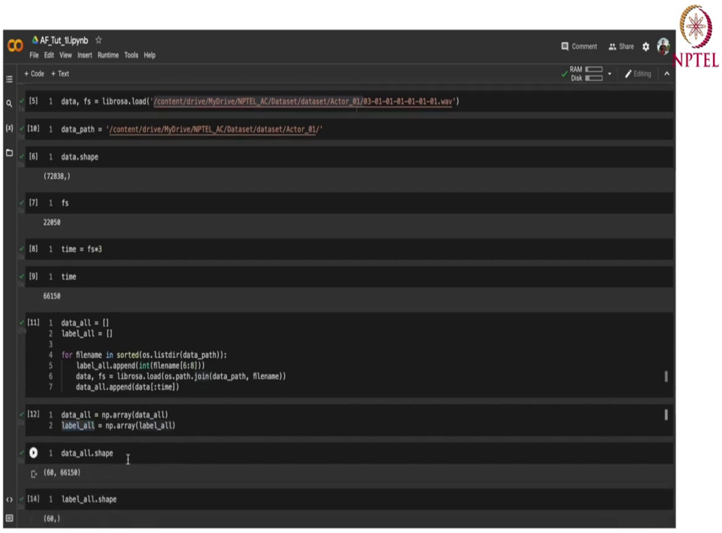
(Refer Slide Time: 08:44)



After pressing this button, you will find a dialog box over here asking for permission to access Google Drive. So, we will simply click on connect to Google Drive. So, after mounting Google Drive with Colab environment, we will now import the data. I am also assuming that some of the participant does not have enough powerful machine or high speed internet connection.

So, to simplify our job, I will be using data from a single subfolder. Since, we are importing audio data, so to read audio data in our Python environment, I will be using librosa dot load function.

(Refer Slide Time: 09:27)



I will create two variable called data and sample rate, then I will simply write librosa dot load and inside the brackets I will pass the I will pass the file name. So, as you can say this function has successfully run, and now I will show you the shape of the data. It consists of 72838 sample values and our sample rate is 22,050.

So, for more simplification I am planning to use just first 3 seconds of our audio data. So, calculating first 3 seconds of data, I can simply multiply our fs by 3, our sample rate by 3 and we will get the exact time. So, first 3 seconds of time. So, it will be equivalent to 66150 samples.

Now, we have imported just one data. So, we need to import all the data inside this folder. To do so, I need to write a piece of code where I will be sequentially reading all the files and saving them into a Python list. So, let us start with our code, I will name my variable as data all I will be also extracting all the labels. Since, we have already seen that data file name contains their respective label. So, we need to extract the relevant label also.

So, I will start with a loop where I will be reading all the file names in a sorted function from os dot list dir and in this list dir I will be importing that data file, I mean the data path of the data folder. So, maybe I can just create a another variable which is data path and this will be treated as a string and I need to write the exact data path for this folder. So, I will simply copy it from this place and paste it over here, ok.

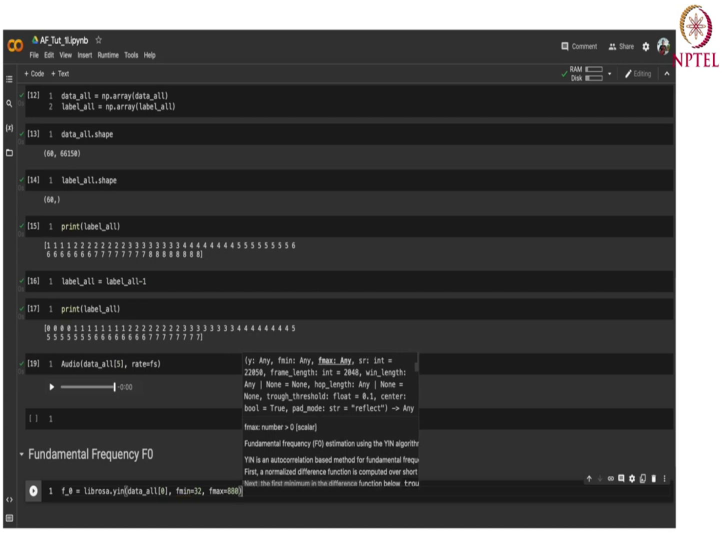
Now, I can simply use this variable wherever I want this data path. So, first I will try to extract all the labels. So, for that I will be pending all the labels, in this label all list and I will simply read the file name and extract the substring, does that sub identify from that particular file name. Maybe I will I also need to write classes to integer as these little bit treated as label, so that that could approach.

Now, I can simply read my data and also my sample rate limb librosa dot load. And now, now this line; this line of code will simply read all the file names and append it with the our data path and then the librosa function will read that corresponding file. Now, I need to store all those files in our data, all list. So, to do so, I will simply write all append data and also I will be you know simply using first 3 second. So, will I will put a colon, and sorry I will put a colon and type the time over here.

So, let me just run this code. It might take a couple of seconds to run this file completely, ok. So, code has been executed now. So, I will simply convert these list into numpy array. So, to do so, I will simply write data all equal to np dot array and label all be become sorry, this is a mistake over here. I need to write underscore not hyphen.

So, after converting these list into numpy arrays I can simply try to see their shape, what are their exact shapes. So, I will simply write data all dot shape, ok. So, now, we have written 60 files each consisting of 66150 sample which corresponding to 3 seconds of initial data. I can also see the label shape which is equal to 60 file. So, I guess we are good to go. Maybe I can also show you the exact labels.

(Refer Slide Time: 16:01)



So, you can simply print, ok. So, these are the labels corresponding to our 60 files. Maybe I will do a simple pre-processing over here as I want my label to start with 0. So, I will simply write a code, ok. So, let me print these labels again, perfect. Now, I will also show you a Euclidean IPython. From IPython dot display, we have imported audio. So, this function can simply play that exact audio file in our Python environment. So, let us try to play a audio file, ok.

In my case suppose, I will be using a 0th indexed file. And our rate will be equal to, our rate was fs was 22 to 50 (Refer Time: 17:24), I will simply write, I will simply type fs over here.

Kids are talking by the door.

So, as you can see using this utility, I can simply play the exact audio file in my Colab. Maybe I can play one more file over here.

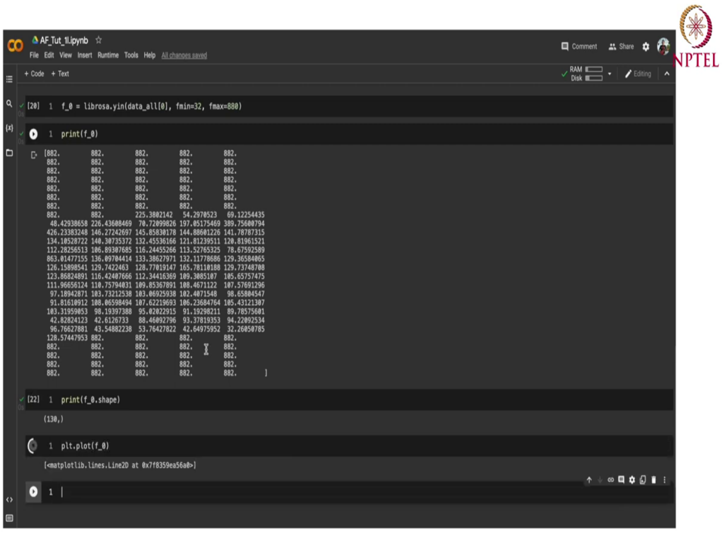
Kids are talking by the door.

Sounds good. So, after importing our data, I will now move towards some sort of a particular feature extraction phase. And in our feature extraction phase, we will be simply using fundamental frequency, and a 0 cost rate, Mel-frequency cepstral coefficients as our basic features.

So, let me show you how to extract a fundamental frequency from this audio files. And to extract the fundamental frequency, I will simply make a I am so sorry; I will simply make a variable f naught. And I will be using a library function called librosa dot yin. And in this function, I just need to pass a data instance with a range of, range of frequency value like minimum frequency value and the highest frequency value.

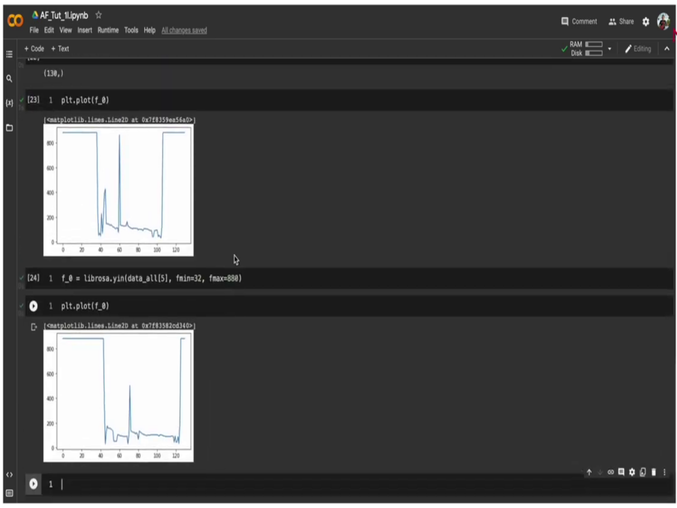
So, let me just extract fundamental frequency for a single instance, then I will show you how to do it for a whole folder, ok.

(Refer Slide Time: 19:10)



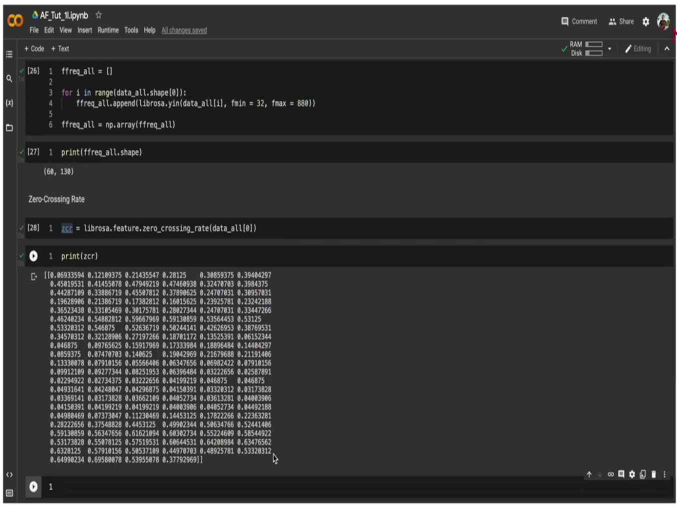
So, it is working. I will simply print the exact value of this fundamental frequency or maybe I can also try to show you a plot where these values are plotted against this time. To do so, I will simply type plt dot plot and I will simply pass the array.

(Refer Slide Time: 19:36)



So, yeah, as you can see that initial values are somewhat around 882, then there was some sort of a variation in this part and then again it is going to 882 over here. Or, maybe I just need to show you another file, let us say 5 and then I will print another plot over here. So, as you can see over here that there is some sort of a difference between the fundamental frequencies in two different emotions.

(Refer Slide Time: 20:25)

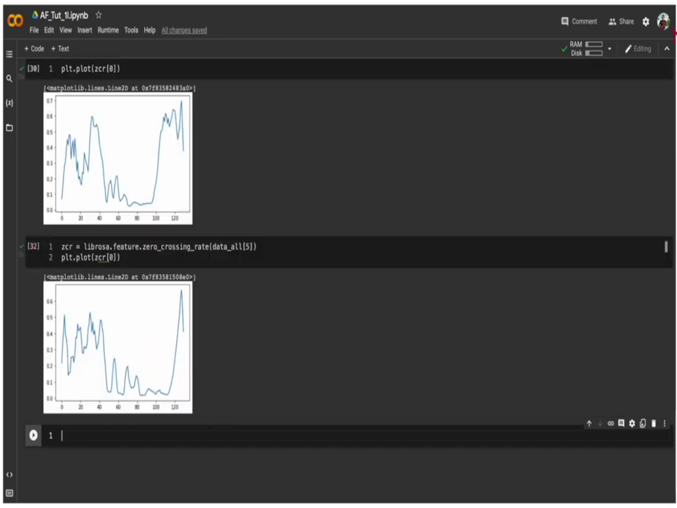


Now, let me simply extract the fundamental frequency for all the data. For that I will simply write another list where; to do so, I will be simply using a for loop, where I will iterate over all the data and extract our fundamental frequency. So, I will simply append this. Later I will convert it into a and by later I will; later I will convert it to a numpy array, this exact list. This is copied, see it sometime.

Let me print the shape of our; now, let me print the shape of this variable, ok. So, we have selected the fundamental frequency for each and every file in the folder 1. Now, I will go with another sort of feature known as our 0 cross ratings. So, I will be extracting 0 cross rate over here. So, to do so, I will be again using librosa library, and there is a function in it called 0 crossing rate which will be giving us the exact 0 cross rating corresponding to these audio files.

So, let me use the variable name as zcr and then I can use, then I can write; let me show you with the single file first. (Refer Time: 23:49) it is working. Let me show the print. Let me print the zcr, ok. So, you can see there is some sort of differences over here. Maybe I can give you a better visualization by simply plotting this over the time.

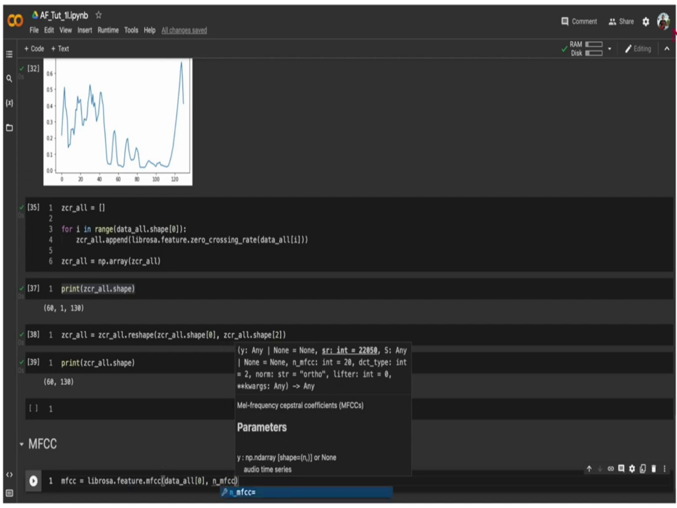
(Refer Slide Time: 24:11)



So, for that I will be writing plot, ok. Maybe I will create the zcr for another emotion. See for emotion number 5 and then plot, ok. There is an issues over here, ok. It is not a cr, it is zcr.

Yeah, here we can also observe that there is some significant amount of differences between these two features. So, I will simply write another code to you know extract the zcr value for all the files.

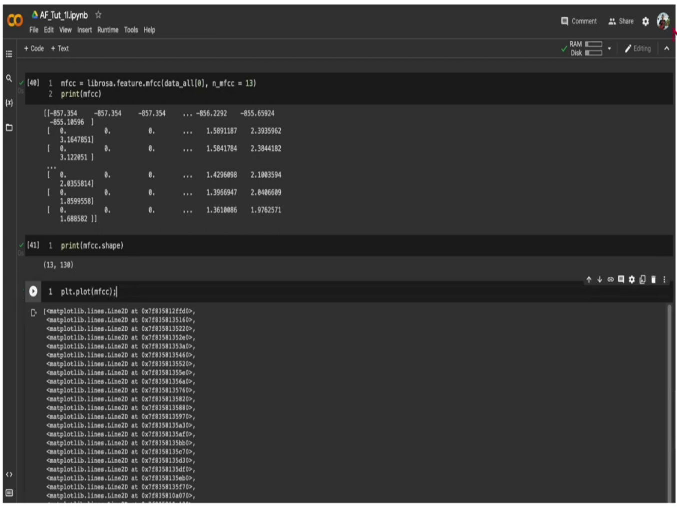
(Refer Slide Time: 25:19)



Now, maybe to just keep it consistent with my previous feature shape which was 16 to 130, you can also reshape this zcr all function. And now, if I run my print function again, so yeah we will got a similar shape over here. This is just to keep back the consistency among the all the feature. Now, I will show you two extract another feature called Mel-frequency cepstral coefficients.

So, for that maybe I just need to let us write MFCC separate over here. And yeah, I can simply extract MFCC from librosa. And yeah, there is a parameter in MFCC, it is called number of MFCC coefficient. In my case, let us say, we will be extracting first 13 coefficients, yeah.

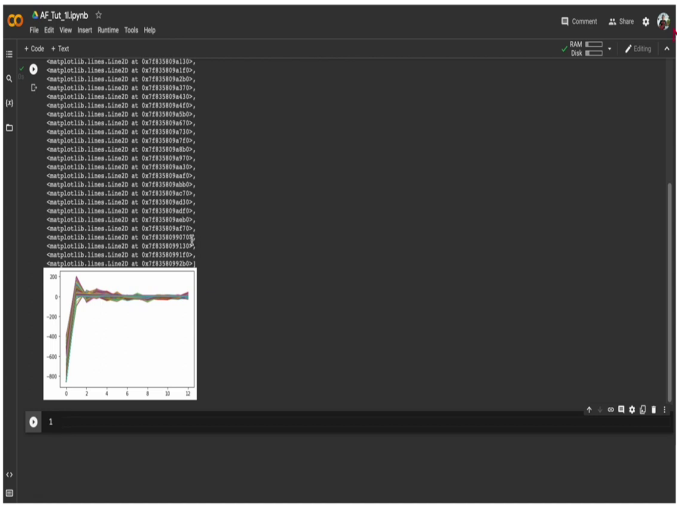
(Refer Slide Time: 26:27)



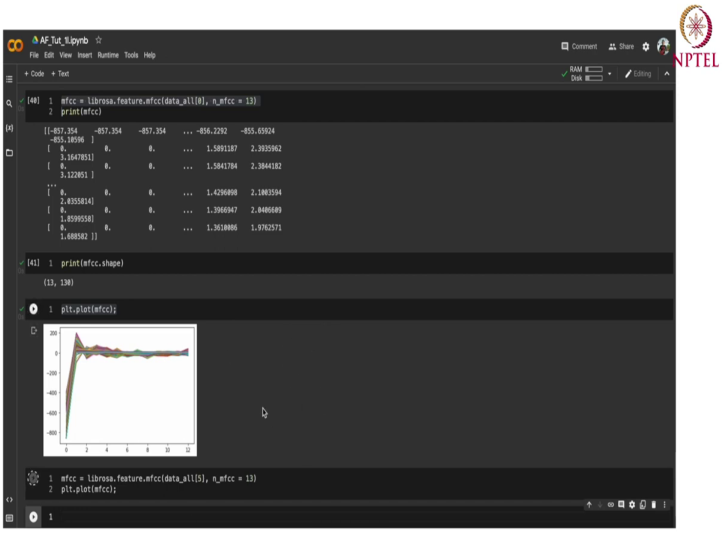
So, these are MFCC coefficients. I will like to print its shape also. So, yeah, there are 13 cross 130 for a single (Refer Time: 26:42). So, in this case, as we are getting 13 rows and 130 columns. So, for each row there are 130 values and each of these 13 values corresponding to one MFCC coefficient.

Now, to visualize this I can simply type and I can simply write MFCC, ok or maybe just to avoid these values I can simply put a semicolon.

(Refer Slide Time: 27:12)

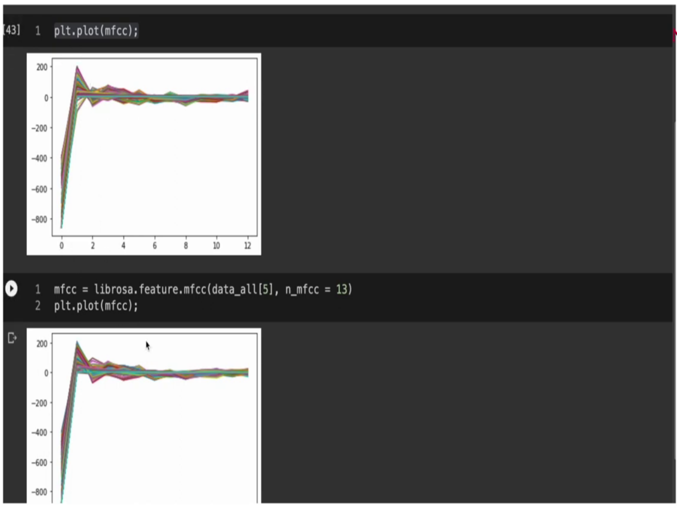


(Refer Slide Time: 27:23)



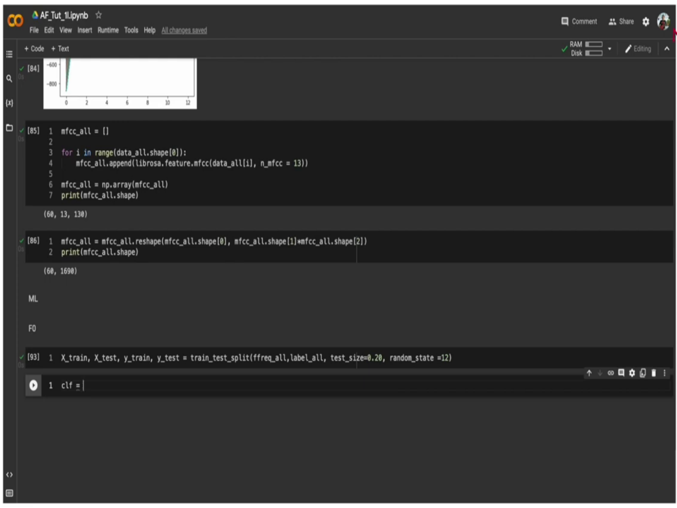
Now, let me plot it for another file. Let us see for file number I mean in the 5th file and our MFCC coefficient will look something like this. So, there is some significant differences over here.

(Refer Slide Time: 27:42)



I believe yes, I can see some differences over here. There are some differences over here also. And since, it is a very complex and very tightly bounded lines, but yeah there are some significant differences over these two files. So, now, again we will simply extract all these MFCC for all the files.

(Refer Slide Time: 28:17)

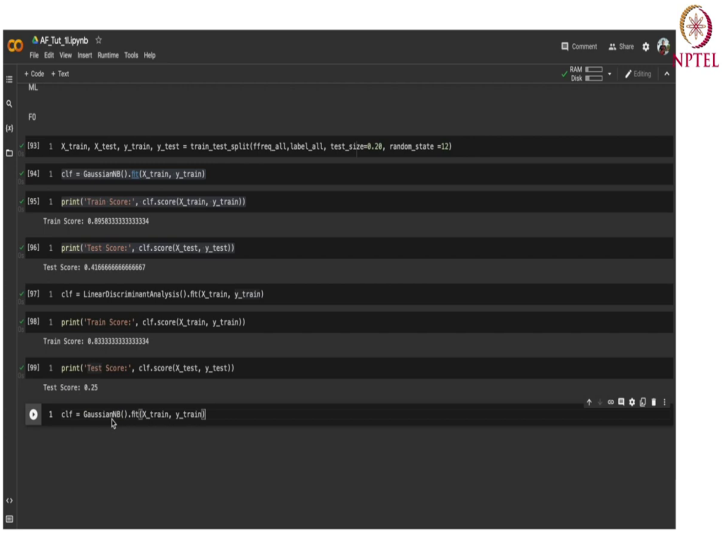


So, now yeah it looked consistent with my prior representations. And now, we have extracted all 3 features of each basic features like MFCC fundamental frequency and 0 cost rating. Now, after this I will be using my basic machine learning inferences. In my machine learning algorithms, I will be using Gaussian Naive Bayes, linear discriminant analysis and support vector machines.

So, now, moving towards the one machine learning part, in machine learning part we will take one feature divide it into their respective train and test parts, and then run our classifier over it. So, starting with the our very first feature which is fundamental frequency. So, let me first divide it into their train and test part. So, for this division, I will be using train test split function from sklearn library. So, code will look something like this, ok.

Now, I can simply run my classifier as clf equal to see my first classifier which is let me show you, Gaussian Naive Bayes. So, I have already inputted like from as sklearn dot Naive Bayes import Gaussian Naive Bayes.

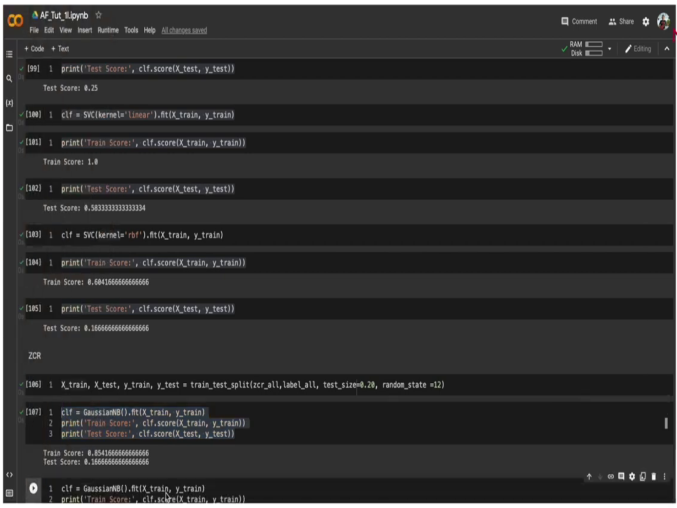
(Refer Slide Time: 29:37)



So, I will simply copy it over here and function over here and fit it on my training set, ok. Now, my classifier is fit on our training set. So, let me check about the training accuracy over here, ok. So, we are getting 89 percent of training accuracy. And let me also check for the testing score, training accuracy I will get, ok; so, I am getting 83 percent of training accuracy over here using lta which is lesser than our Gaussian Naive Bayes.

And let me also check out my test score, ok. So, yeah test score is also getting down to 0.25 percent. So, I believe Gaussian Naive Bayes is performing better in our fundamental frequency. So, guys let me try my third classifier now which is support vector machine. So, for that I will also again reuse my code. And instead of Gaussian Naive Bayes, I will be using our SVC function over here.

(Refer Slide Time: 30:53)



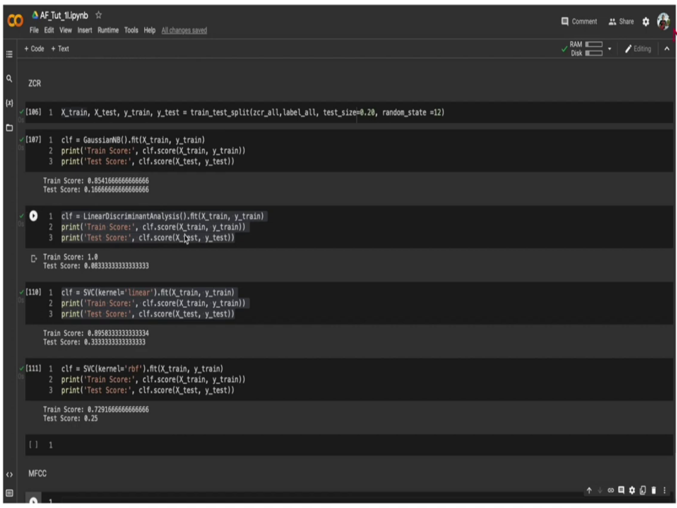
So, I will replace Gaussian Naive with SVC and inside SVC I need to define my kernel, which kernel I will be using. So, kernel equal to let us say we start with a linear kernel and let me check, ok yeah. So, classifier is set on our training data. Let us see about our train score, ok. So, with unit classifier we are getting 100 percent training accuracy. Let me check the test score over here. We are getting 58 percent of accuracy which is you know higher than all of other classifiers.

So, maybe I can also check it with another kernel called RBF kernel, ok. RBF kernel is not getting with that good accuracy. And yeah of course, our testing score also decreased towards 16 percent, which I believe is a chance level. So, yeah, in our case for a fundamental frequency, we can easily see that our support vector machine with linear kernel giving the best results, ok.

Now, let us try a similar classifier using another feature. So, after fundamental frequency, our next feature was 0 cross rates. Let me code similar stuff for zcr. Again, I will be you know simply reusing my code over here. So, instead of f frequency all, I will be using my zcr underscore all and rest of the part will be same, ok. Now, my train and test variable this setting is over the zcr features.

So, again I will simply reuse my code. I will be using Gaussian Naive Bayes over here. And again I have to show my train and test accuracies. So, I will simply use this code, ok. So, for Gaussian Naive Bayes in case of our 0 cross rates, the train accuracy is somewhat around 85 percent, but the test accuracy is around chance level only.

(Refer Slide Time: 33:27)

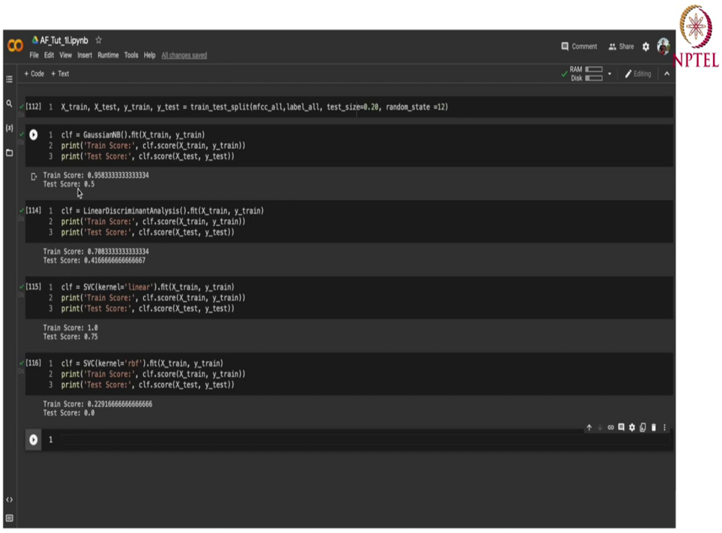


So, we will try with another classifier which our linear discriminant analysis. Again, I will simply reuse my code. So, for linear discriminant analysis we are getting a train accuracy of 100 percent and test accuracy is somewhat around 8 percent which is very lower than I guess chance level. And this is a clear example of overfitting in this case. In fact, this is also example of overfitting.

Let me try with support vector machine now. So, you simply change for a function to SVC, SVC and then we use kernel equal to linear, ok. Some probelm over here, ok. I forgot to put equal to. In this case results look little bit better than, I mean Gaussian Naive Bayes and linear discriminant analysis, but still there is a huge variance between train score and a test score. So, it is another example of overfitting only. Let me try with the RBF kernel, same case overfitting.

So, now, moving towards our final feature, final manual feature that we have extracted MFCC, let us try to run similar code using MFCC. Again, in re-usability of code.

(Refer Slide Time: 35:24)

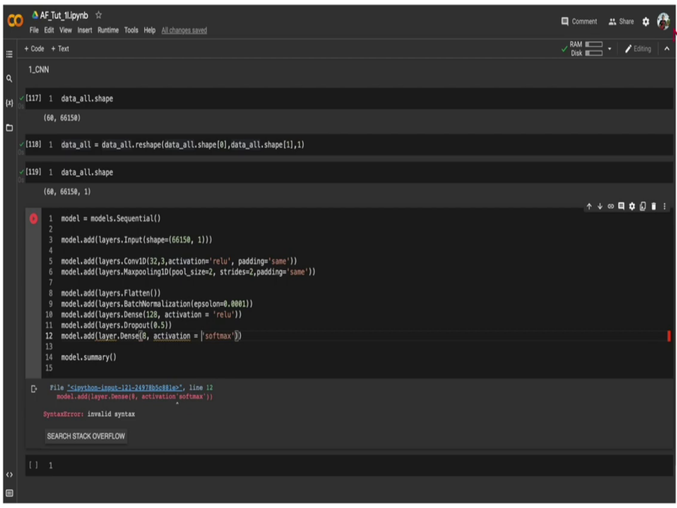


Now, training over Gaussian Naive Bayes classifier for MFCC feature, ok again, ok. These are comparatively better result. We are getting trained accuracy of 95 percent and test accuracy of 50 percent. Now, let us try with linear discriminant analysis, ok. 70, 41, and in case of now in case of our support vector machines, 1 and 75 which was a good result for linear as a linear kernel.

And in case of RBF kernel, ok (Refer Time: 36:34) simply you know working over here. So, yeah, as we can see that support vector machine with linear kernel is giving best results. Now, as we all can see that we have used our basic feature on our basic classifier. After this, I maybe I can do one more exercise where we will be using a raw audio data over a one-dimensional convolution neural network.

And that one-dimensional neural network will you know automatically extract relevant features out of the audio data. And using a soft mix classification method we will simply classify the emotion classes. So, to do so, I will be using a help of a library called Keras. And before that I need to do some minor settings in my environment in terms of data shape.

(Refer Slide Time: 37:30)



I need to just reshape my data, so that I can fit it in a convolution neural network. So, what I am basically doing here is I mean let me show you the exact shape of our data before running this code. So, my original data shape was 60 cross 66150, ok. And after you know reshaping this data my data shape will become. So, my next part will be how to code a 1D CNN. We will be using a library called Keras. I have already imported these library.

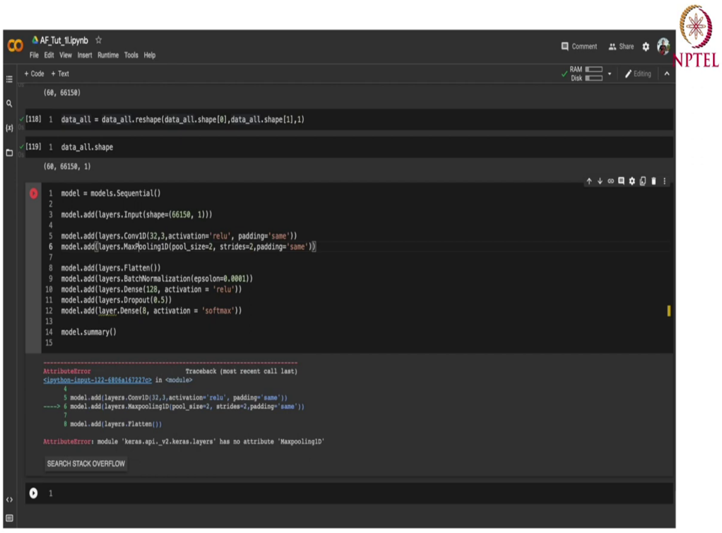
So, to start with I will simply write model equal to models dot sequential, S may be capital over here. I will first define the input shape which is essentially a layer, my network dot input a shape will be a touple consist of 66150 cross 1. So, I will simply copy it. And after inputting the data of this shape, I will add a convolution layer over it.

So, the code will look something like model dot add. For activation function we will be using relu and padding will be same. After a convolution layer maybe I will try another layer called max pooling or average pooling. Let us say I will use max pool, ok. Maybe I will just use single convolution layer and try to see how my result changes with this network.

Maybe I can put a batch normalization layer, then a dense layer (Refer Time: 39:53) layer dot add see number of neuron equal to 128, with activation equal to relu, ok. Now, maybe we can also add a dropout layer you know just to avoid any overfitting case. And as a final layer, we will simply add a dense layer with number of neuron equal to 8 which is equivalent over number of classes and activation will be Softmax, just for the classification purpose, yeah.

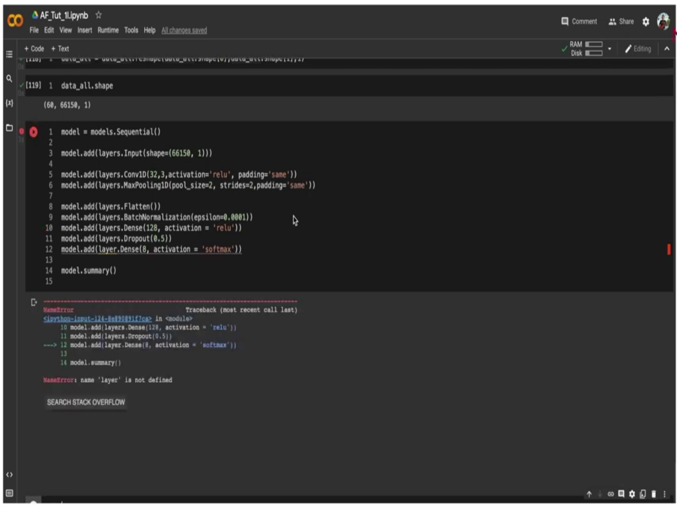
Now, maybe I can just present the summary of this model, ok. There is some error syntax error over here, ok. I forgot to put a equal to over here, ok. Another error activation I again forgot put the equal to over here, no attribute max pooling 1D. Let me check, ok.

(Refer Slide Time: 41:14)



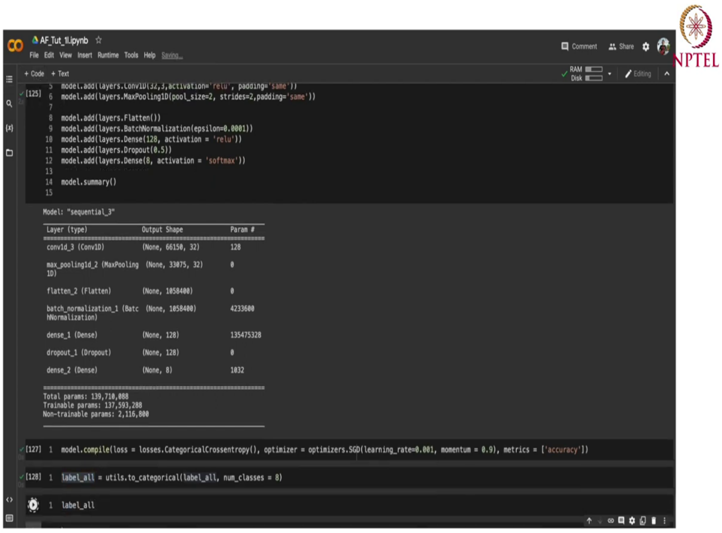
Another error attribute max pooling, ok. The P in max pooling will be in capital, ok. One more error, ok epsilon spelling is wrong.

(Refer Slide Time: 41:34)



One more error, I forgot to put s over here.

(Refer Slide Time: 41:45)

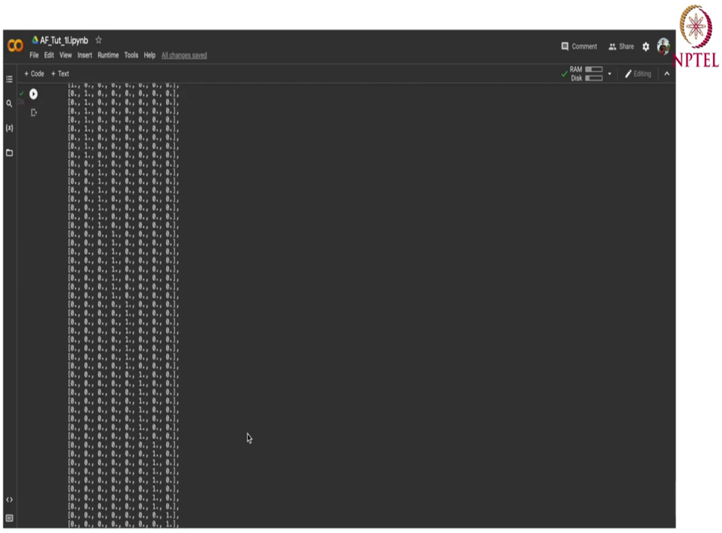


Yeah, now it is working. And this is the summary of our model and these are total number of parameter that the model will be tuning. Out of these parameter, this much will be trainable parameter and others will be non-trainable parameters. Now, we have defined our the structure of our network. Now, we can simply compile this model using model dot compile.

Now, in compilation of model we have to define a loss function, but exact loss function we will be using and a optimizer, but sort of optimizer we will be using to you know optimize that loss function, ok. There is a error over here, sequential model has no attribute compile, ok. I have written the wrong spelling. So, after compilation of my model, I just need to fit my model over a training data.

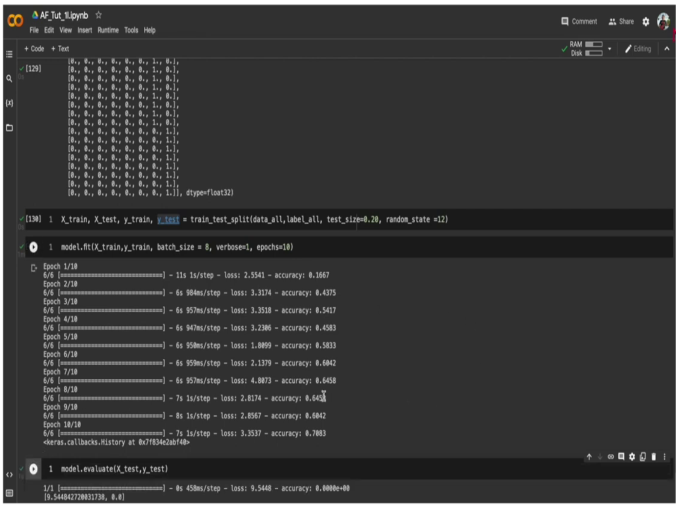
But before dividing our data into train and test split, I just need to convert my labels into categorical classes in terms of one hot encoded vector, since we are using categorical curve course entropy loss. So, for that I will be using in build function in tensor for chaos, ok. This has changed my labels into one hot code encoded vectors. Let me show you, yeah.

(Refer Slide Time: 43:23)



So, instead of a single value over here, we have this vectors.

(Refer Slide Time: 43:31)



Now, I can use my train and test splitting code to you know divide this data into train and test split. I will simply copy my previous code and increased it over here. And this time my training data will be our raw audio data, ok. So, after dividing to train and test split, we have to simply fit our model, but we have already defined and compiled through model dot fit. It might take some time to you know as we are, right now we are just using our CPU you know Google Colab. So, it might take some time to learn, ok.

We can see that our accuracy is improving over here, ok. So, my model has now completed all its epochs. So, let us try to evaluate what test accuracy is over here. We are getting training accuracy of somewhere around 70 percent and test accuracy is 0 percent, ok. So, this, my this network architecture is not learning anything as of now.

Maybe I can you know start with some sort of a hyper parameter tuning, start adding a couple of layers in it or maybe using different sort of activation functions or decreasing or increasing the number of neuron in dense layers. All that sort of hyper parameter tuning we I can do to make this network a better classifier.

So, guys, this was all about this tutorial. Hope I was able to give you a basic idea about programming on these sort of networks. And if you have any sort of doubt, feel free to put a question in discussion form.

Thank you.