

Affective Computing
Dr. Jainendra Shukla
Department of Computer Science and Engineering
Indraprastha Institute of Information Technology, Delhi

Week -11
Lecture - 01
Case Study: Emotional Virtual Personnel Assistance

Hi friends, welcome to this week's module which is a Case Study. And in this week, we are going to talk about how to develop a Emotional Virtual Personal Assistant as part of this case study.

(Refer Slide Time: 00:40)

Agenda



1. Motivation and Challenges
2. Methods and Techniques
3. Ethical Concerns
4. Interaction with Dr. Aniket Bera (Purdue University, USA)



So, basically so far, we have learned a lot about the theoretical aspects of a fatigue computing. Now, the idea of this week's lecture was to bridge the gap between the theory and the practice for you. So, here we will try to understand how can we apply that we have already

learned in the previous weeks to develop an application, to develop a device which is emotionally intelligent and can be used in real life.

I will try to provide as much codes as well and hopefully you will get to play with it in your own time. So, here is the outline for this week's module. First, I will be talking about of course, the motivation behind the development of such an emotionally intelligent virtual assistant. What could be some of the challenges? I will be talking about the methods and the techniques using which such an assistant can be developed.

And I will be talking about the ethical concerns around such a system when it is deployed in real life. And I am glad to inform you that we also have managed to do an interaction with Dr. Aniket, who is a faculty in Purdue University. And his most of the work is related to the effective computing and emotionally intelligent machines.

And we will be talking with him at the last of this lecture to understand how has he been you know investigating in this area and how he has been developing some of the real world applications by making use of his research in his lab at Purdue, ok. So, with that let us get started.

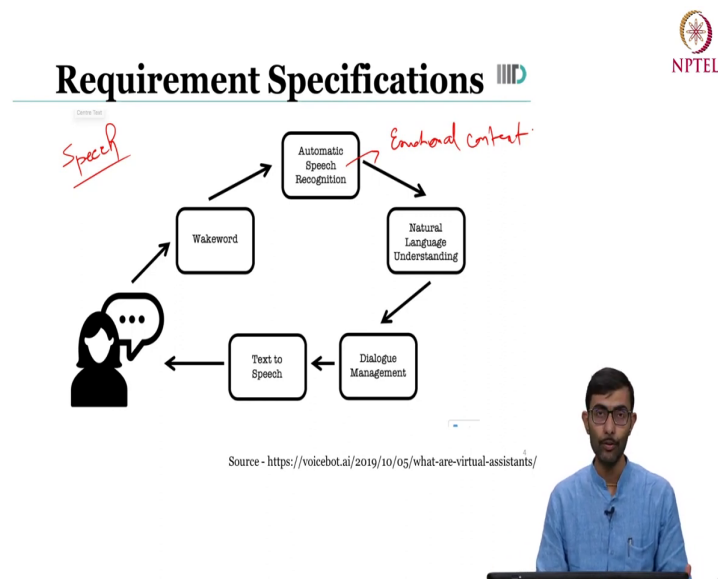
(Refer Slide Time: 02:23)



Motivation and Challenges



(Refer Slide Time: 02:26)



So, first thing that we want to understand is what exactly is a virtual assistant. Now, when we talk about a virtual assistant, basically virtual assistant is something that we all have seen is just like Alexa, Siri, Cortana, etcetera. And the idea of virtual assistant is that it takes an input or it interacts with the humans and it takes as an input their voice. And it could be started with a follow some wake word such as, Hey Siri, Hey Google. And then of course, since speech is the modality that we are taking it as an input.

So, please pay attention that speech becomes the mode of the communication. Then the next thing happens is the automatic speech recognition. Of course, the assistant tries to understand what is that the user wants to communicate or wants to understand. And this is where you know when we talking about the automatic speech recognition, this is where we also we will

try to understand the emotional context of the entire interaction between the agent and the humans.

Once we have understood the intent and the emotions that are there in the speech, we will try to of course, you know apply the NLP and try to understand how can we respond to this particular request that the user has made. And then there is of course, dialogue management system which helps to create this interaction between the user and the agent.


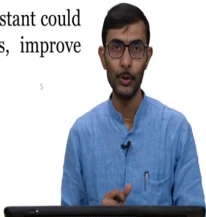
And then once you are the agent is ready with the response, then the agent you know converts the whatever response that it has in the text format to a speech format and send it back to the user. So, as simple as that, you may ask for example, Hey Siri, how is the weather today?

And Siri would respond that, ok, the weather is sunny and warm for example, right? And this is where you know we want to insert the emotional intelligence component into it. So, what we are talking about here is trying to develop an Alexa, Siri or Cortana kind of assistant, but which is also emotionally intelligent. So, I hope that it is exciting and fascinating enough for you.

(Refer Slide Time: 04:42)

Use Cases

- **Mental health and therapy:** An Emotional Virtual Personal Assistant could be used to help people with mental health disorders or emotional struggles.
- **Customer service and support:** An Emotional Virtual Personal Assistant could be used in customer service and support industries to provide a more personalized and empathetic experience to customers.
- **Education:** An Emotional Virtual Personal Assistant could be used in education to help students learn more effectively.
- **Entertainment:** An Emotional Virtual Personal Assistant could be used in entertainment to create more engaging and interactive experiences.
- **Workplace productivity:** An Emotional Virtual Personal Assistant could be used in the workplace to help employees manage stress, improve communication, and increase productivity.
- What else???



But if not, then let us try to look at that what could be some of the use cases of such an emotionally intelligent virtual assistant. So, for example, this kind of assistant can be used in mental health and the therapy, where the what is this kind of assistant can be used to help people with mental health disorders or for example, emotional struggles.

This kind of agent, it can recognize the user's emotional state and it can provide guidance and it can you know support with the coping strategies and emotional support in the real time. And on the top of it, this kind of an agent, it can track the user's emotional state over a period of time. And it can provide insights to the therapists and to other healthcare providers or let us say you know to their nearby stakeholders to their nearby relatives and friends.

So, that is one interesting application. Other application that could be is the customer service in providing the customer service and the support. So, in this case, such an agent can be used

in to provide a more personalized and empathetic experience to the customers. And the idea is here that such an agent while doing an interaction with the customer will be able to recognize user's emotions.

And it will be able to tailor its responses accordingly to provide let us say more positive and satisfactory responses to the user. And we all have seen you know how the customer service agents interact with us. And definitely it is a very tiring job hats off to the customer you know care agents. But of course, we can always do better in this area.

One other domain that it can be really interesting to have this kind of an agent is the education. So, in the education domain, such an agent can be used to help students learn more effectively. So, the idea is very simple, that if you have such an agent, it can try to identify when a for example, a student is struggling and you know needs more support or needs additional support.

And hence, it can help provide targeted feedback and support to them to overcome you know whatever obstacles they are facing in learning a particular topic or a concept. Other for example, very important category could be the entertainment. So, basically this kind of an agent can be used in entertainment to create more engaging and interactive experiences which can be really of lot of demand.

So, for example, imagine that you have a video game which is using this kind of virtual assistant. It can adapt to the player's emotional state and maybe provide a more immersive and you know adaptive environment for to for the user to enjoy. So, this can be a very interesting application.

Other interesting application that frankly speaking I and all my colleagues in the academia would definitely appreciate a lot is having an virtual assistant that can be used in the workplace to help employees manage their stress, improve communication and increase productivity.

So, it is very common for the work place for the individuals who are working in a particular workplace to get overwhelmed from time to time with the anxiety and stress and the lots of tasks that are there hand. So, the idea is that, can this kind of system provide understand you know when the employee is feeling for example, overwhelmed or frustrated.

And maybe it can provide support and additional resources to help them become more effective and overall have a positive experience while conducting their job. So, that is one and of course, these are only few pointers that I am mentioning, but of course, there could be many more applications.



And to answer to what else I would let you think about it that what are the other domains where you would like to use such an emotionally intelligent virtual assistant. So, imagine that if you have a Alexa or a Siri or a Cortana which is also emotionally intelligent what can you do with it. So, the creativity is the only limit here.

(Refer Slide Time: 08:52)

Challenges

Data bias
gender bias

- **Accuracy:** Variability in emotional expression across individuals, cultures, and contexts.
- **Real-time processing:** Must be able to operate in real-time or near real-time to provide a seamless user experience.
- **Multi-language support:** Supporting multiple languages and cultural contexts can be challenging.
- **Bias and Discrimination:** Emotion recognition technology may reflect the biases of the individuals who create and train the models, potentially leading to unfair treatment of certain groups.
- **Privacy Concerns:** Collecting and analyzing speech data for emotion recognition raises privacy concerns, particularly if the data is collected without the user's knowledge or consent.
- What else???



So, I hope that you have been feeling motivated enough to have such a system. And to develop such a system and to go through the case study of it. Now, of course, such a system is not going to be easy to develop. So, let me make your hopes not so high, but of course, we will see what could be some of the challenges. And by no means this is an exhaustive list. I am just trying to provide some pointers here.

Of course, the very first challenge that is there is the accuracy. So, what we want? We want of course, such a system to be as accurate as possible, but it can be really difficult because there is lot of variability in the emotional expressions as across individuals, cultures and the context. And I hope you can see that the type of the modalities that we are talking about here is the speech modality.

So, you may want to recall the concepts that you have learned during the emotions in speech lecture by my colleague Dr. Abinav. So, the idea is that ok, here there is lot of variability. And this variability can be due to different many different factors which was discussed in that particular lecture.

So, for example, it could be due to the speech patterns due to the you know accents and different dialects, right. And as simple as that for example, when you are making a speech there is a even bigger challenge apart from the dialect and accents and everything is, how can such a system detect sarcasm or irony in a speech when it is communicating with the humans.

We humans are very good at making you know sarcastic comments and of course, we use it very efficiently in our day to day communications. So, if we were to use such a communication with an emotionally intelligent agent, how can such an agent detect that kind of thing. And of course, these are can be you know the agent has to look into the tone also has to understand the context and so many other things.

So, nevertheless accuracy remains a big challenge for this. Of course, other big challenge of such a system that we can envision is the operational requirements in real time. It has to be able to perform in real time. Because if not real time then we will not be able to provide a seamless user experience, right.

And of course, what it requires? It requires the processing and the response time should be very very efficient. And of course, it has to be supported with a high performance hardware and the software. So, this comes as a trade off between what we want to invest and what we want to gain out of it. But nevertheless, the idea is that we should not be able to make the compromise with the user experience. Otherwise, the users will not interact with it and it will lose its purpose, ok.

So, third point is the multi-language support. Multi-language support, I think this is even more applicable to a country like us like India, where we already have 20 plus official

languages. And unless and until we are able to cater to the needs of all the languages, we will not be able to reach to the big portion of the individuals in the domain.

One other interesting challenge is of course, is of the bias and the discrimination. And it turns out that like any other technology, this particular technology or the virtual assistant that we are going to develop. It may also reflect the biases of the individuals like us and also, like the data sets, who create and train the models and it can potentially lead to the unfair treatment of certain groups.

And this biases that we are talking about and hence resulting discrimination can be due to several factors. It can be for example, due to data bias. So, for example, the data that has been recorded. Imagine if we are recording a data that is only making use of individuals or the participants from northern part of India.

Then of course, we may not be able to cater to the exact requirements of the users from the southern part of India for example. Other thing could be is the gender bias. What is gender bias? Imagine that we are having a data where all the participants or majority of the participants, they are belonging to one particular gender, say male.

And then there are very few females in the category. Then of course, the model may have a very hard time in trying to understand the emotions from other gender or other genders in this case. Similarly, of course, we can have a culture bias as well, which we already talked about in the last thing.

Culture bias is basically, you know it turns out that there are certain regions of certain regions where people may have a higher tone in comparison to the other regions. And similarly, there could be other cultural differences. So, taking into unless and until we are able to understand the entire demography of the participants, which are our target users, we will not be able to completely get rid of the bias that may be present in the such a system that we are trying to develop.

And last, but not the least, of course, there is a privacy concern can be a big concern when it is comes to the development of the such virtual assistant. And the idea is of course, we should not be able to collect the data without the user's knowledge or their consent. And of course, even if we are collecting the data, we need to have a very robust data encryption protocol. We need to have a secure storage of such a data and all the transmission protocols.

And we need to have the compliance with the privacy regulations, not only at the national, but also at the international levels such as GDPR and the HIPAA regulations that we have now. So, this is the very, very important requirement. And these are some of the interesting challenges that can be there in while developing such a system.

And of course, as I said, I mean these are by no means an exhaustive list. So, there could be many other challenges. And I invite you to please brainstorm about it. What do you think could be more challenges when it comes to the development and the usage and the deployment of such a emotionally intelligent virtual assistant, ok.

(Refer Slide Time: 15:15)

- essential*
- ## Requirement Specifications IIIID
1. The system should be able to recognize emotions in speech and distinguish at least the basic emotions of happy, sad, angry, and neutral.
 2. The system should be able to process natural language input in voice format.
 3. The system should be able to understand user intents and generate appropriate responses based on the user's input and recognized emotions. *(intent + emotion)*
 4. The speech emotion recognition and virtual assistant components should be able to operate in real-time or near real-time.
 5. The system should protect user data, including speech data and personal information.
 6. The system should support multiple languages and be able to operate in different cultural contexts. *→ good to have*



So, now having understood that what exactly we are talking about, we are talking about a development of an emotionally intelligent virtual assistant. In short, its Alexa or Siri plus having emotional intelligence. We already saw that can be have lots of use cases in the education, entertainment, mental health, workplace productivity so on so forth. And then we also looked at lots of challenges.

So, having looked at all those things, now we can try to formulate some of the requirements that it may have. Please note that the exact requirements may vary from one to another and it will depend on the business use case that you are trying to have and which may come from you know the managers or which may come from the business stakeholders also.

So, of course, the very first requirement that whatever system we are developing, it should be able to recognize emotions in speech as simple as that. And let us say that you know for the

sake of the simplicity, taking as an example, it should be able to distinguish at least the basic emotions of happy, sad, angry and neutral. So, we are just talking about the basic emotions and rather than talking about you know like the entire spectrum of the emotions to keep our life easier. Of course, we have already gone through the emotions in speech.

So, we are going to rely on that understanding a lot. If you have not gone through that or if you do not recall some of the concepts, then I will invite you to please go through that and revise emotions in speech and then maybe come back to this part later. Of course, this is the first requirement. Of course, second requirement is it should be able to take the data as an input in a voice format. So, we do not expect the user to type like a chatbot, but we expect the user to say something using a speech modality.

So, it should be able to recognize the emotions in their speech, it should be able to take which underlying hypothesis, it should be able to use the speech as an input. Of course, while doing so, the idea is that it not only should be able to understand the emotions, but also it should be able to understand user's intents and it should be able to generate appropriate responses.

So, for example, maybe the user is asking something and it should be able to understand what is the user asking and there may be certain emotions attached to it so both. So, intent plus emotions. And as of now, most of the time the virtual assistants that we see, Alexa, Siri, Cortana and all that, they are able to understand the intents, but maybe they are not they do not have a lot of understanding of the emotions or emotional intelligence, right.

And of course, it should be able to provide generate appropriate responses taking into account both the intent plus emotions, ok. That is the difference between the agent that we are trying to develop and the agent that has been developed in the past. And of course, we already established that unless and until its working in real time environment or in a near real time with a near real time efficiency, it will not be able to serve the need of the user.

And having raise the concern of the privacy, of course, we already understand now that we should be able to protect the user data, primary we are talking about the speech data and of

course, we may have access to certain demographic information such as you know the user's identification information and location and all that.

So, we should be able to protect user data, speech and the personal information. As I said by no means all this is an exhaustive list. And for example, we can keep on adding the requirements to it and one nice requirement to have would be. So, I would say that these, the top 5, these are something that we can term it as a essential requirement because without having it, I we may not be able to you know launch even to a particular city or even a particular region.

And this is something that we can call it as you know good to have kind of thing. But of course, if it is not, if you do not have this multilingual support, we may not be able to cater to the multiple users to a user-static scale. But nevertheless, having top five is a good start for us.

And of course, as I said, the exact specifications of such an agent may depend on the business use cases. And I will invite you to brainstorm about it that what could be the business use case that you are targeting and what additional requirements could be there for such a business use case, perfect.

(Refer Slide Time: 19:51)



Method



So, having understood now that what exactly we want to develop. So, we want to develop an emotionally intelligent virtual personal assistant. Just like Alexa, Cortana, Siri, but with an emotional intelligence. I will keep repeating it again and again. So, that you understand it very thoroughly. Now, let us try to dig in a bit more technical aspect of it, that how exactly can we develop such an agent?

(Refer Slide Time: 20:21)

Skelton Code

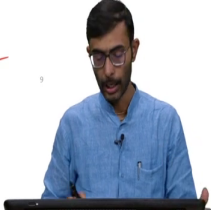
```
import speech_recognition as sr ✓

# Initialize the speech recognition and language identification objects
r = sr.Recognizer()
language_identifier = ...

# Initialize the emotional virtual personal assistant components
virtual_assistant = ...
emotion_recognition = ...

# Initialize the context object
context = ...

# Start the conversation loop
while True:
    with sr.Microphone() as source: # Prompt the user to provide voice input ✓
        audio = r.listen(source) ✓
```



NPTEL

And I think we know already the answer to it. But of course, here we will try to put different pieces together and hopefully it will make some sense. So, here is a very skeleton code that I have tried to prepare which gives rough idea about what are the different steps that we are going to follow to develop such an agent, ok.

Of course, imagine you know this skeleton agent code has been developed in Python. I will provide you the code files as well, just so that you can play with it. So, for example, you know this is of course, you are going to have to input some libraries such as speech recognition as simple library which.


And then you know of course, you are going to initialize the speech recognition and the language identification objects, you know create some initial objects and do some initialization. You are going to initialize the virtual personal assistants depending upon you

know you may be using a particular set of hardware and the software. So, you want to initialize them.

And then this is where you know you are going to keep track of the user's personal preferences and users for example, history over a period of time and things like that and that is where we are going to just call it as a context. So, basically you would like to keep a track of the user's context and hence you may have a context object as well and you want to keep updating that. Then of course, you know then we are going to this is something that is going to run in a infinite loop continuous loop and basically zero.

So, that is why we have a while true loop of course, while true then we are going to have we will keep listening to the user's input. And once we listen to the user's input, we are going to get an audio of it.

(Refer Slide Time: 21:57)

```
Skelton Code   
# Determine the language of the user's input  
language = language_identifier.identify(audio) ✓  
  
# Recognize the user's emotions from the speech input  
emotion = emotion_recognition.recognize(audio) ✓  
  
# Process the user's natural language input to understand their intent and context  
input_text = r.recognize_google(audio, language=language) ✓  
intent = virtual_assistant.process_input(input_text, emotion, context)  
  
# Generate an appropriate response based on the user's input, recognized emotion, and context  
response_text = virtual_assistant.generate_response(intent, emotion, context)  
# Output the response in voice format  
response_audio = TextToSpeech.convert(response_text, language=language)  
response_audio.play() ✓  
  
# Update the context object based on the current interaction  
context.update(intent, emotion) ✓
```



Once we have the audio of it then of course, depending upon what are the functionalities that we have, may be one of the first thing that we may want to do, we may want to identify what is the language in which the user is speaking. Having identified the language, may be may of course, the next step would be to understand what is the emotion that is there in the this particular speech data.

We may want to you know understand; convert this thing in the text format to be able to do some more processing over it. We may want to having identified the emotions; we also want to identify the intent. For example, if the user is asking something, saying something, commenting something, things like that. And so, you may want to understand the intent of the user as well.

And then of course, then taking into account please pay attention to this point, taking into account the intent, the emotion and the context. So, the context could be the user's preferences, users history over time, how the user has been interacting with the system, maybe the system is able to you know understand what sort of preferences the user is developing while having the interaction with the agent. So, this is the context.

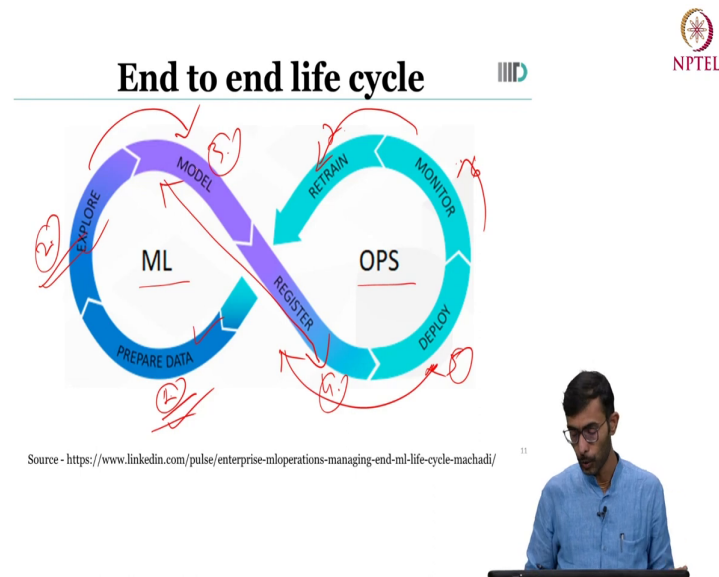
So, basically now the appropriate response is going to be developed with the help of the intent, emotion and context. These are the three things that are really important for such an emotionally intelligent virtual agent. And after that of course, you know once we have a response text.

We can simply convert the text to a speech because we want to convert communicate back to the user in the form of a speech. And once we have this response audio, we can simply play the response audio in a speaker and then that is how the user is going to understand to it. And of course, then accordingly if we have certain update in the user's context, we can do that update in the context and then this can go on you know in a while low.

Roughly speaking, this is the skeleton of the our emotional intelligent virtual assistant. As you can see, this is really, really, really simple. But of course, this captures the gist of the

entire system. I will invite you to please go through the skeleton code that will be provided to you with this week's lecture module, perfect.

(Refer Slide Time: 24:11)



So, we have understood the top level idea. Now, let us try to see what how an end-to-end life cycle of this system will look like. So, this turns out that the end-to-end life cycle of a system is not going to be very very different from a typical machine learning or deep learning systems, where for example, we have two major components of it.

One is the machine learning aspect of it and one is the operational aspect of it, where we have the deployment and all that. So, we will talk about all these aspects in bit detail, but basically when we talk about the machine learning aspect of it, what it means? Of course, we need to have some data.

We are going to you know, play with the data, do some pre-processing, feature selection extraction, all those kind of things. And then after playing, we are going to create a machine learning model, which is going to you know in this case, going to help us understand the intent, the emotions and maybe the context also to a certain extent.

Of course, then we are going to you know, do the registry of such a model and then we deploy in real system. And once we have done the deployment, then maybe of course, we will have to keep monitor the usage and the working and the functioning of such a system. And as and when required, we may want to re-train the system and having retrained, we may create another version of the model. And then that is how you know the operational deployment it goes on.

So, basically this is the rough roughly the end to end life cycle of our virtual assistant will look like. One aspect that we are not focusing a lot here is the hardware aspect is because it is understood that of course, along with this software, we may want to give it a shape of a hardware, a tangible interface, which is could be in the form of you know like for example, a Google speaker or for example, any device which is you know sitting on your table.

But nevertheless, there is going to be an hardware plus software component. Here, we are focusing more on the software, but you can easily integrate the hardware development part into it as well, perfect. So, now we have understood the end to end life cycle. So, the very first step is the preparation of the data.

And this is I would say this is the second step, I would say this is the third step, this we can call it as the fourth step, this is the fifth step and of course, this is the sixth and the seventh step as required. So, let us talk about the very first step, which is the data or the preparation of the data.

(Refer Slide Time: 26:52)

Data



- A dataset of audio recordings and corresponding emotion labels, such as the Berlin Database of Emotional Speech (EmoDB) or the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)
- Text data labeled with emotion categories, such as the Affective Text dataset or the EmotionLines dataset.
- Example of the training data:
 - Audio signal: "I am so happy to be here!"
 - Emotion label: happy ✓
 - Audio signal: "I'm feeling really sad today"
 - Emotion label: sad
 - Audio signal: "I'm so angry with you right now"
 - Emotion label: angry
 - Audio signal: "I don't really care one way or the other"
 - Emotion label: neutral ✓



So, ok, before I discuss it, what do you think what could be the type of the data that we will need? So, our idea is that we want an agent which is able to recognize the emotions in the speech. If you recall your concept and understanding from the emotions in speech module, then of course, what we need? We need a data set of audio recordings and hopefully corresponding emotion levels.

And there are some datasets which are existing. And for example, we have talked about the EmoDB and RAVDESS datasets. So, maybe we can make use of either of these datasets. Apart from this, there is this IATKGB datasets also, also that was discussed in the class. And we may you may want to use the data set as well. Because for example, EmoDB and RAVDESS, for example, they do not cater to the Indian population, because they have been recording recorded mostly using western participants.

Nevertheless, for the sake of the simplicity, let us assume that we take one dataset, it is the EmoDB dataset. What exactly this EmoDB dataset or these datasets that we are going to use will have or need to have? Of course, they need to have the text data and the speech data that is labeled with the emotion categories.

And such as, for example, what we are talking about is imagine that we have an audio signal, which is saying, I am so happy to be here. There is an audio, there is a corresponding text, and then of course, there is an emotional label associated with it, which is happy. Similarly, we have another audio signal; I am feeling really sad today. There is an audio, there is a corresponding text, and there is an emotional label to it, which is sad.

Similarly, for example, we have another audio signal, I am so angry with you right now. Audio corresponding text, audio, corresponding text, and then there is this emotional label angry ah. And then just for the sake of completeness, there is another type of audio signal, which could be you know, for example, I do not really care one way or the other. It is something like you know a like neutral tone. So, you have a speech data, which says, I do not really care one way or the other.

Then you have a text data corresponding to it, and then you have a emotional label attached to it, which is the neutral label. So, this is sort of data set that we are going to have. Of course, it the exact choice of the data will depend on so many different things, ways.

Maybe you may want to curate your own data set as well, but nevertheless curating your own data set so far, I hope that you understand it can be a significant time consuming process, and it will provide significant resources, depending upon your need and the requirements you may want to use the existing data sets. Just for the sake of this case study and example, we are saying that, ok, let us just go ahead with the using of using the EmoDB data set.

(Refer Slide Time: 29:24)

Skelton Code



```
import os
import librosa
import pandas as pd
import numpy as np

# Set the path to the Em008 dataset
data_path = 'path/to/Em008/'

# Define the list of emotions we want to classify
emotions = ['happy', 'sad', 'angry', 'neutral']

# Set the sampling rate for the audio files
sr = 16000

# Define a function to preprocess the audio files
def preprocess_audio(audio_path):
    # Load the audio file
    y, _ = librosa.load(audio_path, sr=sr)

    # Apply pre-emphasis filtering
    y = librosa.effects.preemphasis(y)

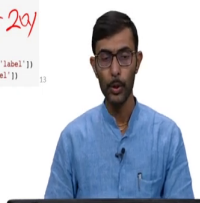
    # Resample the audio file to the desired sampling rate
    y_resampled = librosa.resample(y, orig_sr=16000, target_sr=sr)

    # Normalize the audio file
    y_norm = librosa.util.normalize(y_resampled)

    return y_norm

# Define a function to load the Em008 dataset
def load_dataset(data_path):
    data = []
    for emotion in emotions:
        # Get the path to the audio files for the current emotion
        emotion_path = os.path.join(data_path, emotion)
        # Get the list of audio files for the current emotion
        audio_files = os.listdir(emotion_path)
        # Iterate over the audio files and load each one
        for audio_file in audio_files:
            audio_path = os.path.join(emotion_path, audio_file)
            # Preprocess the audio file
            y = preprocess_audio(audio_path)
            # Get the label for the audio file
            label = emotions.index(emotion)
            # Add the audio file and label to the dataset
            data.append((y, label))
    return data

# Load the Em008 dataset
data = load_dataset(data_path)
# Shuffle the dataset
np.random.shuffle(data)
# Split the dataset into training and validation sets
split_idx = int(0.8 * len(data))
train_data = data[:split_idx]
val_data = data[split_idx:]
# Convert the dataset to a pandas dataframe
train_df = pd.DataFrame(train_data, columns=['audio', 'label'])
val_df = pd.DataFrame(val_data, columns=['audio', 'label'])
```



And hence, what we would like to do? We would like to of course, load this data and maybe you process it to certain extent. So, this is the simple skeleton code. Again, as I said, I will be providing you the code. So, basically, essentially what we are doing in this particular thing, that of course, we are importing certain libraries. You may be already familiar with certain some of them.

numpy, pandas all are basic pre-processing libraries and Librosa OS basically, you know, to connect to OS for example, in the operating system libraries and so on. So, the idea is of course, we are going to have a path to the data set. Maybe it is going to be in your virtual in your drive online drive or maybe it is going to be an audio system.

Then of course, there is the labels we already agreed about it, that we are going to have only four emotions, happy, sad, angry, neutral for now. We can of course, since we are talking

about the speech data, there is going to be some sampling frequency to it. We are going to limit the sampling frequency let us say to the 16 Hertz, 16 kilo Hertz. Then we are going to have a function to pre-process the audio files that we are going to capture.

So, basically, you know what type of pre-processing we can do? We can basically, you know, simply do the re-sampling of the data from whatever target frequency whatever frequency it had to the, maybe you know, original target frequency. So, which could be, you know. So, for example, 16,000.

We can normalize the file. So, basically, if you remember the normalization, the idea is, if the data is coming from different users, we may want to normalize it so that we should be able to make a fair comparison across different users. So, you may want to normalize it. So, that is how you know, you are going to have a normalized or pre-processed version of the audio file, that is the function to pre-process the audio files.

Next, we are going to have another function which is going to load the EmoDB data set. We assume that we are using the EmoDB data set. You can replace it with any other data set of your choice. Of course, we are going to have, you know, like all the audio files folder, we are going to have the emotion path. And for all the audio file in the audio files, simply we are going to get the y, that is the sample of it. And then we are going to get the corresponding label. So, the audio file and the corresponding label.

So, basically y and the label. And then of course, we are going to, we can simply append to the data that we have and then we can create a matrix kind of a structure for it. And this is the final data that we have already loaded from the data set, from the data set that was available in a particular folder in a drive or in a on our operating system, ok.

So, now ah, when we call the main function, what is happening? Basically, the very first thing, we are going to load the EmoDB data set. Once we have done the loading of the EmoDB data set, we are going to shuffle the data set. So, if you recall shuffling the data set is basically done to avoid any sorts of bias, you know, that can be there.


And then we simply shuffle it, you know, and have a good mix of the data that is available to us. Next, of course, we are going to split the data set into the training and the validation sets or training in the testing for the sake of the simplicity. So, for example, in this case, we decided to use 80 percent of the data for the training and the remaining 20 percent will go for the validation or the testing set of the data, ok.

Just to make our life easier, we are going to convert the entire data set into a pandas data frame. So, that to have better manipulation control over the data. And simply we converted the entire thing into a data frame, which we call the train df and the testing data frame, we call it as the validation df.


So, this is just an skeleton code. Of course, you may want to tweak it here and there in order to suit your needs, right, perfect. So, now this caters to our first part of the life cycle, which is the preparation of the data. Next, we want to explore the data.

(Refer Slide Time: 33:29)

Explore: Features



- Relevant features for emotion recognition, such as:
 - Mel Frequency Cepstral Coefficients (MFCCs)
 - Prosody features (pitch, loudness, speech rate, etc.)
 - Spectral features (spectral centroid, spectral flux, etc.)
 - Time-domain features (zero-crossing rate, energy, etc.)
- Feature selection can be used to reduce the dimensionality of the feature space and improve the performance of the emotion recognition model:
 - Principal Component Analysis (PCA)
 - Linear Discriminant Analysis (LDA)
 - Mutual Information-based feature selection
 - Recursive Feature Elimination (RFE)
- It's important to balance the number of features used with the complexity of the model, to avoid overfitting.



When we say that we want to explore the data, what it means, that we may want to understand, you know, what sort of feature selection and feature extraction can be done. We are for now for the sake of the simplicity, we are restricting to a very simple machine learning, machine learning pipes pipeline rather than you know going into the deep learning pipeline.

More or less the steps are going to be the same. So, it turns out if you recall your emotion in the speech lecture, then there are lots of relevant features for the emotion recognition. We may want to use for example, some of them, which is the MFCC coefficients. We can look at prosody features such as pitch loudness, speech rate; we can look at some of the spectral features, spectral centroid, spectral flux etcetera.

We can look at some of the time domain features. Now, what exactly these features represent? You may want to refer to the emotion in speech class to have a better understanding of it. The idea is that we may want to extract certain features, which are going to be relevant for the recognition of the speech, emotion in the speech data, ok.

So, let us say that we just identified a list of relevant features. Then of course, once we have certain features, which could be huge in number, then what we want to do? We want to we may want to perform the feature selection to reduce the dimensionality of the entire features and to optimize the performance. And there are lots of ways in which we can reduce the dimensionality of the feature space.

And for example, there is PCA, LDA, mutual information based feature selection methods there are many and then recursive feature elimination and, but, ok. So, we may want to use one or the other depending upon your specific requirements. The discussion of all these techniques is beyond the scope of the class.

But of course, I will request for the interested users to look at these techniques and understand what could be the pros and the cons of one technique over the other when we want to do the feature selection. For the sake of the simplicity, we are just going to use let us say PCA for the feature selection and the reduction of the dimensionality.

Of course, it is important to have an balanced number of features that we are going to use in accordance with the model that we are going to choose. Because it turns out that if we have very small number of features and if we have chosen a very complex model, the entire thing is going to result in an overfitting.

We want to avoid that. Of course, the basics of it can be understood in the in a common machine learning or a deep learning course. So, to summarize, we have already identified the data. Now, we are trying to identify what are the some of the features that we can use and what is the feature selection technique that can be used to reduce the dimensionality of it.

(Refer Slide Time: 36:18)

Skeleton Code



```
import librosa
import numpy as np
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score
from sklearn.pipeline import Pipeline

# Define the number of MFCCs to extract
n_mfcc = 13

# Define the number of prosody features to extract
n_prosody = 4

# Define the number of spectral features to extract
n_spectral = 2

# Define the number of time-domain features to extract
n_time = 1

# Define the number of components to keep after PCA
n_components = 10

# Define a function to extract MFCCs from an audio signal
def extract_mfcc(audio, sr):
    mfccs = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=n_mfcc)
    return mfccs.T

# Define a function to extract prosody features from an audio signal
def extract_prosody(audio, sr):
    pitch, magnitude = librosa.core.pitch(audio, sr)
    prosody_features = ShortTermFeatures.feature_extraction(audio, sr, 0.05*sr, 0.025*sr)
    prosody_features = prosody_features[:, :n_prosody]
    prosody_features = np.hstack([prosody_features, pitch], np.newaxis)
    return prosody_features

# Define a function to extract spectral features from an audio signal
def extract_spectral(audio, sr):
    spectral_features = ShortTermFeatures.feature_extraction(audio, sr, 0.05*sr, 0.025*sr)
    spectral_features = spectral_features[:, :n_spectral]
    return spectral_features

# Define a function to extract time-domain features from an audio signal
def extract_time(audio, sr):
    time_features = ShortTermFeatures.feature_extraction(audio, sr, 0.05*sr, 0.025*sr)
    time_features = time_features[:, :n_time]
    return time_features

# Load an audio file
p, sr = librosa.load('path/to/audio.wav', sr=44100)

# Extract MFCCs from the audio signal
mfccs = extract_mfcc(p, sr)

# Extract prosody features from the audio signal
prosody = extract_prosody(p, sr)

# Extract spectral features from the audio signal
spectral = extract_spectral(p, sr)

# Extract time-domain features from the audio signal
time = extract_time(p, sr)

# Concatenate all the feature matrices into a single feature matrix
features = np.hstack([mfccs, prosody, spectral, time])

# Apply PCA to the features
pca = PCA(n_components=n_components)
features_pca = pca.fit_transform(features)
```



So, as I said, we are going to use these many features. We are for the sake of the simplicity; we are going to use PCA. So, let us try to see a skeleton code which is going to help us do what we want to do for the feature exploration. So, of course, this is you know some library, initial libraries that we are going to use.

And then these are some of the parameters that we have to define in order to initialize and certain components of related to the different features and the feature selection technique that is the PCA that we have chosen. So, of course, the exact understanding can be taken by going through by going in detail about a particular feature or the particular technique.

So, once we have done that, then let us try to define different functions that can extract the different features for us. Of course, one simple feature could be is the MFCC feature as we

agreed. So, we defined an extract MCC function for that. Other function is to extract the prosody features from an audio signal which is going to be you know the pitch magnitude.

Again, some other features and then we are going to have another function to extract spectral features from an audio signal. This is again some spectral features that we have already defined. We can calculate from the given same audio input. And then we can have some time domain features from the given audio signal.

So, please pay attention that these are the skeleton code. So, you may want to tweak it or you may want to add more details to it in order to make it executable. I will be providing you all these codes for your understanding, ok. So, once we have defined how to do the feature extraction, how to do the feature you know selection, how to do the feature extraction here. Next, we can simply do the feature extraction of the selection in a main file let us say. So, we can simply load a particular audio file.

We can extract the MFCC; we extracted the prosody features using the earlier defined functions. Similarly, the spectral features, similarly the time domain features, we simply concatenated all the features into a feature matrix and we can simply apply the PCA on the top of it.

And then this features PCA is going to is going to represent the reduced dimensionality of the entire features that we have selected. And how many components since we have defined 20 components. So, this is going to reduce the dimensionality to the 20 principal components of the PCA module.

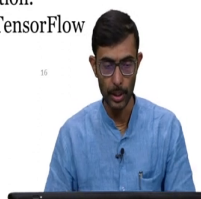
Perfect, so, I hope that now the second step is also a bit clear. So, to summarize what we are trying to do is we have done the prepared data. We have already explored the features that are there. Next, we may want to dive a bit into the model.

(Refer Slide Time: 39:09)

Model and Register



1. Define the machine learning model architecture.
2. Compile the model by selecting the loss function, optimizer, and metrics for evaluation.
3. Train the model on the training set using the fit() function.
4. Evaluate the model on the validation set using the evaluate() function.
5. Tweak the model architecture, hyperparameters, or dataset if needed and repeat steps 4 and 5 until satisfactory performance is achieved.
6. Test the final model on the testing set to get an unbiased estimate of its performance.
7. Save the trained model and its weights using the save() function.
8. Register the saved model with a model registry, such as TensorFlow Serving or MLflow, for deployment.



So, let us try to look into the model now. So, basically for the model, now there are two, we are trying to club two steps, model and the registry, we will talk a bit more about it. So, basically when it comes to the model, of course, we are talking about a machine learning or a deep learning model. The very first thing we have to do is we have to define the machine learning model architecture.

It could be as simple as that. So, for example, maybe if you want to define the type of the model first. For example, you may want to define use, make use of neural networks. Then of course, you will have to define how many layers and you know like how many neurons and what would be the input layer, what would be the output layer look like and so on so forth. So, that is how you define first the machine learning model architecture.

You are going to of course, look at what could be the loss function optimizer and the metrics for the evaluation, for a particular problem. Of course, accuracy can be one particular metric that is simply to have F1 score could be another metric. So, you can you know of course, make use of one or the other metric depending upon the specific requirements.

Then of course, you are going to do the training of the model on the training set that we have split before. We are going to evaluate the model's performance on using the evaluate function on the validation set. So, that you know it does not hamper the it does not affect the models and introduce any bias in the model performance, ok. And then of course, you know depending upon what the response we are getting, we are going to we can tweak the model architecture.

For example, reduce the number of layers, increase the number of layers, reduce the number of neurons, increase the number of neurons so on so forth. Of course, we may want to also tweak the hyper parameters. For example, what could be the learning rate? Can we reduce the learning rate? Can we increase the learning rate so on so forth. And of course, data set itself, maybe you want to use the entire data set, you may want to use the another data set, you may want to use multiple data sets.

So, all these are the questions that you will have to answer while tweaking the performance of the model. And then at the end, you will have to re-tweak this 4 and 5 again and again and again until you arrive to a satisfactory performance. Of course, what can we say a satisfactory performance?

So, for example, if we are talking about a four class classification problem here, for our emotionally intelligent virtual agent, a random chance of doing the classification or random chance of identifying the emotion that is there in the audio is 25 percent right, four classes, 25 percent chance.

So, we would like to have you know at least 75 percent to begin with. And then of course, ideally, we want to have as high performance as we can without having you know a lot of

variance in the performance. So, of course, these are certain you will have to look into the model architecture to understand you know what how can it be achieved.

Once we have done the training and the testing, training and the validation of the entire thing, then we may have a separate testing set where we can get an, this is important unbiased estimate of the model's performance. So, then this is what we are going to report to the external stakeholders. Once we have model training and testing done, we may want to save the trained model.

And when we say we want to save the trained model means model and its parameters. Of course, using some save function, we will see a bit in a bit. And it turns out that whatever model that we are saving; we have to register those models with some model registry. So, basically without going into too much detail, the model registration is what? We want to register the model and its metadata along with its performance into some repository.

So, that we can have a better control version control over it. And this is something that is an essentially step sort of you know before doing the deployment of the model. And this is something this register model registry allows you to track the different versions of the model. And of course, you know if required, replace one version over the another depending upon how it is performing in the real deployment. So, let us see how can let us see any skeleton code which is going to do these 8 steps for us in short, ok.

(Refer Slide Time: 43:28)

Skeleton Code



```
# Define the neural network model architecture
model = keras.Sequential([
    keras.layers.Dense(10, activation='relu'),
    keras.layers.Dense(10, activation='relu'),
    keras.layers.Dense(4, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model on the training set
model.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=(x_val, y_val))

# Evaluate the model on the validation set
loss, acc = model.evaluate(x_val, y_val)
print('Validation accuracy: %s' % acc)

# Tweak the model and repeat training and validation until satisfactory performance is achieved

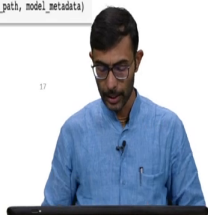
# Test the final model on the testing set
loss, acc = model.evaluate(x_test, y_test)
print('Test accuracy: %s' % acc)

# Save the trained model
model_path = "emotion_detection_model.pkl"
with open(model_path, "wb") as f:
    pickle.dump(model, f)

# Register the model with a model registry
model_version = "v1"
model_registry_uri = "https://my-model-registry.com/api"
model_metadata = {"version": model_version, "accuracy": accuracy}
model_registry_client = ModelRegistryClient(model_registry_uri)
model_registry_client.register_model(model_path, model_metadata)
```

70/20/10

steps 4/5



So, just for the sake of the simplicity, let us say that we defined a neural network model architecture with some you know like layers and with some this ReLU functions as an activation softmax, exact details of course, you can look into the architecture of the neural network.

Once we have defined it, then we are going to you know compile the model by defining what is going to be the loss function, what is going to be the optimizer and what will be the my matrix for the accuracy. You of course, the next is going to be you will be training the model on the training data set.

These are certain parameters that are required for doing the training of the model and then you are also passing the validation data on which you want to tweak the hyper parameters of

the model. You are going to look at the validation evaluation of the model on the validation set.

You may have you may obtain certain accuracy. Once you have obtained the certain accuracy, of course, this is you know the repetition of steps 4 and the 5 as we said before until we are achieving a certain satisfactory performance. Once it is achieved, then of course, the next thing is finally, you are going to report the test accuracy on the test set and then.

So, ok like just to make it clear you know you may want to split your data into three different sets training, testing and validation and usually for example, the simple thumb of rule is 70, 20, 10. So, basically you have 70 percent data for the training, 20 percent data for the testing, 10 percent data for the validation and or you may want to interchange one thing over the other, right.

So, that is how you create three different sets from one given data, ok. So, now, and this is basically you do all to you get an unbiased estimate and the details of it can be understood by taking any machine learning course. So, once we have you know created a particular model. So, basically this is what we are saying that our model is now ready here.

So, you may want to save the model to a certain path. So, basically you know like this is where you are going to dump the model, we are actually saving the model with like the in pickle format. And once you have dumped the model, then maybe you want to register the model with a model registry.

You may use any model registry. So, for example, this is your model registry, URI, there is going to be some metadata that is associated with it such as you know what is the model version, what is the accuracy of the model that you just obtained. And then these are certain other parameters that you require to do the registry of the model onto in on some registry data, right, ok.

So, then roughly this is what the skeleton code will look like for our emotional intelligent virtual assistant. So, and now let us quickly go back. So, now where are we are where we are?

We already completed the prepared data; we already explored the features. So, we already sort of you know clubbed the model building and the registry. Now, we want to look into a bit in the deployment and of course, the monitoring can go further, ok. So, model and registration is done for us ok, deployment.

(Refer Slide Time: 46:34)

Deployment

- Determine that the hardware (e.g. Raspberry Pi) has sufficient processing power, memory, and storage to run the model.
- Choose the TensorFlow Lite framework for deployment, which is compatible with the Raspberry Pi and TensorFlow, the machine learning library used to build the model.
- Optimize the model by converting it to a TensorFlow Lite format and quantizing it to reduce its size and computational requirements.
- Write code to load the optimized model onto the Raspberry Pi and interface with its microphone and speaker.
- Test the deployed model on the Raspberry Pi by running it with different audio inputs and comparing the predicted emotions with the actual emotions.

So, of course, when we are talking about the final deployment, it can be done you know on a particular hardware. Maybe you know like of course, you may you can keep it in a laptop itself, you may want to keep it in a mobile phone itself or you may want to create a dedicated tangible device for example, which you can place on your table or on the user's table.

So, assuming that you know like you decide for a sub-particular hardware, which could be for example, as simple as that Raspberry Pi, you have to understand and take into account that it should have sufficient processing power. We are talking about evaluate running some

machine learning models on it and it should have sufficient memory of course, and storage RAM and the storage to run the model that we are trying to create.

Nevertheless, whatever model we are creating, having identified a particular architecture, we may want to convert the entire model into a light format, which can be optimized and make compatible with the chosen hardware. So, for example, in this case, we can simply make use of the Tensor Lite framework, TensorFlow Lite framework to make it compatible with the Raspberry Pi and TensorFlow that we have been using to develop the machine learning model.

So, having chosen the light TensorFlow Lite framework, next what we want to do? We want to optimize the entire model by converting it into a TensorFlow Lite format and maybe we may have to you know quantize it to reduce its size and computational requirements, depending upon you know whether for example, it is fitting to the hardware, whether it is giving a real time performance and so on so forth. So, these are really tricky steps and you will have to fine tune depending upon certain things.

Then of course, once you have done this thing, once you have converted into a TensorFlow Lite format, you want to load that optimized model onto a raspberry Raspberry Pi. And then you want to you know interface it with its Raspberry's microphone and of course, speaker. Fortunately, Raspberry Pi has both the microphone and the speaker. So, it can work as roughly as a good hardware for you.

So, of course, having loaded it onto the Raspberry Pi, next we may want to deploy the model on the Raspberry Pi. And then we want to test it by you know running different audio inputs and we want to understand whether for example, the output that it is giving, whether it is the output that we wanted to have or comparing it with the ground truth in the real environment. So, just a rough idea about how the development will look like. Of course, please pay attention that we have been focusing solely now on the like on the emotions only.

More or less the similar kind of software development cycle can be followed for the intent which I am not talking about because this is something that Alexa, Siri, Cortana has already been doing. And other thing that we have not talked about is specifically is the context.

So, context is something that also has been integrated to certain to certain with certain proficiency efficiency in the these existing virtual models. But, so, we have not been talking about the intent and the context, but more or less the idea is that the way we are going to understand the emotions.

We are going to understand the intent and the context and we will be referring to the bigger blog there. While generating a appropriate response we will be looking at the emotions, the intent and the context all together. Roughly I wanted to give you a brief of how an end to end deployment will look like.

(Refer Slide Time: 50:09)

Skelton Code



```
# Import necessary libraries
import tensorflow as tf
import numpy as np
import sounddevice as sd

# Check Raspberry Pi processing power, memory, and storage
# Ensure sufficient resources are available to run the model

# Load the trained TensorFlow model
model = tf.keras.models.load_model('emotion_detection_model.h5')

# Convert the model to TensorFlow Lite format and quantize it
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()

# Save the optimized model
with open('emotion_detection_model.tflite', 'wb') as f:
    f.write(tflite_model)

# Load the optimized model onto the Raspberry Pi
interpreter = tf.lite.Interpreter(model_path='emotion_detection_model.tflite')
interpreter.allocate_tensors()

# Define the function to preprocess audio input
def preprocess_audio(audio):

# Define the function to make predictions using the loaded model
def predict_emotion(audio):
    # Preprocess audio input
    audio = preprocess_audio(audio)
    # Get input tensor to audio data
    input_details = interpreter.get_input_details()
    interpreter.set_tensor(input_details[0], audio)
    # Run inference
    interpreter.run()
    # Get output tensor
    output_details = interpreter.get_output_details()
    output = interpreter.get_tensor(output_details[0])
    # Return predicted emotion
    return output

# Define the function to test the deployed model
def test_model():
    # Define a list of emotions
    emotions = ['neutral', 'happy', 'sad', 'angry']
    # Record audio input
    duration = 5 # seconds
    sample_rate = 44100
    audio = sd.rec(int(duration * sample_rate), samplerate=sample_rate, channels=1)
    sd.wait()
    # Predict emotion
    prediction = predict_emotion(audio)
    # Print predicted emotion
    emotion = emotions[int(prediction)]
    print('Predicted emotion:', emotion)
    # Return predicted emotion
    return emotion

# Test the deployed model!
test_model()
```



So, just an skeleton code for this real time life deployment, again I am saying that this is just a skeleton code. So, basically of course, you may want to check the Raspberry Pi that you are using for its power memory and storage. And once you have this thing. So, you may want to optimize the model by using TensorFlow Lite.

So, just simply load the model. After loading the model, you simply convert the model, you are going to convert, save the optimized model with some other name. And then you know of course, then you are going to load this model onto the Raspberry Pi with some interpreter. And then of course, the next is you may want to now run and test it on the new data that is coming, right. Having trained, having model trained already on an existing data.

So, you may want to define a pre-processing audio. So, basically pre-processed audio function you can use from the previous that we just, we already looked at a skeleton code, how can we

pre-process audio, normalization, re-sampling and all those things. Actions, we have a function for that. We can define another function you know to make predictions using the loaded model for a new data. Of course, simply we can, you know like we can collect the audio.

We can do some pre-processing as required. And basically, this is where you know like we are going to load the model, the optimized model and we are going to run the inference on the optimized model. We are going to get the output in a Tensor format and then we are going to you know convert into a some readable format or some interpretable format. And this is the output of the predictive motion function will look like. Once we have the emotion output, similarly you know like we have this predict emotion.

We can have the similar way, we can you know something like we can have another function such as you know identify underscore, intent for example, which we already had for example. Similarly, we can have something like you know identify or add something like context to it. You know these kind of things we can have also. All together you know these three will come together, these three functions will come together. And then what we can do? We can have a prediction of the emotions.

Similarly, we can have a prediction of the identification of the intent, identification of the context. And then accordingly you know rather than just returning the predicted emotion, maybe we want to generate the appropriate response as we saw in the very first skeleton code. And then that is what we are going to return and that is how we are going to simply finally, test the model in the real time, right.

So, that is roughly and in a very very simplistic way the deployment of a model on a particular hardware such as Raspberry Pi. I will be providing you these codes; hopefully you will be able to go through these codes. These are just two examples, just the skeleton codes, please feel free to tweak it as you wish and feel free to play with it.

And hopefully I will be very happy to see if you are able to integrate all these pieces together and come up with real and nice working emotionally intelligent virtual assistant for yourself maybe, ok. So, with that then let us move to the next module.