Course Name: Oracle Cloud Infrastructure AI Foundations

Module 1: Welcome to AI Foundations

1.1 Welcome to AI Foundations

[AUDIO LOGO]
Hello, and welcome to the course introduction to the OCI AI Foundations 2023 course. I'm Hemant Gahankari. I'm a senior principal training lead at Oracle University and will take you through this overview of the course. The course instructor for this course are Hemant Gahankari-- that is myself-- Himanshu Raj, and Nick Commisso. The content for this course has been contributed by Hemant Gahankari, Himanshu Raj, Kiran BR, Rohit Rahi, Anuradha Srinivasaraghavan, and Roshini Varghese.

This course is intended for anyone who wants to begin learning about AI and ML, for example, Cloud engineers or architects, or IT professionals, or even students who want to upskill but have no prior exposure to AI/ML in the past, though familiarity with OCI or similar Cloud Services is recommended.

This course is split into six modules. The first one is AI foundations. In this module, you will learn about foundational AI concepts and you will learn to differentiate between AI, ML, and DL, that is, artificial intelligence, machine learning, and deep learning. In this second module, you will dive deeper into machine learning and learn about supervised, unsupervised, and reinforcement learning.

In the third module, you will learn about a variety of neural network architectures, like artificial neural network, convolutional neural network, recurrent neural network, and long-/short-term memory targeted at solving problems that need using complex data like images, videos, audio, or even text. In the fourth module, we will explore generative AI and LLMs. In this module, you will learn about basics of LLMs, that is, large language models, and transformers. You will also learn about prompt engineering and fine-tuning of LLMs.

In the fifth module, you will get an overview of OCI AI stack, following which, you will learn about OCI ML service data science, OCI AI infrastructure, and also about trustworthy and responsible AI principles. In the sixth module, you will learn about OCI AI services. In this module, you will learn about OCI AI services for a variety of uses. You will get to know about OCI language, vision, speech, document understanding, and anomaly detection.

Before you move on to the next modules, make sure you complete the skill checks that we have provided to test your knowledge and reinforce your understanding of the topics covered. At any point in time, if you have a specific question about the course material or need extra help, fill out our Ask Your Instructor form. Our expert instructors will get back to you as soon as possible with personalized support.

We also want you to get the most out of your learning experience, which is why we have created this community space where you can connect with the other learners and subject matter experts. If you have any questions or want to start a discussion on a particular topic, this is a place to do it. So don't be shy. Join the OU community today and start collaborating with your fellow learners. We can't wait to see what you will bring to the table.

Some of the best practices to maximize the learning from this course are, go to the entire content of the course, including video lectures, demos, and skill checks. After every lecture or a demo, pause and recall, what did you learn? It may be even helpful to make some notes of what you learned and any difficulty you had. Take breaks if necessary while going through this course.

Completing skill checks is important. It will help you solidify your learning. Taking practice exams will help you judge how comfortable you are for taking the actual certification exam. And finally, go through the exam preparation video and take the OCI AI Foundations 2023 exam itself.
We believe in you and know that you have what it takes to succeed. Keep up the excellent work, and I'll see you at the next milestone on your journey towards completing the course. Thanks for watching.

Module2: AI Foundations

2.1 Module Intro

[AUDIO LOGO] Hi, and welcome to this module on foundations. My name is Nick, and I'm a senior cloud engineer here at Oracle. And I'm so grateful that you're here. Now before we dive in, let's take a look at some of the objectives of this module. By the end of the module, you'll be able to explain AI concepts, describe common AI domains and tasks and distinguish between artificial intelligence, machine learning, and deep learning. Now, let's dive in.

2.2 Introduction to AI

[AUDIO LOGO] So what is artificial intelligence? Well, the ability of machines to imitate the cognitive abilities and problem solving capabilities of human intelligence can be classified as artificial intelligence or AI. So what capabilities or abilities are we referring to? Well, let's take a look.

Human intelligence is the intellectual capability of humans that allows us to learn new skills through observation and mental digestion, to think through and understand abstract concepts and apply reasoning, to communicate using a language and understand the nonverbal queues such as facial recognition, tone variation, and body language.

You can handle objections in real time, even in a complex setting. You can plan for short and long-term situations or projects. And, of course, you can create music and art or invent something new like an original idea.

If you can replicate any of these human capabilities in machines, this is artificial general intelligence or AGI. So in other words, AGI can mimic human sensory and motor skills, performance, learning, and intelligence and use these abilities to carry out complicated tasks without human intervention.

When we apply AGI to solve problems with specific and narrow objectives, we call it artificial intelligence or AI. So let's look at some examples.

AI is all around us, and you've probably interacted with AI, even if you didn't realize it. Some examples of AI can be viewing an image or an object and identifying if that is an apple or an orange.

It could be examining an email and classifying it spam or not. It could be writing computer language code or predicting the price of an older car.

So let's get into some more specifics of AI tasks and the nature of related data. Machine learning, deep learning, and data science are all associated with AI, and it can be confusing to distinguish. So let's look at why we need AI and how it's important.

AI is vital in today's world, and with the amount of data that's generated, it far exceeds the human ability to absorb, interpret, and actually make decisions based on that data. That's where AI comes in handy by enhancing the speed and effectiveness of human efforts.

So here are two major reasons why we need AI. Number one, we want to eliminate or reduce the amount of routine tasks, and businesses have a lot of routine tasks that need to be done in large numbers. So things like approving a credit card or a bank loan, processing an insurance claim, recommending products to customers are just some example of routine tasks that can be handled.

And second, we, as humans, need a smart friend who can create stories and poems, designs, create code and music, and have humor, just like us. So let's break it down further to explore some more benefits.

Here are some AI domains and their examples. We have language for language translation; vision, like image classification; speech, like text to speech; product recommendations that can help you cross-sell products; anomaly detection, like detecting fraudulent transactions; learning by reward, like self-driven cars. You have forecasting with weather forecasting. And, of course, generating content like image from text.

So we're going to be exploring some of these further in the next lesson. Thank you.

[AUDIO LOGO]

2.3 AI - Tasks and Data

[AUDIO LOGO] Let's focus on AI tasks and data pertaining to these three AI domains. We have language, audio and speech, and vision. Let's begin with language tasks.

Language-related AI tasks can be text related or generative AI. Text-related tasks use text as input, and the output can vary depending on the task. Some examples include detecting language, extracting entities in a text, or extracting key phrases and so on.

Consider the example of translating text. There's many text translation tools where you simply type or paste your text into a given text box, choose your source and target language, and then click translate.

Now, let's look at the generative AI tasks. They are generative, which means the output text is generated by a model. Some examples are creating text like stories or poems, summarizing a text, answering questions, and so on. Let's take the example of ChatGPT, the most well-known generative chat bot. These bots can create responses from their training on large language models, and they continuously grow through machine learning.

Let's look at how text as data works. Text is inherently sequential, and text consists of sentences. Sentences can have multiple words, and those words need to be converted to numbers for it to be used to train language models. This is called tokenization. Now, the length of sentences can vary, and all the sentences lengths need to be made equal. This is done through padding.

Words can have similarities with other words, and sentences can also be similar to other sentences. The similarity can be measured through similarity or cosine similarity. We need a way to indicate that similar words or sentences may be close by. This is done through representation called embedding.

Now, let's take a look at various language AI models. Language models refer to artificial intelligence models that are specifically designed to understand process and generate natural language. These models have been trained on vast amounts of textual data that can perform various natural language processing or NLP tasks.

The task that needs to be performed decides the type of input and output. The deep learning model architectures that are typically used to train models that perform language tasks are recurrent neural networks, which processes data sequentially and stores hidden states, long short-term memory, which processes data sequentially that can retain the context better through the use of gates, and transformers, which processes data in parallel. It uses the concept of self-attention to better understand the context.

Speech-related AI tasks can be either audio related or generative AI. Speech-related AI tasks use audio or speech as input, and the output can vary depending on the task. For example, speech-to-text conversion or speaker recognition, voice conversion, and so on. Generative AI tasks are generative in nature, so the output audio is generated by a model. For example, you have music composition and speech synthesis.

Audio or speech is digitized as snapshots taken in time. The sample rate is the number of times in a second an audio sample is taken. Most digital audio have a sampling rate of 44.1 kilohertz, which is also the sampling rate for audio CDs. This means that the audio is sampled 44,100 times per second during recording, and when the audio is played, the hardware then reconstructs the sound 44,000 times per second.

The bit depth is the number of bits in each sample or how information-rich of each of those 44,000 pieces of audio is. However, nothing much can be inferred by looking at just one audio sample. Multiple samples need to be correlated to make sense of the data. For example, listening to a song for a fraction of a second, you won't be able to infer much about the song, and you'll probably need to listen to it a little bit longer.

Audio and speech AI models are designed to process and understand audio data, including spoken language. These deep-learning model architectures are used to train models that perform language with tasks-- recurrent neural networks, long short-term memory, transformers, variational autoencoders, waveform models, and Siamese networks. All of the above models take into consideration the sequential nature of audio.

Vision-related tasks could be image related or generative AI. Image-related AI tasks will use an image as an input, and the output depends on the task. Some examples are classifying images, identifying objects in an image, and so on. Facial recognition is one of the most popular image-related tasks that is often used for surveillance and tracking of people in real time, and it's used in a lot of different fields, including security, biometrics, law enforcement, and social media.

For generative AI tasks, the output image is generated by a model. For example, creating an image from a contextual description, generating images of a specific style or a high resolution, and so on. It can create extremely realistic new images and videos by generating original 3D models of an object, machine components, buildings, medication, people, and even more.

Images as data. Images consist of pixels, and pixels can be either grayscale or color. And we can't really make out what an image is just by looking at one pixel.

The task that needs to be performed decides the type of input needed and the output produced. Various architectures have evolved to handle this wide variety of tasks and data. These deep-learning model architectures are typically used to train models that perform vision tasks-- convolutional neural networks, which detects patterns in images; learning hierarchical representations of visual features; YOLO, which is You Only Look Once, processes the image, and detects objects within the image; and then you have generative adversarial networks, which generates real-looking images.

Some other AI tasks. Anomaly detection. This is time-series data, which is required for anomaly detection, and it can be a single or multivariate for fraud detection, machine failure, et cetera. Recommendations. You can recommend products using data of similar products or users. For recommendations, data of similar products or similar users is required.

Forecasting. Time-series data is required for forecasting and can be used for things like weather forecasting and predicting the stock price.

Thank you.

[AUDIO LOGO]

2.4 Demo: AI

[AUDIO LOGO] In this lesson, I'll walk you through a couple of OCI AI services. The first one, Vision AI service. Vision AI service allows us to carry out a few tasks-- image classification, object detection, text detection, and Document AI. Let us explore each one of these.

So let me pick up one local file for the image analysis over here. So, as you can see, the image has been analyzed, and we have got multiple labels here. And this is the confidence score of those labels. So what it means is, for the vegetation, the AI services 99.23% sure that there is a vegetation in this image.

Let's try another image. This time, let's pick up an image of a zebra. So, once again, you can see here that the Zebra has been detected, vegetation, grassland, plant, animal-- all these labels have been assigned with the confidence scores over here.

Let's pick up another image as well. Let's pick up two zebras this time and see what happens. So it recognizes zebra. It also assigned a label of an animal, vegetation, sky, and a mammal. Let's try object detection. Let's pick up one image of a traffic over here.

So in the object detection, we essentially get all these kind of bounding boxes around the objects. And, as you can see here, we have got the labels also and their confidence scores. And as you can see, multiple cars have been detected. Taxi has been detected. Multiple people have been detected into this image. Let's try another one.

So it's a basket of fruit, and you can see here that multiple fruits, including orange, banana, apple have been detected. Even a bowl has been detected over here.

Let's try our text detection. So let's pick up an image of a bus over here. So, as you can see, this bus has a lot of text written on it. So these are all the text that has been written and extracted by the service over here.

It has even detected this number plate M32HOD. It has also detect this number 45 over here, all this text here, text here, and even the text here, and smaller text blocks over here also. Even this has been detected over here.

Let's take another image for our text detection. So I just picked up image of different kind of fonts and I was just curious to see if it is able to extract the text, which is written into different fonts.

So this is how it goes. It basically goes line by line. So Ariel, and the next word you will see is this. Then you will see Ariel Black and the words over here. So like that, it will keep on scanning the whole-- whole image over here.

Now let's try the Document AI also. So one thing about Document AI is Document AI essentially has been moved to a separate service, which is called as document understanding. But from the feature perspective, the features of the Document AI envision and the document understanding service are the same. So let's try one image over here. So I'll upload one receipt over here, and let's see how does it analyze that receipt.

So, as you can see here, this is all the raw text. First, we are able to see there's a lot more to it. So it is essentially from here like starting from receipt to thank you. It has scanned all the words over here. Now, what it has done for the key value-- so what it has done is actually it has picked up the transaction date somewhere over here.

And transaction time also it has picked up. It has picked up the subtotal. It has picked up the tags and the total. These are the standard items. So it has extracted them as a key value pairs also. And then further also line by line, you can see the key value pairs have been extracted over here.

Let's see what kind of tables it has extracted. So if I just click on this drop down, the three tables have been extracted over here. So the first table, as you can see or see over here. Second table is this particular table-- the total. And the third table is the bottom one here-- card, auth code, and other details were like terminal ID and the amount.

So I hope you find this interesting. We will explore OCI AI Language service. As you can see here, we can do two things here. One is the text analysis, and second is text translation. So, in the text analysis, all these pre-trained models, including language detection, text classification, sentiment analysis, and a few others. They are applied to a block of text that we give.

So let's give some text here. Let's say we'll paste some text here, and we'll analyze this text here. And we can see that the first thing it has done is it has detected the language as English. It has classified the paragraph as a science and technology or a computer-related paragraph.

And it has extracted multiple entities. So entities like food and computers and manual instruments, and so on. And they have been tagged as like, say, a product or an event and many other categories. And these are the confidence scores for those entities. And the key phrases for the text also have been extracted over here, say, something like the early computers or simple manual instruments, and so on.

The sentiments also have been detected-- so aspect-based sentiment. So aspect is like one subject in the paragraphs, so something like food or a service and early computers. All the sentiments have been detected, their score, and whether they are negative or positive-- that has been indicated.

And we also have the sentence level sentiments also, which are marked as either negative or neutral in this particular case. And the personal identifiable information has also been identified. So some of the sensitive data like World War II or dates around it, and they have been indicated as potentially could be being sensitive. So that is an indication that the service has given us. That's the-- that is about the text analysis.

Moving on to the text translation. We have a block of text here, and the source language is English here but we can change it to anything depending on the text that we provide. For now, let's keep it as English. And then for the target also, we have a choice of multiple languages here. Let's try the French first.

So if we click translate just in a couple of seconds, the text has been translated and then we can try some other language like Japanese also over here. So the text has been translated into Japanese language also.

And then, if we go to the Overview once again, we also have the capability of training the custom models also over here. If we provide the custom data, we can train our custom models also. Thanks for watching.

2.5 AI vs ML vs DL

[AUDIO LOGO] Hi, and welcome to this module on AI versus ML versus DL as part of OCI's 2023 AI Foundations. Let's understand the difference between Artificial Intelligence, Machine Learning, and Deep Learning with some simple examples. Let's start with artificial intelligence or AI.

Imagine a self-driving car that can make decisions like a human driver, such as navigating traffic or detecting pedestrians and making safe lane changes. AI refers to the broader concept of creating machines or systems that can perform tasks that typically require human intelligence. Next, we have machine learning or ML. Visualize a spam email filter that learns to identify and move spam emails to the spam folder, and that's based on the user's interaction and email content. Now, ML is a subset of AI that focuses on the development of algorithms that enable machines to learn from and make predictions or decisions based on data.

To understand what an algorithm is in the context of machine learning, it refers to a specific set of rules, mathematical equations, or procedures that the machine learning model follows to learn from data and make predictions on. And finally, we have deep learning or DL. Think of an image recognition software that can identify specific objects or animals within images, such as recognizing cats in photos on the internet. DL is a subfield of ML that uses neural networks with many layers, deep neural networks, to learn and make sense of complex patterns in data.

There are several types of machine learning, including supervised learning, unsupervised learning, and reinforcement learning. Supervised learning where the algorithm learns from labeled data, making predictions or classifications. Unsupervised learning is an algorithm that discovers patterns and structures in unlabeled data, such as clustering or dimensionality reduction. And then, you have reinforcement learning, where agents learn to make predictions and decisions by interacting with an environment and receiving rewards or punishments.

Let's understand the supervised machine learning algorithm. Let's take an example of how a credit card company would approve a credit card. Once the application and documents are submitted, a verification is done, followed by a credit score check and another 10 to 15 days for approval. And how is this done? Sometimes, purely manually or by using a rules engine where you can build rules, give new data, get a decision.

The drawbacks are slow. You need skilled people to build and update rules, and the rules keep changing. The good thing is that the businesses had a lot of insight as to how the decisions were made. Can we build rules by looking at the past data? Let's take a look.

We all learn by examples. Past data is nothing but a set of examples. Maybe reviewing past credit card approval history can help. Through a process of training, a model can be built that will have a specific intelligence to do a specific task. The heart of training a model is an algorithm that incrementally updates the model by looking at the data samples one by one.

And once it's built, the model can be used to predict an outcome on a new data. We can train the algorithm with credit card approval history to decide whether to approve a new credit card. And this is what we call supervised machine learning. It's learning from labeled data. And we're going to talk more about this in the next module.

Now, let's understand the unsupervised machine learning algorithm. Data does not have a specific outcome or a label as we know it. And sometimes, we want to discover trends that the data has for potential insights. Similar data can be grouped into clusters. For example, retail marketing and sales, a retail company may collect information like household size, income, location, and occupation so that the suitable clusters could be identified, like a small family or a high spender and so on. And that data can be used for marketing and sales purposes.

Regulating streaming services. A streaming service may collect information like viewing sessions, minutes per session, number of unique shows watched, and so on. That can be used to regulate streaming services. Let's look at another example. We all know that fruits and vegetables have different nutritional elements. But do we know which of those fruits and vegetables are similar nutritionally?

For that, we'll try to cluster fruits and vegetables' nutritional data and try to get some insights into it. This will help us include nutritionally different fruits and vegetables into our daily diets. Exploring patterns and data and grouping similar data into clusters drives unsupervised machine learning. Let's understand the reinforcement learning algorithm.

How do we learn to play a game, say, chess? We'll make a move or a decision, check to see if it's the right move or feedback, and we'll keep the outcomes in your memory for the next step you take, which is learning. Reinforcement learning is a machine learning approach where a computer program learns to make decisions by trying different actions and receiving feedback. It teaches agents how to solve tasks by trial and error.

This approach is used in autonomous car driving and robots as well. Deep learning is all about extracting features and rules from data. Can we identify if an image is a cat or a dog by looking at just one pixel? Can we write rules to identify a cat or a dog in an image? Can the features and rules be extracted from the raw data, in this case, pixels?

Deep learning is really useful in this situation. It's a special kind of machine learning that trains super smart computer networks with lots of layers. And these networks can learn things all by themselves from pictures, like figuring out if a picture is a cat or a dog. Let's explore what neural networks are.

To make tricky things work, we use layers of neurons. Neural networks are like a bunch of connected brain cells stacked together. They're an example of a supervised machine learning algorithm that is perhaps the best understood in the context of functional approximation. Functional approximation involves estimating a hidden function by examining past or currently known data from the specific domain.

Generative AI, a subset of machine learning, creates diverse content like text, audio, images, and more. These models, often powered by neural networks, learn patterns from existing data to craft fresh and creative output. For instance, ChatGPT generates text-based responses by understanding patterns in text data that it's been trained on. Generative AI plays a vital role in various AI tasks requiring content creation and innovation.

Module 3: Machine Learning Foundations

3.1 Module Intro

[AUDIO LOGO] Hi. My name is Hemant, and I'm a senior principal training lead at Oracle University. I welcome you to this module. In this module, we will begin with foundational machine learning concepts. After that, we will dive into supervised learning, more specifically linear regression and logistic regression. Following that, we will also learn about unsupervised learning. And to end with, we will also explore and learn reinforcement learning. So, let's begin.

3.2 Introduction to Machine Learning

[AUDIO LOGO] Hello. In this lesson, we will learn about machine learning. Machine learning is a subset of artificial intelligence that focuses on creating computer systems that can learn and improve from experience without being explicitly programmed. It is powered by algorithms that incorporate intelligence into the machines by automatically learning from the data it consumes. Let us explore an example to understand how it works.

Consider a pup being taught to differentiate between a ball and a book. You would show the pup various types of books and balls. The pup learns about these objects. In due course, the pup would be able to distinguish and identify a book and a ball.

In place of a pup, if we want, a computer, and we need to teach the computer to differentiate between a cat and a dog, then we give the images of cats and dogs to the computer and train the computer to identify a dog and a cat. These input pictures of cats and dogs that are shown to the computer are the data. Data is the information about objects shown to the computer.

Creating computer systems to learn and improve from experience to identify objects instead of being explicitly programmed is machine learning. Machine first learns to identify the objects, and once it succeeds, it can be used to make predictions.

Machine learning is used all around us. Machine learning provides tools to analyze, visualize, and make predictions from data. All of us, knowingly or unknowingly, use machine learning in our day-to-day activities.

For example, machine learning is in the action when we shop online and we are presented with product recommendations based on our preferences and shopping history; when we are notified about the movie recommendations on Netflix based on our previous viewing; when we are warned about a spam email, and increasingly popular self-driving cars, which extensively use machine learning algorithms. Machine learning is applied to make all these possible. Let us talk some more about how it works.

In machine learning, the term model training is used to build an ML model. During the process of model training, a relationship is established between the input and output parameters. Suppose we define the features by x and the output by y. During machine learning model building, the relationship between the x and y is established. This is defined by the equation y is equal to fx, where x is the independent variable which defines the input features, and y is the dependent variable, which can be a label or a quantitative value.

Features-- defining the input features in a machine learning model involves selecting and representing relevant information from the available data that will be used to train the model. And the label-- the output label represents the information or prediction that the model aims to generate based on the input features. The goal is to approximate the mapping function so well that when you have new input data x, you can predict the output variable y for that data. Let us see how to build the ML model to classify cats and dogs.

Dogs and cats have distinguishing features. The body color, texture, eye color, shape of ears, color of nose, and body shape are some of the defining features which can be used to differentiate a cat from a dog. In this case, the input features can be manually extracted and then used to train a machine learning model. Based on these features, the output label is classified as a cat or a dog. Let us see what are the different data types which can be used for representing the input features.

Data can come in many forms, but machine learning models rely broadly on four primary data types. Numeric data, or quantitative data-- any form of measurable data, such as your height or weight, is a numerical data. Numbers can be whole numbers, such as 12, or continuous like 82.5.

Categorical data represents aspects, like gender, ethnicity, and so on. It can be nominal or ordinal. Nominal data does not have any meaningful order, for example color of an eye or gender. Ordinal data has a natural ordering, for example levels of difficulty, like easy, medium, and hard.

Time series data-- a sequence of numbers collected at regular intervals over a period. This makes it easy to compare data from week to week, year to year, or according to any other time-based metric, for example average number of book sales over a year.

Text data-- for example, words or paragraphs about any topic. Depending on the type of data, this might have some effect on the type of algorithm that you can use for modeling or the type of questions you can ask of it. Let us investigate the common flavors of machine learning, namely supervised, unsupervised, and reinforcement learning.

Machine learning is an algorithm-driven approach to create a model. The types of machine learning models depend on whether we have a labeled output or not. There are, in general, three types of machine learning approaches-- supervised, unsupervised, and reinforcement.

Supervised machine learning is generally used to classify data or make predictions, whereas unsupervised learning is generally used to understand relationships within data sets. Reinforcement learning uses algorithms that learn from outcomes to make decisions or choices. Let us talk about some real examples.

Some of the most popular applications of supervised machine learning are disease detection given the patient data, weather forecasting, stock price prediction, spam detection, and credit scoring. For unsupervised machine learning, some of the most common applications are customer segmentation and targeted marketing campaigns. Most popular among reinforcement learning applications are automated robots, autonomous driving cars, and even playing games. We will explore how supervised, unsupervised, and reinforcement machine learning works in the following lessons.

While ML provides a good solution for cases where the problem statement is simple and huge labeled data is available, there are places where ML is not a good solution. Let us take a look at that. ML is not the optimal solution when simpler alternatives are available. If we have simpler ways to solve a problem using rule-based reasoning, then ML may not be required.

ML might not give satisfactory results in cases like self-driving cars, where it may be easy to identify other cars but might be difficult to understand hand gestures. ML solutions are costly. So the basic image and text processing tasks, like resizing, filtering, and counting-- ML is not the good solution.

Traditional ML techniques may struggle to extract relevant features or patterns from complex data, like audio, video, and textual data. While working with large data sets and complex problems, traditional ML approach may not be scalable. Deep learning models can scale to handle large data sets and complex problems. We will explore deep learning later in this course.

3.3 Supervised Learning – Regression

[AUDIO LOGO] In this lesson, our focus will be on supervised machine learning algorithm, called Linear-Regression. Supervised machine learning is a machine learning model which learns from labeled data. The model learns the mapping between the input and the output. Supervised machine learning is generally used to classify data or make predictions. Let us now talk about some typical applications which use supervised learning model.

Spam detection. If a machine learning model is to be developed as a spam detector, the inputs to the model would be email content, including subject, body, and sender information. The output would be whether the email is spam or not spam. Disease detection. Suppose we need to develop a machine learning model for detecting whether a person's tumor is malignant or not. The input to the model would be the person's medical details. The output would be whether the tumor is malignant or not.

These two applications predict a categorical output, which is binary in nature. Sentiment analysis. If a machine learning model must be developed for sentiment analysis, the inputs to the model would be customer reviews for products, tweets, and comments. The output would be positive, negative, or neutral sentiments. This is, again, a categorical output, which is multi-class in nature.

Stock price prediction. If a machine learning model must be built for a stock price prediction, the inputs would be the opening price, closing price, and volume traded. The output would be the price of the stock. Here, the output is continuous and quantitative in nature. Now, let us explore an example to see how the supervised training works.

Let us consider an example of a supervised learning model to identify fruits. The inputs to a supervised learning model is the input, output pair. In the figure, x is the input feature and t is the target. If we want a model to identify fruits, the training inputs would be the features of the fruits, like color, size, and shape, and the output would be the fruit label, t, which can be apple, banana, and so on.

A model is built using these features and the relationship between the features and class labels are established. The training algorithm iteratively trains the models. During training, the model learns the mapping between the input and output. This mapping is the hypothesis. The correctness of the hypothesis is measured, and an error is compared for the training data set. Based on the error, the hypothesis parameters are fine tuned. This loop iteratively continues till the error becomes very small.

The output of the training algorithm is the trained model, which does the prediction. The accuracy of prediction depends on how well the model is trained. The model is trained with a training data set, in this case, various fruits. For instance, if the shape of the object is round and has a depression at the top, and is red in color, then it will be labeled as apple. The generalized steps in supervised machine learning are as follows.

Data access. Labeled data, as per the requirement of the application, is first collected. This data set should represent the problem to be solved. Data preparation. This is the data pre-processing step, where data is cleaned by removing missing values, outliers, scaling features, and encoding categorical values. The data gets prepared for training.

Modeling. In modeling, we split the data as training and testing data. Appropriate machine learning algorithm is selected and applied, as per the requirement. The model is then trained, where the model learns the mapping between the input features and output labels, or values. Model validation. In model validation, it helps to identify potential issues, like overfitting. That is, when the model memorizes the training data, but fails to generalize well. Or underfitting, that is, when the model is too simple and cannot capture the underlying patterns in the data.

Model deployment. Once the model is trained and evaluated, it is deployed and used for prediction. Monitoring and iteration. Iteratively monitor the model performance and ensure the prediction is accurate with unseen data. In supervised machine learning, the output can be either categorical or continuous. When the output is categorical, it is called classification. And if the output is continuous, it is regression.

If we construct a supervised machine learning model as a spam detector, the model is called a classifier. If the classifier has to predict whether it is spam or not, then it is binary classification. If we set up a model for sentiment analysis, it is multi-class classifier. On the other hand, if we must use predictive analysis to predict the housing prices, given the input parameters like area, number of bedrooms, then it is called regression.

Let us start with the simplest type of regression, linear regression. Regression is a supervised machine learning technique which is used to predict continuous values based on input data. In contrast to classification, where the goal is to assign data points to discrete classes or categories, regression aims to estimate a continuous output.

Linear regression is the simplest form of regression, where the relationship between the variables is assumed to be linear, represented as a straight line. It aims to find the best-fitting line that minimizes the sum of squared errors between the predictor value and the actual target values. The variable you want to predict is called the dependent variable. The variable you are using to predict the dependent variable's value is called the independent variable.

Let us explore an example to see how this model works. Linear regression is used to find the straight line, or hyperplane, that best fits a set of points. Let us take a very simple example of predicting a person's weight when given the height. The input feature here is height in centimeters and the output label is weight in kilograms. The input feature is also called independent feature, and the output feature, which is predicted, is called dependent feature.

The input and output is plotted in a scatter plot, and this visualization helps in understanding the relationship. The plot shows that when height increases, the weight also increases. If a model is built with a given data set of height and weight, the model predicts the weight for a given height. This type of model, where the output is a continuous variable, is called a regression type of problem.

For regression line, we fit the simple line model, which passes through most of the data points. As per elementary algebra, the slope is the incremental change in y-axis for every incremental change in x-axis. In this example, that is the change in weight, given the change in height. The y-intercept is the value of y, given at x is equal to 0. Slope is referred to as weights in regression and it determines the influence of the input variables. Y-intercept is called bias and it allows for shifting the line.

Together, they help linear regression find the best-fitting line to predict the output variable based on the input variables. The equation of the model is given by y equal to wx plus b, where w is the slope and b is the y-intercept. There can be many lines which can pass through most of the points. A regression model arrives at an optimum regression line, which passes through most of the data points. And we can arrive at the best-fitting line by computing loss. Let us see what is loss and how is it computed.

For every data point, the difference between the actual and the predicted value is the loss, or error. In the example of predicting weight for a given height, based on the existing data points, we can arrive at multiple lines, which can pass through most of the data points. The model picks up the most suitable regression line having the minimum loss. The loss function is called L2 loss. It is the square of the difference between a single data point and its predicted value.

Basically, loss is a penalty for a bad prediction. If the model's prediction is perfect, the loss is 0. Otherwise, the loss is high. L2 loss is to be minimized during training throughout the entire data set. Let us see how. Training a model is to determine optimal values for all the weights and biases using labeled examples so that the loss is minimized. W is the slope, and b is the y-intercept.

Training is a good way to reduce the model's loss, and it is done iteratively. The input features are used to calculate the output label using the relationship represented by the linear regression model. The predicted outputs are compared with the label, and the losses generated. The loss is used in computing the parameter updates. With the computed parameters, the model prediction function is run again.

To check the correctness of such a linear model, different evaluation metrics are used. Let us see some of the common metrics which can be used for evaluating regression models. Mean absolute error is a very simple metric which calculates the absolute difference between the actual and predicted values.

Mean squared error finds the squared difference between the actual and predicted value. The impact of the error is lesser when root mean squared error is used. R-squared evaluates the performance of the model, but not in terms of loss. It is like a threshold matrix, also called coefficient of determination, or goodness of fit. The r value, computed, lies in the range of 0 to 1, where 1 indicates a perfect fit and 0 indicates a worse fit.

3.4 Supervised Learning-Classification

[AUDIO LOGO]

In this lesson, we will see how supervised classification works with categorical labeled data sets. Cla ssification is a supervised machine learning technique used to categorize or assign data points into p redefined classes or categories based on their features or attributes.

It is used for pattern recognition and predictive modeling. It works with labeled data sets and creates mapping between the input features and the output class labels, and is most suited for binary classifi cation problems. For example, logistic regression. Logistic regression predicts the output of a catego rical dependent variable given the set of independent variables. It gives the output in the form of pro babilities between 0 and 1.

Let us train a simple model of pass or fail among a set of students given the number of hours of stud y. The input feature here is the number of hours of study, which is the independent variable, and the output is a binary classification pass or fail.
But why do we need logistic regression for this classification instead of linear regression? Let us expl ore. Using a threshold value of 0.5, linear regression can be used to discriminate between pass and fail class.
An output of more than 0.5 would be classified as pass and less than 0.5 as fail class. Issues with th e linear regression are-
- linear regression cannot be used to model when there are outliers in the data. And the output need s to be in the range 0 to 1.
In case of outliers, the linear line must be adjusted, and it is meaningless. If a student studies 15 hou rs and passes the exam, the above linear regression line should be adjusted. Fail and pass class sh ould be limited to 0 and 1.
Logistic regression can be used for solving this issue of squashing the limits from 0 to 1. Logistic reg ression helps you by using a sigmoidal function that takes the number of hours someone studied as i nput and gives you the probability of them passing the test as output.
Before we understand the different evaluation metrics for classification, let us first see the building bl ocks of evaluation metrics used in classification. The building blocks of evaluation metrics used in cl assification is also called confusion matrix.
Let us first examine a simple real life situation. Every day before going out, Sanjay checks the weath er app to see if it would rain or not. If the forecast predicts that it will rain and it actually rains, then w e have a true positive.
Contrarily, if the forecast predicts that it will not rain and it actually doesn't rain, then we have a true negative. Contrary to this, if the forecast predicts that it will rain and it actually doesn't rain, then we h ave a false positive. On the other hand, if the forecast predicts that it will not rain and it actually rains , then we have a false negative.
Now let us move on to the evaluation matrix. Based on the four building blocks-
- true positive, true negative, false positive, and false negative, we have three evaluation matrix. Acc uracy is the fraction of correct predictions. It is the ratio of correct predictions to the total number of p redictions.
But for class imbalanced data sets, accuracy alone is not sufficient as we have skewed class proport ions. That means one or more classes in the data set may have significantly fewer examples compar ed to others.
For example, in the case of cancer data sets, we have fewer malignant cases compared to benign c ases. The next evaluation metric, precision, addresses the class imbalance data set issue. Precision is a measure of how accurately the system identifies positive cases.
To calculate precision, we look at the ratio of true positive cases to all positive cases. As an example , a classifier is built to predict if a person has malignant tumor or not. So out of all positive cases, the number of times a classifier can correctly identify the malignancy is regarded as a classifier with goo d precision.
Recall computes the proportion of actual positives which was identified correctly. Recall helps us un derstand how well the system captures the positive cases without missing them. A higher recall valu e indicates that the system is better at identifying the positive cases. F1 score considers both precisi on and recall, combining their strengths to provide a single evaluation score.

3.5 Demo: Basic Machine Learning

[AUDIO LOGO] Hello, and welcome to our demo on a machine-learning use case, where we will classify iris flower into three species as a part of OCI AI Foundation 2023. What you are seeing here is Oracle Cloud Infrastructure Data Science notebook. OCI Data Science is a fully managed platform for teams of data scientists to build, train, deploy, and manage machine-learning models using Python and open-source tools. It uses a Jupyter lab-based environment to experiment and develop models, as you can see here.

The stages involved in this demo are as following. So first is loading data. Next is preprocessing. Third is training a model. Fourth, evaluating the model. And fifth one is making inferences or predictions.

The iris data set includes three iris species with 50 samples each as well as some properties about each flower. Again, one flower species is linearly separable from the other two. When I say linearly separable, that means the different attributes or data points can be separated using a line, or a linear function, or even a flat hyperplane. But the other two are not linearly separable from each other.

Again, the columns in this data set are ID, sepal length in centimeter, the sepal width in centimeter, petal length in centimeter, petal width in centimeter, and the species itself, which is the target. That's what we are trying to classify into. Again, you can check the data set itself on UC Irvine website or on Kaggle itself.

Again, we will be importing various libraries such as Sklearn, Numpy, and Pandas. Again, these libraries are pretty commonly used library for machine-learning applications. Numpy and Pandas are used for data manipulation and analysis. Again, from sklearn, you have train_test_split, which is used to split our data set into training and testing sets. Again, we have a standard scaler, which is used to standardize the features of the data set. Again, logistic regression is a machine-learning model we will be using. And accuracy score will be used to evaluate the accuracy of our model.

So once we have imported these libraries, we are going to load the iris data set from a CSV file named iris.csv using the pandas.read_csv function. The data set contains information about different species of iris flowers.

So you can see here, I have loaded the iris data set. And now you can see the data set has the attributes, which I was talking about, as well as the target. Once we have done that, we will now split the data into x and y as features and targets respectively. We will further split the data into training and test sets. So basically, the attributes, that goes along with the species as the training part. And few of the attributes, along with their target, will remain for the testing set.

So x contains the features of the data set excluding the species column, which we draw up using the draw function method. Again, y contains the target, which is the output variable, which is the species column. Again, we will split the data into training and testing sets using train_test_split, as we mentioned earlier. Test size, we are going to define 0.2. So 0.2 means 20% of the data will be used for testing. And again, a random state will ensure reproducibility, which we will see later on.

Now we will split the data into features, x, and target, y. We will drop the ID and species, since species is not part of features, and ID is nothing but numbering, 1, 2, 3, 4. So this is not an important attribute of any flower. So we're going to drop those two. And again, while we drop these from x, that will become the correct data set or data frame for our features, and just the species will go into y, this target value.

So you can also see that we have x.head, which shows here your features set. And then here, you have y.head, which shows your target set. So the head, if you don't define, by default, it shows only the top five entries in there. But you can always see the complete data set if you have some knowledge about Python coding.

Again, you split the data into training and testing set. So you have x_train and x_test. That means a part of this x goes for train, and part of it goes for test set. And similarly, the corresponding y will be divided into training and test set.

Again, test size is 0.2. That means 20% of the data will go for test set. And again, this is pretty common. Again, random state equals 42-- ensures that the same randomization is used each time you run the code. So basically, you get the same train and test sets across different educations.

We will now create a StandardScaler instance to standardize the features. The fit_transform function, it computes the mean and the standard deviation from the training data, and scales it. And the transform applies the same scaling to the testing data. So this data standardization is important when features of the input data set have large differences between their ranges, or simply when they are measured in different units.

So to understand standardization with an example, say we have human data set with two features-- height in meters and weight in pounds. That range respectively on average from 1 to 2 meters, and 100 to 200 pounds. The weight feature will dominate over the height feature and will have more contribution to the computations just because it has bigger values compared to the height. So to prevent this problem, transforming features to comparable scales using standardization is the solution. And that's what we have done for the iris species as well.

So we have already learned about regression and classification in supervised learning. And now we are going to evaluate the model. So we will use the trained model to predict the species on the scaled testing data.

So accuracy_score will compare the predicted values with actual values and will calculate accuracy. And then we will print the accuracy of the model predictions. So accuracy is one of the metric for evaluating classification models. And again, it is the number of correct predictions divided by the total number of prediction across all classes.

For this example, we get our accuracy as 1.0. We create a new data array with new samples for prediction now. Again, even on the prediction set, we are going to scale the new data using the same scaler as before. And the model will predict the species for the new data, and we will print the prediction.

So this is the new data sample, which I have generated, which I have written. So you can see it also has four columns. So it's a matrix of 3 by 4. So it has three samples with four attributes. So again, when I say three samples with four attributes, so basically each of the column will represent one of the attribute, which will be sepal length, width, petal length, and petal width.

So on the basis of this, and the patterns our model has learned from the training data, it will make predictions on this new scaled data. And to display the predicted classes, we can see here the first sample is being classified as iris setosa. The second sample is being classified as iris virginica. And again, the third one is classified as iris setosa.

3.6 Unsupervised Learning

[AUDIO LOGO] In this lesson, we will learn about unsupervised machine learning. Unsupervised machine learning is a type of machine learning where there are no labeled outputs. The algorithm learns the patterns and relationships in the data and groups similar data items. In unsupervised machine learning, the patterns in the data are explored explicitly without being told what to look.

For example, if you give a set of different-colored LEGO pieces to a child and ask to sort it, it may the LEGO pieces based on any patterns they observe. It could be based on same color or same size or same type. Similarly, in unsupervised learning, we group unlabeled data sets.

One more example could be-- say, imagine you have a basket of various fruits-- say, apples, bananas, and oranges-- and your task is to group these fruits based on their similarities. You observe that some fruits are round and red, while others are elongated and yellow. Without being told explicitly, you decide to group the round and red fruits together as one cluster and the elongated and yellow fruits as another cluster. There you go. You have just performed an unsupervised learning task.

Clustering is a method of grouping data items based on similarities. Within a cluster, the data items are more similar than the items outside the cluster. If some data items do not fall within any cluster, these data items are deemed as outlier points. For example, grapes do have a different shape, size, and color as of apple, pears, and strawberry, and hence, could be an outlier. Having seen what clustering is, let us explore some of its important use cases.

The first use case of unsupervised machine learning is market segmentation. In market segmentation, one example is providing the purchasing details of an online shop to a clustering algorithm. Based on the items purchased and purchasing behavior, the clustering algorithm can identify customers based on the similarity between the products purchased. For example, customers with a particular age group who buy protein diet products can be shown an advertisement of sports-related products.

The second use case is on outlier analysis. One typical example for outlier analysis is to provide credit card purchase data for clustering. Fraudulent transactions can be detected by a bank by using outliers. In some transaction, amounts are too high or recurring. It signifies an outlier.

The third use case is recommendation systems. An example for recommendation systems is to provide users' movie viewing history as input to a clustering algorithm. It clusters users based on the type or rating of movies they have watched. The output helps to provide personalized movie recommendations to users, for example, Netflix. The same applies for music recommendations also. Let us explore the steps involved in unsupervised machine learning. Before that, we need to understand the concept of similarity.

Similarity is how close two data points are to each other and is a value between 0 and 1. Similarity between objects decide which cluster they will fall into, and hence, important for clustering. Now, suppose from a basket of fruits, we need to identify similar fruits.

We can do it, say, based on color. Then apple and cherry are similar to each other based on color. More the similarity between objects, the closer is the value to 1. So for apple and cherry, in this case, similarity would have a value closer to 1. There are various types of similarity measures used in clustering.

Let us look at the steps involved in supervised learning to see how similarity metrics are used. Just like supervised learning, unsupervised learning also follows a workflow. To cluster the data-- for example, the fruits data-- the following steps are followed.

Prepare the data. While preparing the data, the basic pre-processing steps, like removing missing values, normalizing the data, and feature scaling are performed. Create similarity matrix. The choice on which similarity matrix to use depends on the nature of the data and the specific clustering algorithm being used. Some common similarity metrics which are frequently used are Euclidean distance metric, Manhattan distance metric, cosine similarity matrix, and Jaccard similarity matrix.

Run the clustering algorithm. Clustering algorithms use similarity matrix to cluster the data. The different types of clustering algorithms are partition based, hierarchical based, density based, and distribution based. Interpret the results and adjust your clustering.

Checking the quality of your clustering output is iterative and exploratory. Because there is no labeled output, clustering lacks the ground truth that can verify the output. Verify the results against expectations at the cluster level and the example level. Improving the result requires iteratively experimenting with the previous steps to see how they affect the clustering. Thanks for watching.

[AUDIO LOGO]

3.7 Reinforcement Learning

[AUDIO LOGO] In this lesson, we will learn about Reinforcement Learning. Reinforcement learning is like teaching a dog new tricks. You reward it when it does something right, and over time, it learns to perform these actions to get more rewards. More formally, reinforcement learning is a type of Machine Learning that enables an agent to learn from its interaction with the environment, while receiving feedback in the form of rewards or penalties without any labeled data.

Let us look at some examples. Reinforcement learning is more prevalent in our daily lives than we might realize. To name a few. Autonomous vehicles. The development of self-driving cars and autonomous drones rely heavily on reinforcement learning to make real time decisions based on sensor data, traffic conditions, and safety considerations.

Smart home devices. Virtual assistants like Alexa, Google Assistant, and Siri utilize reinforcement learning to improve natural language processing and adapt to individual users speech patterns and preferences. Industrial automation. In manufacturing and production processes, reinforcement learning is applied to optimize the performance of robots and control systems leading to improved efficiency and reduced service.

Gaming and entertainment. Many video games, virtual reality experiences, and interactive entertainment use reinforcement learning to create intelligent and challenging computer-controlled opponents. The AI characters in games learn from player interactions and become more difficult to beat as the game progresses.

Let us take an example to discuss common RL terminology. Let us say we want to train a self-driving car to drive on a road and reach its destination. For this, it would need to learn how to steer the car based on what it sees in front through a camera. In this example, car and its intelligence to steer on the road is called as an agent.

More formally, agent is a learner or decision maker that interacts with the environment, takes actions, and learns from the feedback received. Environment, in this case, is the road and its surroundings with which the car interacts. More formally, environment is the external system with which the agent interacts. It is the world or context in which the agent operates and receives feedback for its actions.

In this example, what we see through a camera in front of a car at a moment is a state. More formally, state is a representation of the current situation or configuration of the environment at a particular time. It contains the necessary information for the agent to make decisions. The actions in this example are to drive left, or right, or keep straight. Formal definition of actions. Actions are a set of possible moves or decisions that the agent can take in a given state.

Actions have an impact on the environment and influence future states. After driving through the road many times, the car learns what action to take when it views a road through the camera. This learning is a policy. Formally, policy is a strategy or mapping that the agent uses to decide which action to take in a given state. It defines the agent's behavior and determines how it selects actions.

Consider the example of training a dog to learn tricks like pick up a ball, roll, sit, and so on. Here the dog is an agent, and the place it receives training is the environment. While training the dog, you provide a positive reward signal if the dog picks it right and a warning or punishment if the dog does not pick up a trick. In due course, the dog gets trained by the positive rewards or negative punishments.

The same tactics are applied to train a machine in the reinforcement learning. For machines, the policy is the brain of our agent. It is a function that tells what actions to take when in a given state. The goal of reinforcement learning algorithm is to find a policy that will yield a lot of rewards for the agent if the agent follows that policy referred to as the optimal policy.

Through a process of learning from experiences and feedback, the agent becomes more proficient at making good decisions and accomplishing tasks. This process continues until eventually we end up with the optimal policy. The optimal policy is learned through training by using algorithms like Deep Q Learning or Q Learning. Let us explore this a little more with another example.

Reinforcement learning can be used in a robotic arm to optimize the process of placing goods in a warehouse. The goal is to teach the robotic arm how to pick up items and place them efficiently and accurately in the desired locations within the warehouse. Let us see how a robotic arm gets trained using reinforcement learning.

The first step is setting the environment. This includes the robotic arm, the warehouse layout, the goods to be placed, and the target locations for each item. The second step is to define state representations. For the robotic arm, the state can include information, such as the position and orientation of the arm, the position of the items to be picked up, and the positions of the target locations.

Next is action space. Define the action space. Action spaces are the possible actions the robotic arm can take in each state. Decide rewards and penalty. The robotic arm should be rewarded when it successfully places an item in the correct location and penalized for dropping items, damaging goods, or failing to place items accurately. In training, the robotic arm starts in a random state and takes actions in the environment.

Initially, it explores different actions randomly and observes the rewards and penalties it receives for each action. As it learns, it starts to prioritize actions that lead to higher rewards and avoids actions that result in penalties. Through multiple training iterations, the robotic arm learns better strategies for picking up and placing items in the warehouse. Thanks for watching.

Module 4: Deep Learning Foundations

4.1 Module Intro

[AUDIO LOGO] Hello, my name is Hemant, and I'm a senior principal training lead at Oracle University. I welcome you to this module. In this module, we will begin with Foundational Concepts in Deep Learning. Following that, we will also discuss Neural Network Architectures, which are suitable for image analysis. And following that, we will also discuss Neural Network Architectures that are suitable for sequence problems. So let us begin.

[AUDIO LOGO]

4.2 Introduction to Deep Learning

[AUDIO LOGO] In this lesson, we will learn about deep learning. So let's begin. Deep learning is a subset of machine learning that focuses on training Artificial Neural Networks to solve a task at hand. Say, for example, image classification. A very important quality of the ANN is that it can process raw data like pixels of an image and extract patterns from it. These patterns are treated as features to predict the outcomes.

Let us say we have a set of handwritten images of digits 0 to 9. As we know, every one writes the digits in a slightly different ways. So how do we train a machine to identify a handwritten digit? For this, we use ANN.

ANN accepts image pixels as inputs, extract patterns like edges and curves and so on, and correlates these patterns to predict an outcome. That is what digit does the image has in this case. In short, given a bunch of pixels, ANN is able to process pixels data, learn an internal representation of the data, and predict outcomes.

We need to specify features while we train machine learning algorithm. With deep learning, features are automatically extracted from the data. Internal representation of features and their combinations is built to predict outcomes by deep learning algorithms. This may not be feasible manually.

Deep learning algorithms can make use of parallel computations. For this, usually data is split into small batches and process parallelly. So these algorithms can process large amount of data in a short time to learn the features and their combinations. This leads to scalability and performance. In short, deep learning complements machine learning algorithms for complex data for which features cannot be described easily.

Let us quickly look at the history of deep learning. Some of the deep learning concepts like artificial neuron, perceptron, and multilayer perceptron existed as early as 1950s. One of the most important concept of using backpropagation for training ANN came in 1980s.

In 1990s, convolutional neural network were also introduced for image analysis task. Starting 2000, GPUs were introduced. And 2010 onwards, GPUs became cheaper and widely available. This fueled the widespread adoption of deep learning uses like computer vision, natural language processing, speech recognition, text translation, and so on.

In 2012, major networks like AlexNet and Deep Network were built. 2016 onward, generative use cases of the deep learning also started to come up. Today, we have widely adopted deep learning for a variety of use cases, including large language models and many other types of generative models.

Deep learning algorithms are targeted at a variety of data and applications. For data, we have images, videos, text, and audio. For images, applications can be image classification, object detection, and so on. For textual data, applications are to translate the text or detect a sentiment of a text. For audio, the applications can be music generation, speech to text, and so on.

Selecting right deep learning algorithm based on the data and application is important. For image task like image classification, object detection, image segmentation, or facial recognition, CNN is a suitable architecture. For text, we have a choice of the latest transformers or LSTM or even RNN. For generative task like text summarization, question answering, transformers is a good choice. For generating images, text to image generation, transformers, GANs, or diffusion models are available choice.

So let us understand what is Artificial Neural Network or ANN. Artificial Neural Networks are inspired by the human brain. They are made up of interconnected nodes called as neurons. Let us get a simplified understanding of how inputs are processed by a neuron.

In ANN, we assign weights to the connection between neurons. Weighted inputs are added up. And if the sum crosses a specified threshold, the neuron is fired. And the outputs of a layer of neuron become an input to an another layer. Let us also understand the building blocks of ANN.

So first, building block is layers. We have input layer, output layer, and multiple hidden layers. The input layer and output layer are mandatory. And the hidden layers are optional. The second unit is neurons. Neurons are computational units, which accept an input and produce an output.

Weights determine the strength of connection between neurons. So the connection could be between input and a neuron, or it could be between a neuron and another neuron. Activation functions work on the weighted sum of inputs to a neuron and produce an output. Additional input to the neuron that allows a certain degree of flexibility is called as a bias.

Now having understood the components of the ANN, let's also take one small example and see how to train ANN to recognize handwritten digits from images. For that, we have to collect a large number of digit images, and we need to train ANN using these images.

Now let us see how do we train ANN using images. So, in this case, the images consist of 28 by 28 pixels which act as input layer. For the output, we have neurons-- 10 neurons which represent digits 0 to 9. And we have multiple hidden layers. So, in this case, we have two hidden layers which are consisting of 16 neurons each.

The hidden layers are responsible for capturing the internal representation of the raw image data. And the output layer is responsible for producing the desired outcomes. So, in this case, the desired outcome is the prediction of whether the digit is 0 or 1 or up to digit 9.

So how do we train this particular ANN? So the first thing we use the backpropagation algorithm. During training, we show an image to the ANN. Let us say it is an image of digit 2. So we expect output neuron for digit 2 to fire. But in real, let us say output neuron of a digit 6 fired.

So what do we do? We know that there is an error. So to correct an error, we adjust the weights of the connection between neurons based on a calculation, which we call as backpropagation algorithm. By showing thousands of images and adjusting the weights iteratively, ANN is able to predict correct outcome for most of the input images. This process of adjusting weights through backpropagation is called as model training. Thanks for listening.

4.3 Deep Learning Models-Sequence Models

[AUDIO LOGO] In this lesson, we will learn about deep learning models that are suitable for processing sequential data. For example, a paragraph of a text. Sequence models are used to solve problems, where the input data is in the form of sequences. The sequences are ordered lists of data points or events.

The goal in sequence models is to find patterns and dependencies within the data and make predictions, classifications, or even generate new sequences. Some common examples of the sequence models are in natural language processing, deep learning models are used for tasks, such as machine translation, sentiment analysis, or text generation. In speech recognition, deep learning models are used to convert a recorded audio into text.

In deep learning models, can generate new music or create original compositions. Even sequences of hand gestures are interpreted by deep learning models for applications like sign language recognition. In fields like finance or weather prediction, time series data is used to predict future values. Let us see some of the deep learning models, which can be used to work with sequence data.

Recurrent Neural Networks, abbreviated as RNNs, are a class of neural network architectures specifically designed to handle sequential data. Unlike traditional feedforward neural network, RNNs have a feedback loop that allows information to persist across different timesteps.

The key features of RNN is their ability to maintain an internal state often referred to as a hidden state or memory, which is updated as the network processes each element in the input sequence. The hidden state is then used as input to the network for the next time step, allowing the model to capture dependencies and patterns in the data that are spread across time. There are different types of RNN architecture based on application.

One of them is one to one. This is like feed forward neural network and is not suited for sequential data. One to many model produces multiple output values for one input value. Music generation or sequence generation are some applications using this architecture.

A many to one model produces one output value after receiving multiple input values. Example is sentiment analysis based on the review. Many to many model produces multiple output values for multiple input values. Examples are machine translation and named entity recognition.

RNN does not perform that well when it comes to capturing long term dependencies. This is due to the vanishing gradients problem, which is overcome by using LSTM model. Long Short-Term memory, abbreviated as LSTM, works by using a specialized memory cell and a gating mechanisms to capture long term dependencies in the sequential data.

The key idea behind LSTM is to selectively remember or forget information over time, enabling the model to maintain relevant information over long sequences, which helps overcome the vanishing gradients problem. Let us see how LSTM works. Let us see the step by step working of LSTM.

At each timestep, the LSTM takes an input vector representing the current data point in the sequence. The LSTM also receives the previous hidden state and cell state. These represent what the LSTM has remembered and forgotten up to the current point in the sequence.

The core of the LSTM lies in its gating mechanisms, which include three gates, the input gate, the forget gate, and the output gate. These gates are like the filters that control the flow of information within the LSTM cell. The input gate decides what new information from the current input should be added to the memory cell.

The forget gate determines what information in the current memory cell should be discarded or forgotten. The output gate regulates how much of the current memory cell should be exposed as the output of the current time step. Using the information from the input gate and forget gate the LSTM updates its cell state. The LSTM then uses the output gate to produce the current hidden state, which becomes the output of the LSTM for the next time step. Thanks for watching.

4.4 Deep Learning Models-CNN

[AUDIO LOGO] Welcome to a new lesson. In this lesson, we will learn about Convolutional Neural Networks, abbreviated as CNN. Let us get an overview of deep learning model architectures. The first one is Feedforward Neural Networks, abbreviated as FNN. This is also called as Multilayer Perceptron as MLP and is the simplest form of neural networks.

Second is CNN, which is Convolutional Neural Network. This can automatically detect and learn local patterns and features in images and videos. The third one is RNN, which is a recurrent neural network. RNNs are designed to handle sequential data, such as time series data or natural language. They have a feedback loop that allows them to maintain hidden states and capture temporal dependencies.

The fourth one is autoencoders. Autoencoders are unsupervised learning models used for feature extraction and dimensionality reduction, and is commonly employed in data compression and anomaly detection. The fifth one is LSTM that is long short term memory. LSTM is a specialized RNN variant designed to handle long term dependencies in sequential data.

The sixth one is GAN, which is Generative Adversarial Network. This is a powerful deep learning model, which is used for generating realistic synthetic data, such as images, audio, and text. And the last one is transformers. Transformers are widely used in natural language processing and have become state of the art models for tasks, like machine translation, text generation, and language understanding.

In this lesson, we will go deeper into the understanding of CNN. So what is a Convolutional Neural Network? CNN is a type of deep learning model specifically designed for processing and analyzing grid like data, such as images and videos. In the ANN, which was seen in the previous lesson, the input image is converted to a single dimensional array and given as an input to the network.

But that does not work well with the image data because image data is inherently two dimensional. CNN works better with two dimensional data. The role of the CNN is to reduce the image into a form, which is easier to process and without losing features, which are critical for getting a good prediction. Let us see how this happens using a CNN architecture.

Let us get an overview of the CNN layers. The first one is input layer. Input layer is followed by feature extraction layers, which is a combination and repetition of multiple feature extraction layers, including convolutional layer with ReLu activation and a pooling layer.

And this is followed by a classification layer. These are the fully connected output layers, where the classification occurs as a output classes. The feature extraction layers play a vital role in Image classification. Let us see what happens in that layer with an example.

In order to understand feature extraction layers better, let us take an analogy. Let us say we have a robot to inspect a house and tell us what type of a house it is. It uses many tools for this purpose. The first tool is a blueprint detector. It scans different parts of the house, like walls, floors, or windows, and looks for specific patterns or features.

The second tool is a pattern highlighter. This tool marks areas detected by the blueprint detector. The next tool is a summarizer. It tries to capture the most significant features of every room. The next tool is house expert, which looks at all the highlighted patterns and features, and tries to understand the house.

The next tool is a guess maker. It assigns probabilities to the different possible house types. And finally, the quality checker randomly checks different parts of the analysis to make sure that the robot doesn't rely too much on any single piece of information. Let us see how this is mapped to the feature extraction layers.

Let us check the analogy between the robot, house inspector, and different layers of the feature extraction. Similar to blueprint detector, we have a convolutional layer. This layer applies convolutional operations to the input image using small filters known as kernels.

Each filter slides across the input image to detect specific features, such as edges, corners, or textures. Similar to pattern highlighter, we have a activation function. The activation function allows the network to learn more complex and non-linear relationships in the data. Pooling layer is similar to room summarizer.

Pooling helps reduce the spatial dimensions of the feature maps generated by the convolutional layers. Similar to house expert, we have a fully connected layer, which is responsible for making final predictions or classifications based on the learned features. Softmax layer converts the output of the last fully connected layers into probability scores.

The class with the highest probability is the predicted class. This is similar to the guess maker. And finally, we have the dropout layer. This layer is a regularization technique used to prevent overfitting in the network. This has the same role as that of a quality checker.

To summarize, the purpose of feature extraction layers is to automatically learn and extract relevant patterns and features from the input images. Convolutional layer applies convolutional operations to the input image using small filters known as kernels. Each filter slides across the input image to detect specific features, such as edges, corners, or textures.

The second one is the activation function, which is applied after each convolutional operation. The activation function allows the network to learn more complex and non-linear relationships in the data. And the third one is pooling. It helps reduce computational complexity and also reduces the spatial dimensions of the feature, maps generated by the convolutional layers.

CNNs do have some limitations that are important to be aware of. Training CNNs on large data sets can be computationally expensive and time consuming. CNNs are susceptible to overfitting, especially when the training data is limited or imbalanced. CNNs are considered black box models making it difficult to interpret.

And CNNs can be sensitive to small changes in the input leading to unstable predictions. One of the most widely used applications of CNNs is image classification. For example, classifying whether an image contains a specific object, say cat or a dog.

CNNs are used for object detection tasks. The goal here is to draw bounding boxes around objects in an image. CNNs can perform pixel level segmentation, where each pixel in the image is labeled to represent different objects or regions. CNNs are employed for face recognition tasks as well, identifying and verifying individuals based on facial features.

CNNs are widely used in medical image analysis, helping with tasks like tumor detection, diagnosis, and classification of various medical conditions. CNNs play an important role in the development of self-driving cars, helping them to recognize and understand the road traffic signs, pedestrians, and other vehicles. And CNNs are applied in analyzing satellite images and remote sensing data for tasks, such as land cover classification and environmental monitoring. Thanks for watching.

Module 5: Generative AI and LLM Foundations

5.1 Module Intro

[AUDIO LOGO] Hello, and welcome to the module for generative AI and large language model foundations. My name is Himanshu, and I'm a data scientist at Oracle. Thank you for enrolling in this course. Before we dive in, let's look at the objectives of this module.

By the end of this module, you will be able to explain generative AI and how it works, describe large language models, elaborate on how attention mechanism works. You'll be also able to describe transformer architecture and how that works, define prompt engineering and related concepts, and you'll be also able to explain the large language model life cycle and role of fine tuning. So let's dive in.

5.2 Introduction to Generative AI

[AUDIO LOGO] Hello and welcome to this module on introduction to generative AI, part of OCI 2023 AI Foundations. So what is generative AI? Generative AI refers to a type of AI that can create new content. It is a subset of deep learning, where the models are trained not to make predictions but rather to generate output on their own.

Again, generative models can create a wide range of outputs, such as images, music, speech, text, or other types of data. Again, think of generative AI as an artist who looks at a lot of paintings and learns the patterns and styles present in them. Once it has learned these patterns, it can generate new paintings that resembles what it learned.

It opens exciting possibilities for creating tasks, automation, and helping humans come up with new ideas. However, it is important to note that generative models are not truly creative or understanding humans as they rely on patterns in the data they were trained on and don't have emotions or personal experiences. Let's explore how generative works through an example.

So generative works by learning the underlying patterns in a given data set, and then using that knowledge to create new data that shares those same patterns. This type of AI gets its name because it's generating new content. To explain this in a simple way, let's imagine you are trying to teach a generative AI model to draw a dog.

You would start by giving it a lot of pictures of dogs to learn from. The AI does not know anything about what a dog looks like. But by looking at these pictures, it starts to figure out common patterns and features, like dogs often have pointy ears, narrow faces, whiskers, et cetera. You can then ask it to draw a new picture of a dog.

The AI will use the patterns it learned to generate a picture that hopefully looks like a dog. But remember, the AI is not copying any of the pictures it has seen before but creating a new image based on the patterns it has learned. This is the basic idea behind generative AI. In practice the process involves a lot of complex maths and computation.

And there are different techniques and architectures that can be used, such as variational autoencoders, VAs, and Generative Adversarial Networks, GANs. Since we understood how does generative work, now let's see how is generative AI different from other AI approaches. So many of the AI models are trained to make predictions.

Generative AI, on the other hand, aims to understand and model the underlying distribution of the data to create new examples that resemble the training data. And we saw that through the earlier example. Instead of focusing solely on decision making, generative AI model aims to understand the structure of data and learn how to generate similar samples. AI approaches like supervised learning or reinforcement learning are typically used for tasks, like classification, recommendation systems, and game playing, where focus is on decision making and optimization.

So here, you basically provide the predictive ML model with data and the labels, and then you ask the predictive ML model to give you output depend on those training data. In case of Gen AI model, you just provide it with some samples, and then ask this model to be creative on that data and create similar samples. Generative AI is widely used in applications, such as image synthesis, text generation, music composition, video generation, and more.

So let's now look at some types of generative AI models. So generative AI models are broadly divided into two categories. One is image based, the other one is text based. So the image based generative AI models usually generates visual content, such as images, artwork, and videos. It learns from large collection of images to understand visual patterns, object representations, and styles.

For the text based, on the other hand, it generates textual content, such as sentences, paragraphs, or entire articles. It again learns from large collection of text data to capture patterns, language structures, and semantic relationship. So let's look at a model of each type. So first, for the image based generative model.

So generative adversarial network, which we call GAN, is widely used for image generation tasks. GANs have shown impressive results in generating high quality images, but their application extend well beyond this. Again, GANs have been used to create original artworks. For example, the artwork you see here was created by a GAN model and sold at an auction for almost $432,000.

Next, let's look at a text based generative model. So transformers and large language models are very good example for a text based generative model. There have been revolutionary in the field of natural language processing and understanding. Again, they are built to understand generate and process human language at a massive scale. These models are typically based on deep learning architectures, such as transformers, and are trained on vast amount of text data to learn language patterns and relationships.

Let's now explore some generative real world use cases. Again, generative AI models have a wide variety of applications across numerous domains. Here are a few examples. Again, we have seen the image generation. For the image generation, generative models like GANs are used to generate realistic images. Again, they can be used for tasks, like creating artwork, synthesizing images of human faces, or transforming sketches into photorealistic images.

For text generation, large language models like GPT 3, which are generative in nature, can create human like text. This has applications in content creation, like writing articles, generating ideas. And again, conversational AI, like chat bots, customer service agents. They are also used in programmings for code generation and debugging, and much more.

For music generation, generative AI models can also be used. They create new pieces of music after being trained on a specific style or collection of tunes. A famous example is OpenAI's musenet. Generative AI is also used in pharmaceutical industry for drug discovery, which can propose novel molecular structures for potential new drugs, significantly speeding up the early stages of drug discovery. Some other use cases are image to image translation and data augmentation.

5.3 Introduction to Large Language Models

[AUDIO LOGO] Hello and welcome to this lesson on Introduction to Large Language Models a part of OCI 2023 AI Foundations. So what are large language models? As mentioned in the last lesson, LLMs are a type of artificial intelligence models built to understand, generate, and process human language at a massive scale. They were primarily designed for sequence to sequence tasks such as machine translation, where an input sequence is transformed into an output sequence. Let's look at an example.

LLMs can be used to translate text from one language to another. For example, an LLM could be used to translate English text into French. To do this job, LLM is trained on a massive data set of text and code which allows it to learn the patterns and relationships that exist between different languages. The LLM translates how are you from English to French, Comment allez-vous.

It can also answer questions like, what is the capital of France? And it would answer the capital of France is Paris. And it will write an essay on a given topic. For example, write an essay on a French Revolution, and it will come up with a response like with a title and introduction. Let us now look at some of the LLM features.

So LLM models are typically based on deep learning architectures such as transformers. They are also trained on vast amount of text data to learn language patterns and relationships, again, with a massive number of parameters usually in order of millions or even billions. LLMs have also the ability to comprehend and understand natural language text at a semantic level. They can grasp context, infer meaning, and identify relationships between words and phrases.

These models have shown remarkable capabilities in natural language understanding and generation, and perform very well in various NLP tasks such as sentiment analysis, question and answering, language translation, summarization, and named entity recognition. Moving on to brief history of large language models.

So the history of large language models is a fascinating journey that showcases the remarkable progress in natural language processing. Again, the field of this language modeling dates back to the 20th century, when researchers first developed probabilistic language models to predict the next word in a sequence based on a context. These early models laid the groundwork for later advancement.

So in 2000, we got neural probabilistic language models. With the resurgence of neural networks, researchers started exploring neural language models. These models used neural networks to capture sequential dependencies and improve the performance of language modeling tasks. The next which came up was Word2Vec and N-grams. So in 2013, two significant natural language processing techniques, Word2Vec and N-grams, were introduced both of which have had a profound impact on the various language related tasks.

In 2017, the groundbreaking paper attention is all you need by Vaswani Itil, introduce the transformer architecture. The transformers introduce the self-attention mechanism allowing the model to capture long range dependencies and contextual information more effectively. This brought significant improvements in machine translation and other NLP tasks. Then bidirectional encoder representation, which we call BERT, it was introduced by Google in 2018.

It's a large bi-directional transformer-based language model. The bidirectional context allowed BERT to understand the full context of a word in both directions leading to substantial improvements in language understanding tasks. And again in 2020, OpenAI released GPT-3 the staggering 175 billion parameters making it the largest language model at the time. GPT-3 showcased remarkable language understanding and generation abilities enabling a wide range of application.

And then in 2022, Google announced a large language model or LLM similar to GPT-3 series, which it called PaLM. Like other LLMs, PaLM is a flexible system that can potentially carry out all sorts of text generation and editing tasks. And in 2023, LLaMa, GPT-4, PaLM2, and even LLaMa 2, all made 2023 the year of AI with companies racing to launch their own generative models. LLaMa 2 is free for research and commercial purposes as opposed to Google's PaLM2 and OpenAI's GPT-4.

Model size and parameters are crucial aspects of large language models and other deep learning models. They significantly impact the models capabilities, performance, and resource requirement. Again, so what is model size? The model size refers to the amount of memory required to store the model's parameter and other data structures. Larger model sizes generally led to better performance as they can capture more complex patterns and representation from the data.

Again, the parameters are the numerical values of the model that change as it learns to minimize the model's error on the given task. In the context of LLMs, parameters refer to the weights and biases of the model's transformer layers. Parameters are usually measured in terms of millions or billions. For example, GPT-3, one of the largest LLMs to date, has 175 billion parameters making it extremely powerful in language understanding and generation.

Tokens represent the individual units into which a piece of text is divided during the processing by the model. In natural language, tokens are usually words, subwords, or characters. Some models have a maximum token limit that they can process. And longer text can may require truncation or splitting. Again, balancing model size, parameters, and token handling is crucial when working with LLMs.

But again, you need to keep in mind that not always the bigger models are always better. Model size and number of tokens should be scaled equally. Again, scaling to the larger data sets is only beneficial when the data is high quality. Expanding data set size by a larger degree is increasingly difficult. And again, larger models may perform better, but focus should be on always the high-quality data. And this is one of the paper you should refer to learn and understand better about why always the bigger models are not always better.

So different benefits of LLMs. Again, large language models can understand and interpret human language more accurately and contextually. They can comprehend complex sentence structures, nuances, and word meanings enabling them to provide more accurate and relevant responses to user queries. This model can generate human-like text that is coherent and contextually appropriate. This capability is valuable for context creation, automated writing, and generating personalized response in applications like chatbots and virtual assistants. They can perform a variety of tasks.

Again, large language models are very versatile and adaptable to various industries. They can be customized to Excel in applications such as language translation, sentiment analysis, code generation, and much more. Again, LLMs can handle multiple languages making them valuable for cross lingual tasks like translation, sentiment analysis, and understanding diverse global content.

Large language models can be again, fine-tuned for a specific task using a minimal amount of domain data. The efficiency of LLMs usually grows with more data and parameters. Again, by extracting insights from text data, large language models can assist business in making informed and data-driven decisions.

They can analyze customer feedback, social media content, and market trends to gain valuable business intelligence. Large language models can serve as valuable tools for researchers in linguistics, psychology, and other fields. Again, researchers can explore language patterns, sentiment analysis, and even simulate human-like conversation interactions for experiments. Let's look at different applications of LLMs.

Here are some of the key uses of large language models. Again, LLM can classify text into predefined categories or sentiments making them valuable for tasks like spam detection, sentiment analysis, and content categorization. LLMs can comprehend questions and provide relevant answers making them suitable for question answering systems and chatbots. LLMs are also excellent at understanding human language, including parsing sentences, identifying relationships between words, and extracting information from text.

Hence, they can generate human-like text making them useful for chatbots, language translation, and text summarization. Also LLMs can generate concise summaries of lengthy text making them valuable for extracting key information from documents. Since we have learned about different applications of LLM, now we will see a demo in our next lesson on playground, which is a large language model based Visual interface.

5.4 LLM and Transformer Model

[AUDIO LOGO] Hello and welcome to this lesson on transformers and large language models, a part of OCI 2023 AI Foundations. Understanding language is difficult for computers and AI systems. The reason being the attributes often have meanings based on context. Consider a sentence, such as Jane threw the frisbee, and her dog fetched it.

In this sentence, there are a few things that relate to each other. Jane is doing the throwing. The dog is doing the fetching. And it refers to the frisbee. Suppose we are looking at the word it in the sentence. As a human, we understand easily that it refers to the frisbee. But for a machine, it can be tricky.

Let's see which model can help address this. The answer is sequence models. Sequence models are used to solve problems, where the input data is in form of sequences that is ordered lists of data points or events. The goal in sequence problems is to find patterns dependencies or relationships within the data and make predictions, classification, or generate new sequences based on that understanding.

Some common example of sequence models includes natural language processing, which we call NLP, tasks such as machine translation, text generation sentiment analysis. Language modeling involve dealing with sequences of words or characters.

Speech recognition. Again, converting audio signals into text, involves working with sequences of phonemes or subword units to recognize spoken words. Music generation. Generating new music involves modeling musical sequences, nodes, and rhythms to create original compositions.

Gesture recognition. Sequences of motion or hand gestures are used to interpret human movements for applications, such as sign language recognition or gesture based interfaces. Time series analysis. In fields such as finance, economics, weather forecasting, and signal processing, time series data is used to predict future values, detect anomalies, and understand patterns in temporal data.

Let us see some of the deep learning models which can be used for sequence data. Recurrent Neural Networks, RNNs, are a class of neural network architectures specifically designed to handle sequential data. Unlike traditional feed forward neural networks, which is unidirectional and only goes from input to output, RNNs have a feedback loop that allows information to persist across different time steps, making them well-suited for tasks involving sequences of data.

The key feature of RNN is their ability to maintain an internal state, often referred to as the hidden state or memory, which is updated as the network processes each element in the sequence. This hidden state is then used as input to the network for the next time step, allowing the model to capture dependencies and patterns in the data that are spread across time.

Sequential models, such as RNNs and LSTM process input data one element at a time, maintaining a hidden state that summarizes the previous elements information. While this works well for short sequences, it becomes problematic when dealing with long sentences or documents. Let's see the steps for the sentence we saw earlier.

The first step, the model only knows about Jane at this point. In the second step, the model knows about Jane and threw but not about Frisbee or and. In the third step, the model knows about Jane, threw, and the but not about frisbee or and. It goes on. As the sequence grows, the model's ability to retain relevant context and dependencies weakens, leading to a phenomena called vanishing gradient.

You can read more about vanishing gradients. Vanishing gradients is a very interesting topic, and you must read about it. Consequently, long range dependencies becomes challenging to capture, limiting the model's understanding of the entire sequence. So what is the solution?

And the solution is transformers. It's like model has a bird's eye view of the entire sentence and can see how all the words relate to each other. This allows it to understand the sentence as a whole instead of just a series of individual words. Transformers with their self-attention mechanism can look at all the words in the sentence at the same time and understand how they relate to each other.

For example, transformer can simultaneously understand the connections between Jane and dog even though they are far apart in the sentence. Let's see how. The answer is attention, which adds context to the text. Attention would notice dog comes after frisbee, which comes after dog, and it comes after fetched.

Transformer does not look at it in isolation. Instead, it also pays attention to all the other words in the sentence at the same time. But considering all these connections, the model can figure out that it likely refers to the frisbee. This paper attention is all you need, was a great advance in the use of attention mechanism, and introduced transformers.

The most famous current models that are emerging in natural language processing tasks consist of dozens of transformers or some of their variants, for example, ChatGPT or Burt. You can see that transformer architecture using stacked self-attention and fully connected layers for both the encoder and decoder on the right. Transformer is a type of deep learning model that was introduced, again, in the paper Attention Is All You Need, in 2017. And it differs significantly from RNN and LSTM models.

Through self-attention, it allows each word in the input sequence to attend to all other words, determining the relevance and importance of each word concerning the entire sequence. It has encoded decoder architecture, where encoded processes the input sequence, whereas the decoder generates the output sequence. We will explore the transformer architecture in lesson titled transformer walkthrough.

5.5 Demo: LLM

[AUDIO LOGO] Welcome to the large-language model demo on Cohere, a part of OCI Foundation 2023. Cohere provides access to advanced large-language models and NLP tools through its API. Again, the Cohere Playground is a visual interface for users to test Cohere's large-language models without writing a single line of code. So here we will use a command model, which is of 50 billion parameters from Cohere. And we will do something very basic to showcase the capability of a generative model.

So in the input prompt, I'm going to write, pretend you are a writer for children books. Please write a short story on a fictional character of dog called Snowy who loves playing in the snow. Again, on the right-hand side, you can see there are various models. So we are going to use command. And for the number of words, this is nothing but the number of words you want to see in the output. So let's increase this to 199 or 200.

And again, randomness is nothing but how much creative you want the model to be. So if you increase the Temperature, that means if you go 0.91, it will increase the randomness, or it will induce more creativity. And if you decrease it, the model will be less creative and will mostly stick to the instructions you have provided.

So let's click Run here in the control panel. And if everything goes well, we will have a short story on Snowy in the output window. Seeing the story being generated, we can see the model can do pretty creative tasks. So it's taking some time.

Now, the complete story has been generated. I'm not going to read the whole story, but it looks like it's a very interesting story on Snowy. So now let's solve a real-world problem. Here we will correct all the grammar error within the transcript. So here are few premade examples. So we will take one of it. So I will take this Correct Errors in the Voice Transcript.

So again, for the input, this is voice-to-text transcription correction. Given a transcribed excerpt with errors, the model responds with the correct version of the excerpt. So you can see we have given a few examples. So you have the incorrect transcription. And you have the correct transcription. So you have a couple of examples here. And then you ask the model to correct the incorrect transcription.

So this is what we will learn, and this is what we call as in-context learning. So again, let's click Run here in the control panel and see the output. OK, so this is so much better than the original text and very easy to understand.

So the incorrect transcription was, "I got charged interest on ly credit card but I paid my pull balance one day due date. I not missed a pavement year yet. Period. Man you reverse the interest charge." And the model has corrected the transcription and came up with something, with, "I got charged interest on my credit card but I paid my full balance on the due date. And I have not missed a payment this year yet. Can you reverse the interest charge?"

So again, this is very better than the original text and very easy to understand. And again, this could have various use cases. Suppose you are very technical, but English is not your first language. So in that case, you can use a large-language model and correct your response to your viewers or for your support services. So I will use this clear command from control panel to clear it.

Now you will see the next demo where we won't transform or generate info but we'll extract information. So this will be a sample use case of question answering. I have provided our model a news piece released from Oracle on its 2023 fourth quarter financial results. So let me copy/paste that. So I copy/paste it.

So again, this is a news article released from Oracle on 2023 fourth quarter financial results. So you'll see this is a pretty good size news bulletin. So now I have provided our large-language model this text. And now I'm asking it to answer, what was the revenue in fiscal year 2023? And what was the short-term deferred revenue on the basis of this article?

You can clearly see here that the Oracle revenue for FY23 was $50 billion. And again, our next question, which was, what was the short-term deferred revenue? You can see the short-term deferred revenue was $9 billion. And we can also see the correct answer in the output pane. That is the revenue in FY23 was $50 billion and short-term deferred revenue was $9 billion. So this is a very good case of a question answer capability from large-language model.

Now we'll see our last demo. And using the same news piece, we will ask this model to summarize it. So here it goes. So this is the same news article. And now we are asking model that you are a text summarizer. Summarize the below in five bullet points. And we'll click again on the Run from the control panel. And let's see what it outputs.

So now you can see that this text summarizer has summarized this article beautifully in five bullet points. But keep in mind, this is still in beta phase. So the answer might not be perfect each time. Maybe the number of bullet points also change. But yeah, this pretty much concludes our demo. We have seen different functionalities in Cohere Playground. So thank you all.

5.6 Transformer Walkthrough

[AUDIO LOGO] Hello and welcome to this lesson on transformer walkthrough, a part of OCI 2023 AI Foundations. The transformer architecture is designed to handle sequential data, such as sentences, in natural language by using a mechanism called self-attention. This mechanism allows the model to weigh the importance of words, or tokens, within a sequence with respect to each other, enabling the model to focus on relevant context, regardless of the distance between tokens. Again, tokens are nothing but words in the sentence.

Transformer architecture eliminates the need for recurrent or convolutional layers that were common in previous sequence modeling architectures, such as RNNs and LSTMs. Let's break it down and explore a simple transformer architecture.

Transformer models have two main parts, the encoder and decoder. Both parts are made up of layers. And each layer has its own attention mechanism. The encoder reads the input sentence, for example, a sentence in English, and uses the attention mechanism to create a new representation of the sentence that captures the meaning of the words and their connections. The decoder then uses this information to generate output sentence, for example, a translation of the input sentence in English into French, again, using its own attention mechanism.

Let's look at some of the transformer model types. So we are primarily looking at three different model types. First one is encoder only. It focuses on encoding input sequences for downstream tasks. BERT is a very good example of this. It's chosen for context-rich representation learning. The second one is encoder decoder. It combines an encoder for input processing and a decoder for sequence generation. A very good example for this is machine translation, chosen for sequence-to-sequence tasks.

The third one is decoder only. It generates sequences autoregressively. GPT is a very good example for this. It is chosen for creative text generation and autoregressive tasks. Each architecture type is chosen based on the specific requirements of the task, whether it involves understanding context, generating sequences, or both.

Let us now look at a simple transformer architecture. The simple transformer architecture involves tokenization, embedding, positional encoding, transformer, and softmax. Our example sentence, which was, "Jane threw the Frisbee and her dog fetched it," this is given as an input and which is then tokenized. So in the tokenization step, tokenization involves breaking down the sentence into smaller pieces called tokens.

Tokens can be short as one character or as long as one word. In this case, these are the words. Here you can see this period is one character and is still a token. The choice of how to break down the sentence depends on the specific tokenizer used.

The next step in this is embedding. So in embedding, each token is converted into a numerical form, which we call a vector, that the model can understand. So you can see each of the words which were tokenized. They have a vector associated, a vector calculated associated with it. And then these vectors are created in a way that captures the meaning of the word.

For instance, words that have similar meanings will have similar vectors. For example, the words like a "dog" and a "canine" could have a very similar vector. After embedding, the next step, positional encoding is done. So in the positional encoding, the model needs to know the order of the words in the sentence because the meaning of a sentence can change depending on the order of its words.

Again, for example, "Jane threw the Frisbee" has a different meaning from "The Frisbee threw Jane." So the model adds information about the position of each word in the sentence to their corresponding vectors. And this is called positional encoding. So for example, you can see here, this is the order of all the words or the characters involved in this sentence. And this is then fed to the transformer block.

So each transformer block is made up of two parts, an attention and a feed forward. The attention part helps the model understand the context of each word. It does this by letting each word look at all the other words in the sentence. The feed forward part applies a specific function, known as the feed forward function, to each word individually. And then finally, the softmax function is applied.

So what softmax does, after processing through the attention and feed forward parts of the transformer blocks, the model generates a list of scores for each word in the vocabulary. The model uses the softmax function to turn these scores into a probability distribution. And these steps are then repeated for each word in the sentence, allowing the model to generate a translation, a summary, an answer to a question, or any other type of language output.

Next, so how does transformer draws on info from previous prompts? So in the attention part, the model realizes that Jane is associated with action of throwing the Frisbee because of the attention it pays to the connection between Jane and threw in the original statement. The model predicts the next word given the context who threw the Frisbee. In this case, because the model has correctly associated Jane with the action of throwing the Frisbee, Jane would be assigned a higher probability. And it would be chosen as the answer to the question.

5.7 Prompt Engineering

[AUDIO LOGO] Hello and welcome to this lesson on prompt engineering, a part of OCI 2023 AI Foundations. Let's explore an example of how LLM used for a bot. So imagine you have a versatile recipe robot named chef bot. Suppose that chef bot is designed to create delicious recipes for any dish you desire.

However, bot needs clear instructions to prepare the perfect meal. For chef bot to make a mouth watering pizza when you ask for it, you engineer the prompt. That involves giving chef bot a specific format for pizza related questions. For example, you prompt bot with, please, make a pepperoni pizza.

Chef bot recognizes the prompt as a request for a pizza recipe, and it knows exactly what to do. This is called prompt engineering. However, if you want chef bot to be an expert in a particular type of cuisine, such as Italian dishes, you fine tune chef bot for Italian cuisine by immersing it in a culinary crash course filled with Italian cookbooks, traditional Italian recipes, and even Italian cooking shows.

During this process, chef bot becomes more specialized in creating authentic Italian recipes, and this option is called fine tuning. LLMs are general purpose models that are pre-trained on large data sets but are often fine tuned to address specific use cases. You will get the best results from LLMs when you combine these two methods. Let's explore that.

So now, when you combine prompt engineering and fine tuning, and you get a culinary wizard in chef bot, a recipe robot that is not only great at understanding specific dish requests but also capable of following a specific dish requests and even mastering the art of cooking in a particular culinary style. When you ask chef bot, please make a spaghetti carbonara, it prompt engineering helps it recognize your request as an Italian pasta dish.

And fine tuning ensures that chef bot prepares a delicious authentic spaghetti carbonara with the perfect blend of ingredients. In summary, prompt engineering helps chef bot understand a specific recipe requests, and fine tuning enhances its expertise in Italian cuisine. Combining these two processes, mix shift bot a versatile and skillful cooking companion for all your culinary adventures.

So now, let's look at prompt and prompt engineering. A prompt is the input or initial text provided to the model to elicit a specific response or behavior. So this is something which you write or ask to a language model. Now, what is prompt engineering? So prompt engineering is the process of designing and formulating specific instructions or queries to interact with a large language model effectively.

So let's now get through some prompt basics. So first of all, text prompts are how users interact with LLM models. LLM models attempt to produce the next series of words that are most likely to follow from the previous text. For example, this is a famous address from Lincoln at Gettysburg.

So the prompt was four score and seven years ago. And you can see the LLM model itself completes this prompt. So prompt instructions. In the context of large language models, such as GPT 3 or Burt, prompts are the input text or questions given to the model to generate responses or perform specific tasks.

The goal of prompt engineering is to ensure that the language model understands the user's intent correctly and provide accurate and relevant responses. The use of prompts with no examples is sometimes referred to as zero shot learning. Successful prompts often rely on the practice of one shot or few shot learning.

This refers to the inclusion of one or more examples of the desired behavior of the model, typically by including input and output pairs. This is not learning in the sense that the model is permanently changed, but rather that the examples better conditions the model to respond as desired for only the current inference. Few shot inference involving more than one example is called in-context learning.

It does not require us to further train or fine tune pre-trained LLMs if you want to perform a specific or new task that the LLM wasn't explicitly trained on. This example shows a movie sentiment classifier can rate how positive or negative a movie review is and therefore the overall rating for a movie.

So here, you can see the task description. These two are the examples. And based on this input, the model will prompt an output. So here are some best practices. First one is try writing the prompt input in multiple ways.

For example, how many Harry Potter books are there could be also prompted as how many books are there in Harry Potter novel series. Again, always describe the task with clear and specific instructions. For example, instead of generate five titles about my book, you can better describe the purpose of the book so the titles would be more relevant.

Hence, the better prompt would be I am writing a book about my traveling experiences to Africa. Generate five titles for my book. The third best practice is to handle edge cases and conditions responses. So for example, create an MCQ on prompt engineering, you can specify if you would like to have a single correct choice or multiple correct choice for your MCQ. The fourth one is give examples for completing the task for a desirable outcome. In context learning is a very good example for this approach.

5.8 LLM Fine Tuning

[AUDIO LOGO] Hello and welcome to this lesson on fine tuning, a part of OCI 2023 AI Foundations. So let's first look at the LLM life cycle. So the life cycle of a Large Language Model, LLM, involves several stages, from its initial pre-training to its deployment and ongoing refinement. Here are the components of a typical Large Language Model life cycle described in sequential order.

So the first of this lifecycle is pre-training. The LLM is initially pre-trained on a large corpus of text data from the internet. During pre-training, the model learns grammar, facts, reasoning abilities, and general language understanding. The model predicts the next word in a sentence given the previous words, which helps it capture relationships between words and the structure of language.

The second phase is fine tuning initialization. After pre-training, the model's weights are initialized, and it's ready for task specific fine tuning. Fine tuning can involve supervised learning on labeled data for specific tasks, such as sentiment analysis, translation, or text generation.

The model is fine tuned on specific tasks using a smaller domain specific data set. The weights from pre-training are updated based on the new data, making the model task aware and specialized. The next phase of the LLM life cycle is prompt engineering. So this phase craft effective prompts to guide the model's behavior in generating specific responses.

Different prompt formulations, instructions, or context can be used to shape the output. The next phase is evaluation and iteration. So models are evaluated using various metrics to access their performance on specific tasks. Iterative refinement involves adjusting model parameters, prompts, and fine tuning strategies to improve results.

So as a part of this step, you also do few shot and one shot inference. If needed, you further fine tune the model with a small number of examples. Basically, few shot or a single example, one shot for new tasks or scenarios. We have already discussed this in prompt engineering lesson.

Also, you do the bias mitigation and consider the ethical concerns. These biases and ethical concerns may arise in models output. You need to implement measures to ensure fairness in inclusivity and responsible use. The next phase in LLM life cycle is deployment.

Once the model has been fine tuned and evaluated, it is deployed for real world applications. Deployed models can perform tasks, such as text generation, translation, summarization, and much more. You also perform monitoring and maintenance in this phase.

So you continuously monitor the model's performance and output to ensure it aligns with desired outcomes. You also periodically update and retrain the model to incorporate new data and to adapt to evolving language patterns. This overall life cycle can also consist of a feedback loop, whether you gather feedbacks from users and incorporate it into the models improvement process.

You use this feedback to further refine prompts, fine tuning, and overall model behavior. RLHF, which is Reinforcement Learning with Human Feedback, is a very good example of this feedback loop. You also research and innovate as a part of this life cycle, where you continue to research and develop new techniques to enhance the model capability and address different challenges associated with it.

Let's see the example of LLM life cycle through the architecture of Llama-2-chat model. So this is a critical diagram for training of Llama-2-chat. This process begins with the pre training of Llama-2 to using publicly available online sources.

Following this, they create an initial version of Llama-2-chat through the application of supervised fine tuning. Subsequently, the model is iteratively refined using Reinforcement Learning with Human Feedback methodologies. Going further in detail would be out of scope for this foundational course. So let's focus back to the crux of this lesson, which is fine tuning.

So as discussed in the lesson titled prompt engineering, the prompt engineering approach offers a more resource efficient alternative to parameter fine tuning. However, its performance typically falls short of fine tuning. Because it does not update the model's parameters for a specific task, which may limit its adaptability to task specific nuances.

Moreover, prompt tuning can be labor intensive, as it often demands human involvement in comparing the quality of different prompts. In the context of large language models, fine tuning refers to the process of further training a pre-trained language model on a specific task or domain to make it more specialized and accurate for that particular use case. Large language models such as Bert GPT 3 are pre-trained on vast amounts of diverse text data to learn the patterns and structures of language.

This initial pre-training allows the model to capture general language understanding and knowledge. However, fine tuning takes this generic model and adapts it to perform a specific task, such as text classification, question answering, or language translation. Pre-trained models are trained on a massive amount of general text data, and fine tuning allows adapting them to perform well on specialized task, making them more versatile and efficient.

Weights from pre-training are updated based on the new data. So why fine tune? Fine tuning is required for task specific adaptation, domain specific vocabulary, efficiency and resource utilization, and for ethical concerns. Let's review each one of them.

So the first one is task specific adaptation. So pre-trained language models are trained on extensive and diverse data sets and have good general language understanding. They excel in language generation and comprehension tasks, though the broad understanding of language may not lead to optimal performance in specific task.

These models are not task specific. So the solution is fine tuning. The fine tuning process customizes the pre-trained models for a specific task by further training on task specific data to adapt the model's knowledge.

Moving on, the second reason is domain specific vocabulary. So pre-trained models might lack knowledge of specific words and phrases essential for certain tasks in fields, such as legal, medical, finance, and technical domains. This can limit their performance when applied to domain specific data.

Fine tuning enables the model to adapt and learn domain specific words and phrases. These words could be, again, from different domains. So the third reason to fine tune is efficiency and resource utilization. So fine tuning is computationally efficient compared to training from scratch.

Fine tuning reuses the knowledge from pre-trained models saving time and resources. Again, fine tuning requires fewer iterations to achieve task specific competence. Shorter training cycles expedite the model development process. And again, it conserves computational resources, such as GPU memory and processing power.

Again, fine tuning is efficient in quicker model deployment. It has faster time to production for real world applications. Fine tuning is, again, a scalable enabling adaptation to various tasks with the same base model, which further reduce resource demands, and it leads to cost saving for research and development.

The fourth reason to fine tune is of ethical concerns. So again, pre-trained models learns from diverse data. And those potentially inherit different biases. So fine tune might not completely eliminate biases. But careful curation of task specific data ensures avoiding biased or harmful vocabulary. Again, the responsible uses of domain specific terms promotes ethical AI applications.

Module 6: OCI AI Portfolio

6.1 Module Intro

[AUDIO LOGO] Hi. My name is Hemant. And I'm a senior principal training lead at Oracle University. I welcome you to this module. To begin with, in this module, we will discuss OCI AI services. Next to that, we will also discuss OCI Machine Learning services. Following that, we will also discuss OCI AI Infrastructure. And we will end with responsible AI principles. So let's begin.

6.2 AI Services Overview

[AUDIO LOGO] In this lesson, I will take you through an overview of OCI AI services. Oracle has been focusing on bringing to the enterprise at every layer of our stack. It all begins with data and infrastructure layers. OCI AI services consume data, and AI services, in turn, are consumed by applications.

This approach involves extensive investment from infrastructure to SaaS applications. Generative AI and massive scale models are the more recent steps. Oracle AI is the portfolio of cloud services for helping organizations use the data. They may have for the business specific uses.

Business applications consume AI and ML services. The foundation of AI services and ML services is data. AI services contain pre-built models for specific uses. Some of the AI services are pre-trained, and some can be Additionally trained by the customer with their own data.

AI services can be consumed by calling the API for the service, passing in the data to be processed, and the service returns a result. There is no infrastructure to be managed for using AI services. In this lesson, the focus is on OCI AI services.

OCI AI services provide multiple methods for access. The most common method is the OCI Console. The OCI Console provides an easy to use, browser based interface that enables access to notebook sessions and all the features of all the data science, as well as AI services.

REST API. The REST API provides access to service functionality but requires programming expertise. And API reference is provided in the product documentation. Language SDKs, OCI also provides programming language SDK for Java, Python, TypeScript, JavaScript, .Net, Go, and Ruby. The command line interface provides both quick access and full functionality without the need for scripting.

Let us see what type of services are available. OCI AI services is a collection of services with pre-built machine learning models that make it easier for developers to build a variety of business applications. The models can also be custom trained for more accurate business results. The different services provided are digital assistant, language, vision, speech, document understanding, anomaly detection. Let us explore each one of these services.

OCI language allows you to perform sophisticated text analysis at scale. Using the pre-trained and custom models, you can process unstructured text to extract insights without data science expertise. Language services are provided by pre-trained models, custom models, and text translation. Pre-trained models include language detection, sentiment analysis, key phrase extraction, text classification, named entity recognition, and personal identifiable information detection.

Custom models can be trained for named entity recognition and text classification with domain specific data sets. In text translation, natural machine translation is used to translate text across numerous languages. Using OCI Vision, you can upload images to detect and classify objects in them.

Pre-trained models and custom models are supported. In image analysis, pre-trained models perform object detection, image classification, and optical character recognition. In Image analysis, custom models can perform custom object detection by detecting the location of custom objects in an image and providing a bounding box.

The custom image classification builds a model to identify objects and scene based features in an image. The OCI speech service is used to convert media files to readable texts that's stored in JSON and SRT format. Speech enables you to easily convert media files containing human speech into highly exact text transcriptions.

Using OCI document understanding, you can upload documents to detect and classify text and objects in them. You can process individual files or batches of documents. In OCR, document understanding can detect and recognize text in a document. In text extraction, document understanding provides the word level and line level text. And the bounding box, coordinates of where the text is found.

In key value extraction, document understanding extracts a predefined list of key value pairs of information from receipts, invoices, passports, and driver IDs. In table extraction, document understanding extracts content in tabular format, maintaining the row and column relationship of cells. In document classification, the document understanding classifies documents into different types.

The OCI Anomaly Detection service is a service that analyzes large volume of multivariate or univariate time series data. The Anomaly Detection service increases the reliability of businesses by monitoring their critical assets and detecting anomalies early with high precision. Anomaly Detection is the identification of rare items, events, or observations in data that differ significantly from the expectation.

The Anomaly Detection service is designed to help with analyzing large amounts of data and identifying the anomalies at the earliest possible time with maximum accuracy. Different sectors, such as utility, oil and gas, transportation, manufacturing, telecommunications, banking, and insurance use anomaly detection service for their day to day activities.

Oracle Digital Assistant is a platform that allows you to create and deploy digital assistants, which are AI driven interfaces that help users accomplish a variety of tasks with natural language conversations. When a user engages with the Digital Assistant, the Digital Assistant evaluates the user input and routes the conversation to and from the appropriate skills.

Digital Assistant greets the user upon access. Upon user requests, list what it can do and provide entry points into the given skills. It routes explicit user requests to the appropriate skills. And it also handles interruptions to flows and disambiguation. It also handles requests to exit the bot. Thanks for listening.

6.3 ML Services Overview

[AUDIO LOGO] Hello and welcome to this lesson on ML services overview as a part of OCI 2023 AI Foundations. As we have already seen, Oracle AI is this portfolio of cloud services for helping organizations take advantage of all data for the next generation of scenarios. The foundations of AI services and ML services layer is data.

Obviously, AI and machine learning work on data and require data. The top layer of this diagram is applications and loosely refers to all the ways AI is consumed. This could be an application a business process or an analytics system. Between the application and data layer, you see two groups here, the AI services on the top and the machine learning services on the bottom.

We have already seen about AI services in the lesson titled AI services overview. In this lesson, we are going to primarily focus on OCI Data Science as a part of machine learning services. So what is Oracle Cloud Infrastructure Data Science? OCI Data Science is the cloud service focused on serving the data scientist throughout the full machine learning life cycle with support for Python and open source.

As you can see by icon, in this graphic, the service has many features, such as model catalog, projects, JupyterLab notebook, model deployment, model training, management, model explanation, open source libraries, and AutoML. There are three core principles of OCI Data Science.

The first one, accelerated. The first principle is about accelerating the work of the individual data scientist. OCI Data Science provides data scientists with open source libraries along with easy access to a range of compute power without having to manage any infrastructure. It also includes Oracle's own library to help streamline many aspects of their work.

The second principle is collaborative. It goes beyond an individual data scientists productivity to enable data science teams to work together. This is done through the sharing of assets, reducing duplicative work, and putting reproducibility and auditability of models for collaboration and risk management. Third is enterprise grade.

That means it's integrated with all the OCI Security and access protocols. The underlying infrastructure is fully managed. The customer does not have to think about provisioning compute and storage. And the service handles all the maintenance, patching, and upgrades so user can focus on solving business problems with data science.

So having seen the core principles, let's drill down a bit more into the specifics of OCI Data Science. So again, what is OCI Data Science? It's a cloud service to rapidly build, train, deploy, and manage machine learning models whom does it serve. It serves data scientists and data science teams throughout the full machine learning life cycle with support for Python and open source.

Where it is being used. Users work in a familiar JupyterLab notebook interface, where they write Python code. And how it is used? So users preserve their models in the model catalog and deploy their models to a managed infrastructure.

Some of the important product terminology of OCI Data Science are projects. The projects are containers that enable data science teams to organize their work. They represent collaborative work spaces for organizing and documenting data science assets, such as notebooks sessions and models.

Note that tenancy can have as many projects as needed without limits. Now, this notebook session is where the data scientists work. Notebook sessions provide a JupyterLab environment with pre-installed open source libraries and the ability to add others. Notebook sessions are interactive coding environment for building and training models.

Notebook sessions run in a managed infrastructure. And the user can select CPU or GPU, the compute shape, and amount of storage without having to do any manual provisioning. The other important feature is Conda environment. It's an open source environment and package management system and was created for Python programs.

It is used in the service to quickly install, run, and update packages and their dependencies. Conda easily creates and switches between environments in your notebooks sessions. The next one is accelerated data science SDK. Oracle's Accelerated Data Science ADS SDK is a Python library that is included as part of data science.

ADS has many functions and objects that automate or simplify the steps in the data science workflow, including connecting to data, exploring, and visualizing data. Training a model with AutoML, evaluating models, and explaining models. In addition, ADS provides a simple interface to access the data science service mode model catalog and other OCI services, including object storage.

The next one is models. You already know what a model is. Models define a mathematical representation of your data and business process. You create models in notebooks, sessions, inside projects. The next terminology is model catalog. The model catalog is a place to store, track, share, and manage models.

The model catalog is a centralized and managed repository of model artifacts. A stored model includes metadata about the provenance of the model, including Git related information and the script. Our notebook used to push the model to the catalog. Models stored in the model catalog can be shared across members of a team, and they can be loaded back into a notebook session.

The next one is model deployments. Model deployments allow you to deploy models stored in the model catalog as HTTP endpoints on managed infrastructure. Deploying machine learning models as web applications, HTTP API endpoints, serving predictions in real time is the most common way to operationalize models. HTTP endpoints or the API endpoints are flexible and can serve requests for the model predictions. The last one, which we are going to see here is jobs. Data science jobs enable you to define and run a repeatable machine learning tasks on fully managed infrastructure.

6.4 AI Infrastructure Overview

[AUDIO LOGO] In this lesson, we will learn about OCI AI Infrastructure. Oracle AI Stack consists of AI services and machine learning services. These services are built using AI infrastructure. In this lesson, our focus will be on OCI AI Infrastructure.

OCI AI Infrastructure is mainly composed of GPU based instances. Instances can be virtual machines or bare metal machines. High performance cluster networking that allows instances to communicate to each other. Super clusters are a massive network of GPU instances with multiple petabytes per second of bandwidth. And a variety of fully managed storage options from a single byte to exabytes without upfront provisioning are also available. Let us explore each of these components further.

Let us understand why do we need GPUs. ML and AI needs lots of repetitive computations to be made on huge amounts of data. Parallel computing on GPUs is designed for many processes at the same time. A GPU is a piece of hardware that is incredibly good in performing computations.

GPU has thousands of lightweight cores, all working on their share of data in parallel. This gives them the ability to crunch through extremely large data set at tremendous speed. Let us see the various GPU instances offered by OCI.

GPU instances are ideally suited for model training and inference. Bare metal and virtual machine compute instances powered by NVIDIA GPUs H100, A100, A10, and V100 are made available by OCI. For large scale AI training, data analytics, and high performance computing, bare metal instances BM 8 X NVIDIA H100 and BM 8 X NVIDIA A100 can be used.

These provide up to nine times faster AI training and 30 times higher acceleration for AI inferencing. The other bare metal and virtual machines are used for small AI training, inference, streaming, gaming, and virtual desktop infrastructure. Oracle offers all the features and is the most cost effective option when compared to its counterparts.

For example, BM GPU 4.8 version 2 instance costs just $4 per hour and is used by many customers. Superclusters are a massive network with multiple petabytes per second of bandwidth. It can scale up to 4,096 OCI bare metal instances with 32,768 GPUs.

We also have a choice of bare metal A100 or H100 GPU instances, and we can select a variety of storage options, like object store, or block store, or even file system. For networking speeds, we can reach 1,600 GB per second with A100 GPUs and 3,200 GB per second with H100 GPUs. With OCI storage, we can select local SSD up to four NVMe drives, block storage up to 32 terabytes per volume, object storage up to 10 terabytes per object, file systems up to eight exabyte per file system, OCI File system, employs five replicated storage located in different fault domains to provide redundancy for resilient data protection.

HPC file systems, such as BeeGFS and many others are also offered. OCI HPC file systems are available on Oracle Cloud Marketplace and make it easy to deploy a variety of high performance file servers. Mosaic ML platform is designed to make it as easy as possible to train and deploy deep learning models. The platform stack begins with a state of the art generative models.

The models can be trained from scratch and come with proven training recipes. Mosaic ML also provides high quality pre-trained checkpoints. You can domain tune these checkpoints on proprietary data to maximize model quality for specific use cases?

For training LMS at scale, OCI SuperCluster provides ultra low latency cluster networking, high performance computing storage, and OCI compute bare metal instances powered by NVIDIA A100 and H100 tensor core GPUs and connect ex-mod NICs. In the next lesson, we will listen to Oracle experts Pradeep Vincent and Jack Brar about their experience in building OCI RDMA superclusters. Thanks for watching.

6.5 GPUs and Superclusters in OCI

[AUDIO LOGO] [MUSIC PLAYING]

Welcome to the First Principles video and blog series where we talk about various architectural aspects behind OCI services. At OCI, we take a lot of pride in delivering the maximum performance possible for the least cost possible to customers.

Remote DMA, or RDMA, as we call it, has been a core technology behind many OCI services pretty much from the beginning. And we actually recently released a video blog talking about various architectural aspects, historical aspects right from the beginning about why we decided to invest in RDMA and the various products that use RDMA and various trade-offs associated with that.

RDMA is essentially, in a nutshell, is a technology that allows for data transfer or network communication that bypasses CPU, goes from one machine to another without any CPU interference. And this allows things like GPUs to communicate at extremely low latency, high bandwidth with low overhead from a CPU perspective.

And this has actually been a foundational building block for our database services, such as ExaCS and Autonomous Database services and as well as for HPC workloads, as well as GPU workloads. And a few years back, we decided to make a strategic bet on Rocky. Rocky stands for RDMA or converged ethernet. So right now, we essentially have an ethernet fabric that essentially enables Rocky. And our HPC workloads, GPU workloads, as well as database workloads all run on-- leverage that and run on top of it.

In the last Cloud World, we announced a strategic partnership with NVIDIA around GPU workloads. And in that context, we've been talking to customers as well as NVIDIA. And it's become very clear that there is a clear demand for very high scale GPU workloads that spans maybe thousands or maybe even tens of thousands of GPUs that can essentially run within a single RDMA network, if you will.

It's essentially like a Supercluster that essentially supports RDMA. And in order to cater that workload, we went back and decided to make some engineering and architectural design enhancements to roll out what we call as RDMA Supercluster that is designed to support very large number of GPUs. To talk about some of the architectural aspects of our Supercluster, we have with as again Jack, who is the lead architect for RDMA networking. Welcome, Jack.

Thank you, Pradeep. I'm here to represent the team.

As always. All right, so tell us a little bit more about the Superclusters.

Yes, so let's take a look at schematic here. This is probably a high level drawing. So what we're showing here is we're showing two GPU nodes at the bottom where each node is comprised of eight NVIDIA A100 class of GPUs that are interconnected using NVIDIA's NVLinks. This is NVLinks.

And the GPU nodes themselves connect to what we call the network fabric. And the way to think about this is network fabric provides a nonblocking interconnect between all the GPUs, meaning the fabric can provide all the benefit that all the GPU nodes need. And in this case, each GPU node connects to the fabric at 1.6 terabits per second or 1,600 gigabits per second. That's like the high level view. Now let's take a look at what the fabric looks like.

So this is like for every GPU that's 200 gigabits per second?

Correct. Each of the GPU gets 200 gigabits of proportional bandwidth because there's a total of 1.6 G.

Right. And it's fully operational. And it's nonblocking in the sense that any GPU in the fabric can talk to any other GPU fabric at a time.

At the same time, yes.

Good observations.

So looking at-- this is the next-level picture of the fabric. So here we look at what the fabric looks like. Now, we use the word Supercluster. By Supercluster what we mean is, it's much larger than an existing cluster network. In particular, we have a concept of blocks. So there's block number one here and block number n here.

Each of the blocks have two tiers of what we call the Clo fabric. Clo is a particular kind of network design. This is a three-tier Clo network. So this network, we plan to scale to tens of thousands of GPUs. And we can envision designs and scale over 100,000 GPUs as well.

Got it. So this is essentially a three-tier Clo network. And so thinking about this, let's say a GPU all the way over here in the block number one wants to talk to the GPU at the very right side of the n, the GPU block, if you will. Well, I can imagine there's going to be additional network latency because of this three-tier Clo design correct in terms of the number of hops. So how do you actually manage that?

Very good point. So you observed correctly. Because this Supercluster is so large, we're talking tens of thousands GPUs. Potentially, we can talk over 100,000 GPUs. That's a lot of GPUs. And just physical constraints, as in power and space, would mean, we have a long what's called a cable distance.

So the worst-case latency in this case, we're talking around 20 microseconds round trip. And as we previously noted in the other blog, the latency within a block is about 6.5 microseconds round trip. So latency has gone up, but still we're talking not very high latency.

Now, in order to account for that higher latency, what we're doing is we have-- because this is QOS-enabled network, as we talked about in the other blog, we have made sure that the switches and the silicon and the chipsets we chose have the right amount of buffering. And we have made sure to allocate the right amount of buffers to the queue that the NLGPU workloads would use. So we have taken care of the higher worst-case latency here.

Got it. So this is not just like a regular three-tier Clo which has been around for a while. It's essentially something that's designed so that we can do lossless networking, if you will, using higher buffers that are specifically tuned across all these devices in order to make this three-tier Clo Supercluster work in conjunction with RDMA.

Very true. You used the right word, the word that I missed. The operating word is lossless. So this is a lossless RDMA network. And lossless networking means the switches don't drop packets. And they have intelligent congestion notification system built into it to avoid packet loss. And all of that still applies. But we just made sure we account for the worst-case higher latency, so the packets are never dropped still.

Cool. So that's great for-- 20 microseconds is still pretty great compared to any other virtualized networking, if you will. Typically, that's like an order of magnitude better than a cross AD virtual network that we might find. But still there are workloads that are extremely sensitive to latency. And they still may want the 6, 7 microsecond latency, How do we deal with that in the Supercluster.

Very good point. Firstly, as you said, the latency, the worst-case latency we talked about is still 10 to 20 times lower than what you would typically experience in a cloud network. But in order to support workloads that need lower latency or that don't really need the scale we're talking about here, imagine you have a small GPU cluster. You don't need tens of thousands of GPUs. You need maybe 1,000 GPUs, or an HPC workload, or an Oracle database workload that maybe doesn't need that larger scale or really cares about low latency.

So what we do for those workloads is we have a control plane that figures out where to deploy a customer's workload. And it takes into account this latency. So if you have a small-enough workload that fits in a cluster, it would only be deployed in what we call a single block. So you would then experience--

Like a HPC or a database cluster workloads essentially go into a single block.

Exactly, or even a smaller workload. It'll go in a single block. And it'll experience the lower latency. So we struck the balance between scale and latency using what we call placement.

Yeah, makes sense. Cool. How about GPU workloads, though? Let's say they actually span multiple of these blocks. Are they essentially going to have to deal with the 20 microsecond latency all the time?

Very good point. And the answer is not all the time. And let's take a look at how we do that. In fact, we believe it's an innovative thing we've done. So as we talked about in the other blog, we have this thing called network locality hints. So we have a service that provides network locality information to the customers and help them build, for example, their GPU topology, their ML topology to take advantage of that.

So let's take a look at that here. So here we're showing a workload that has eight hosts behind four different top of the rack switches in two different blocks. So this is one block. That's another block. Now, if the workload didn't know anything about locality, then any GPU could be talking to any other

GPU at the same time. And you could have relatively higher latency. Still lower numbers but higher than what it needs to be.

So what we've done is we make network locality transparent to the customer. We make that information available to the customer. And what in this case the customer has done is they've interconnected their GPUs in a way where two GPUs on the top of the rack switch, half of their traffic stays local to the top of the rack switch.

For GPUs within the block, something like 85% of the traffic stays local. If you look at all these links, they're local to the block. Just occasionally, some of these links go across blocks. And what this means is most of the time, most of the traffic has really low latency.

Occasionally, some of the traffic has that 20-ish microsecond latency. But the end result is on average, your latency is much lower. In fact, half of the traffic has even less than 6.5 microseconds of latency because they're local to the top of the rack switch. So this is what we call network locality or placement hints.

Got it. And if there are a lot of the traffic is actually constrained within a tour or a block, that also helps reduce flow collisions at many levels of these, the three--

Very good observation. So we got a very nice side benefit, which is customers get really high throughput because this thing called flow collision, which is a common phenomena in all networks where two flows could collide on a single link. The probability of that is much reduced. So you not only get lower latency, you get higher throughput.

Got it. Cool. Well, to summarize, essentially the Supercluster is a three-tier cloud network. But it's a whole bunch of optimizations?

Correct.

The three big things that I think I heard were, one, we have buffers that are tuned for the approximate latency diameter in the network itself so that we can actually preserve the lossless aspect of the Supercluster.

Correct.

And second, we have placement mechanism in our control plane where when you launch GPU nodes or maybe database nodes or whatever, they actually tend to as much as possible be placed within the same block or maybe even within the same tour to reduce the latency as well as to decrease the probability of flow collisions across the network--

That's higher throughput.

--and which results in higher throughput. And then the third, we have even for workloads that actually span multiple blocks or tours. We actually have placement hints that can be passed down to the orchestration mechanism like the GPU workloads where the algorithms can essentially use the locality within the orchestration itself to limit the traffic within tours or maybe blocks as much as possible, reducing the latency as well as reducing the probability of flow collision.

Correct. You said it better than I could have. Thank you.

Cool. Thank you, Jack, appreciate--

My pleasure.

--your time.

6.6 Responsible AI

[AUDIO LOGO] In this lesson, we will learn about Responsible AI. We are increasingly using AI in our day to day work. But can we trust AI? For example, do we trust self-driven cars completely, or do we accept AI driven disease diagnosis without a second opinion? For us to trust AI, it must be driven by ethics that guide us as well.

The guiding principles for AI to be trustworthy are, AI should be lawful, complying with all applicable laws and regulations. AI should be ethical, that is it should ensure adherence to ethical principles and values that we uphold as humans. And AI should be robust, both from a technical and social perspective. Because even with the good intentions, AI systems can cause unintentional harm.

AI systems do not operate in a lawless world. A number of legally binding rules at national and international level apply or are relevant to the development, deployment, and use of AI systems today. The law not only prohibits certain actions but also enables others, like protecting rights of minorities or protecting environment. Besides horizontally applicable rules, various domain specific rules exist that apply to particular AI applications. For instance, the medical device regulation in the health sector.

So let us discuss human ethics and fundamental rights. Human dignity encompasses the idea that every human being possesses an intrinsic worth, which should never be diminished or compromised. AI systems should therefore be developed in a manner that respects and protects humans physical and mental integrity.

Freedom of the individual means a commitment to protect the freedom of expression and the right to private life and privacy. AI system should serve to maintain and foster Democratic processes and respect the choices of individuals. AI systems must not undermine Democratic processes or Democratic Voting Systems.

In AI context, equality entails that the system's operations cannot generate unfairly biased outputs . And while we adopt AI, citizens right should also be protected. So how are AI ethics derived from these? There are three main principles.

AI should be used to help humans and allow for oversight. It should never cause physical or social harm. Decisions taken by AI should be transparent and fair, and also should be explainable. AI that follows the ethical principles is responsive AI.

So if we map the ethical principles to responsible requirements, these will be like, AI systems should follow human centric design principles and leave meaningful opportunity for human choice. This means securing human oversight. AI systems and environments in which they operate must be safe and secure, they must be technically robust, and should not be open to malicious use.

The development, and deployment, and use of AI systems must be fair, ensuring equal and just distribution of both benefits and costs. AI should be free from unfair bias and discrimination. Decisions taken by AI to the extent possible should be explainable to those directly and indirectly affected. Let us quickly see what does typical responsible AI implementation process looks like.

First, a governance needs to be put in place. Second, develop a set of policies and procedures to be followed. And once implemented, ensure compliance by regular monitoring and evaluation. Typical roles that are involved in the implementation cycles are developers, deployers, and end users of the AI.

One of the key challenges in using AI in health care is ensuring its fairness and lack of bias. AI systems are only as good as the data that they are trained on. If that data is predominantly from one gender or racial group, the AI systems might not perform as well on data from other groups.

Another challenge in using the AI in health care is ensuring that it's transparent and explainable. AI systems often make decisions based on complex algorithms that are difficult for humans to understand. As a result, patients and health care providers can have difficulty trusting the decisions made by the AI. AI systems must be regularly evaluated to ensure that they are performing as intended and not causing harm to patients. Thanks for watching.

Module 7: OCI AI Services

7.1 Module Intro

[AUDIO LOGO] Now, before we dive in, let's take a look at some of the objectives of this module. By the end of the module, you'll be able to explain OCI Language service, describe OCI Speech service concepts, describe how OCI Vision service works, explain OCI Document Understanding service, and finally, describe OCI Anomaly Detection service concepts. We got a lot to cover, so let's dig in.

7.2 Language Intro

[AUDIO LOGO] OCI Language analyzes unstructured text for you. It provides models trained on industry data to perform language analysis with no data science experience needed. It has five main capabilities. First, it detects the language of the text. It recognizes 75 languages, from Afrikaans to Welsh.

It identifies entities, things like names, places, dates emails, currency, organizations, phone numbers-- 14 types in all. It identifies the sentiment of the text, and not just one sentiment for the entire block of text, but the different sentiments for different aspects. So let's say you read a restaurant review that said, the food was great, but the service sucked. You'll get food with a positive sentiment and service with a negative sentiment. And it also analyzes the sentiment for every sentence.

It identifies key phrases in the text that represent the important ideas or subjects. And it classifies the general topic of the text from a list of 600 categories and subcategories.

7.3 Demo: Language

[AUDIO LOGO] Let's look at OCI Language in the console. Now, this is the console home screen. Yours might look the same or different, just depending on a number of factors. But what you need to do is go to the menu Analytics & AI. Under AI Services, we have Language.

And we're going to go in here, and now we see the Language page. And there's some good resources here that you might be able to take advantage of. One, if you're interested in particular topics, you can use the links here to take you right into the documentation. Also some resources listed over here, like the API reference, SDK guide and such other listing of blogs related to OCI Language service.

But what we want to do right now is just actually try out the service. So let's go into Text Analytics. And you can see there's some default text already in here that we can analyze, and all you have to do is analyze it. Just click the Analyze button.

And we see the different kinds of analysis that the language service produces. So first we see the language detection, which is English, and the confidence indicator here-- very high. The classification of the text is listed as science and technology, with a subcategory of Earth sciences, and a fairly low probability on that one.

But you can see it's actually accurate in this case. We see the text with the named entities highlighted. So it highlights both the entity, the type of entity, and then the confidence. And so we see different color codings for different types of entities, from locations, quantities, products, date times.

And then the next area is the key phrase extraction. So in this case, it's just listing all of the key phrases that were extracted from the text. And then for sentiment, we get three different types of sentiment. So we get a document-level sentiment, which refers to the whole text and indicates as mixed, and then we get the aspect-based sentiment, which are specific aspects that it is identified in the text, which then it assigns a sentiment to those. And in all cases, these were negative sentiments.

And then it also provides a sentence-level sentiment. So for each sentence in the provided text, it assigns either a positive, neutral, or negative sentiment. And you can see we've got some of each in this example. So now you get an idea of what OCI Language provides in terms of the analysis.

7.4 Speech Intro

[AUDIO LOGO] Now, let's look at OCI Speech. OCI Speech is very straightforward, that it locks the data in audio tracks by converting speech to text. Developers can use Oracle's time-tested acoustic language models to provide highly accurate transcription for audio or video files across multiple languages. OCI Speech automatically transcribes audio and video files into text using advanced deep learning techniques. There's no data science experience required. It processes data directly in object storage. And it generates timestamped, grammatically accurate transcriptions.

OCI Speech supports multiple languages, specifically English, Spanish, and Portuguese, with more coming in the future. It has batching support where multiple files can be submitted with a single call. It has blazing fast processing. It can transcribe hours of audio in less than 10 minutes. It does this by chunking up your audio into smaller segments, and transcribing each segment, and then joining them all back together into a single file. It provides a confidence score, both per word and per transcription. It punctuates transcriptions to make the text more readable and to allow downstream systems to process the text with less friction.

And it has SRT file support. SRT is the most popular closed caption output file format. And with this SRT support, users can add closed captions to their video. OCI Speech makes transcribed text more readable to resemble how humans write. This is called normalization. And the service will normalize things like addresses, times, numbers, URLs, and more. And you can see in the example here where you have a literally transcribed text. But on the right is the normalized text where the number is changed from words to numeric symbols.

It also does profanity filtering, where it can either remove, mask, or tag profanity and output text, where removing replaces the word with asterisks, and masking does the same thing, but it retains the first letter, and tagging will leave the word in place, but it provides tagging in the output data.

[AUDIO LOGO]

7.5 Demo: Speech

[AUDIO LOGO] Let's take a look at OCI Speech in the console. We'll use the menu to go to Analytics and AI, then under AI Services, we'll select Speech, and that takes us to the console page for Speech service.

Now, before we jump into using the service, one of the requirements is that we have some data already loaded into object storage, and we're going to have to create a transcription job to then convert the speech to text. Now, I'm selected a particular compartment here, where I know I have data already available.

So we'll start, and we'll create our transcription job. Let's call our job Training. I've already got the compartment is preselected.

I want to pick the right bucket here, the object storage bucket in my compartment where I know I have the data. And now, it shows me the data I have that's in that bucket, and I see a WAV file here. That's the one that I previously loaded, and we'll run the job.

Now, the job only takes a few seconds to run. I know, in this case, because I've run it several times before, and it's still in progress. Now, the job succeeded, and we can go down, select the file, and we see the transcription result.

So it says, I'm having an issue with my wireless headphones. I think my Bluetooth connectivity to the TV is not working. I have high quality headphones. I know the issue is not with them.

And then we see a second voice come in here. Hi, thanks for contacting support. I'm sorry to hear about your Bluetooth issue.

Now, notice the punctuation that has been added to here. We also see some normalization where, instead of spelling out the words 100 and percent, it's showing them symbolically. So we get a very nice transcription.

7.6 Vision Intro

[AUDIO LOGO] Next is OCI Vision. Vision is a computed vision service that works on images, and it provides two main capabilities-- image analysis and document AI. Image analysis analyzes photographic images. Object detection is the feature that detects objects inside an image using a bounding box and assigning a label to each object with an accuracy percentage. Object detection also locates and extracts text that appears in the scene, like on a sign.

Image classification will assign classification labels to the image by identifying the major features in the scene. One of the most powerful capabilities of image analysis is that, in addition to pretrained models, users can retrain the models with their own unique data to fit their specific needs.

7.7 Document Understanding

[AUDIO LOGO] A second major capability of Vision is called document AI, and it's used for working with document images. You can use it to understand PDFs or document image types, like JPEG, PNG, and Tiff, or photographs containing textual information. The features of document AI are text recognition, also known as OCR or optical character recognition.

And this extracts text from images, including non-trivial scenarios, like handwritten texts, plus tilted, shaded, or rotated documents. Document classification classifies documents into 10 different types based on visual appearance, high level features, and extracted keywords. This is useful when you need to process a document, based on its classification, like an invoice, a receipt, or a resume.

Language detection analyzes the visual features of text to determine the language rather than relying on the text itself. Table extraction identifies tables in docs and extracts their content in tabular form. Key value extraction finds values for 13 common fields and line items in receipts, things like merchant name and transaction date.

7.8 Demo: Vision Demo

[AUDIO LOGO] Let's take a look at OCI Vision in the console. Here we are at the console homepage, and we'll navigate to Analytics and AI and to Vision. Now on the Vision homepage, you can see there's a number of resources here that are available to you. And then you can exercise the different features of the service, both the image analysis through image classification or object detection or the Document AI.

There's also a section there for creating custom models. We're not going to do that today. Let's just go right into image classification. Now the service already has some default images in here, that it will analyze. And you can quickly see what kind of tags it assigns for image classification purposes to this particular image-- overhead power line, transmission tower, plant, sky, line.

OK, that looks like pretty good results here. Let's bring in another image. I'm going to drag one in from my desktop here. Now this is an image of iconic London. You can see some famous landmarks here. What we get tagged in this image-- skyscraper, water, building, bridge, boat. All good tags for this kind of a scene.

All right, let's go into object detection. Same thing applies here. There's some default images that it will do object detection on immediately. So in this scene, now it's looking for specific objects in the scene. It puts a bounding box around those and then tags those.

And this is the same results you get in the actual data that comes back. And over here we see what those results are. But front center, we've got a car, all right? In the back, we've got a bus. Over here on the sidewalk, we've got some people. Over here is another car.

And even just a little fraction of a car is visible, but it still detects it. All right, those are the objects that were detected, but there's also some text in this scene that is detected. And that's highlighted now. It reads this license plate, reads the Oracle logo on this car.

It gets the bus route that's shown on there. I can't even read that in the image, but the object detection does. Also, it picks up the advertisement on this cab. Even way back there, it's picking up a route on that bus. So lots of information extracted from this image.

Now, let's take a look at another scene. Here's a small image, fairly low resolution. And you might not even notice it initially, but there's a person standing on this roof, which is a little unusual, but it picks out that person. It does not, however, do any object detection around these unusual circular objects, which happen to be microwave transmission antennas for communications purposes.

And the model is not trained to detect those. You could then build a custom model to detect those, if you'd like. But it's the person that it picks out in this particular image. Let's look at one more. Now here's a scene of some cyclists out on the road.

And this one's interesting because it detects the person, even though the person is hunched over end facing away from us. And it also detects that they're on a bicycle. And you can tell that because the boxes are nested, .

And then it even picks up a bicycle wheel associated with a bicycle. And same over here, a little further away, still detects a person and this person being on a bicycle. No text in this scene for it to recognize other than what it thinks is a dash on the person's clothing.

7.9 Demo: Document Understanding

[MUSIC PLAYING] All right, let's go to Document AI. Now, the default image here is a receipt, and it detects it as English. And it extracts a lot of raw text off of this receipt. You can see everything that's highlighted here, which is the entire contents of the receipt, have been extracted. And we see what it is extracted here, both in line format or in individual word format.

Now, let's look and see if it got any key values, and yes, it did. Remember, that for receipts, it has a number of specific keys that it looks for, and then if it finds a matching value, it will assign that value to the key.

So we get the merchant name, example cafe, the merchant address, merchant phone number, transaction time, transaction date, a lot of information on this receipt that's useful in processing it as an expense. And then we also see the line item data down here, and we can also see tabular data. It's picked out both two items here, an Americano and a water.

Let's look at a different image example for Document AI. This is an invoice, and now, we see all the highlighted items of text that it has extracted, which is everything that's on the image. One thing that's interesting about this invoice, it's had a stamp put on it after it was received, which the accounts payable clerk used to process that invoice.

And we do extract some information off there, even though some of that is handwritten, and some of it's stamped. And the stamp is not very-- at least didn't come through very readable in the scan of that image. We see all the relevant information extracted.

Because this is an invoice, we're not doing key value extraction on it at this time, but we do pull some tabular data out here. So not only the quantities and descriptions and unit price in total, it also pulled, actually, some of the stamp information off and put it into the description column. But it didn't interfere with the other information that was extracted.

7.10 Anomaly Detection Intro

[AUDIO LOGO] Oracle Cloud Infrastructure anomaly detection identifies anomalies in time series data. Equipment sensors generate time series data, but all kinds of business metrics are also time-based. The unique feature of this service is that it finds anomalies, not just in a single signal, but across many signals at once. That's important because machines often generate multiple signals at once and the signals are often related.

Think of a pump that has an output pressure, a flow rate, an RPM, and an electrical current draw. When a pump's going to fail, anomalies may appear across several of those signals but at different times. OCI anomaly detection helps you to identify anomalies in a multivariate data set by taking advantage of the interrelationship among signals.

The service contains algorithms for both multi-signal, as in multivariate, single signal, as in univariate anomaly detection, and it automatically determines which algorithm to use based on the training data provided. The multivariate algorithm is called MSET-2, which stands for Multivariate State Estimation technique, and it's unique to Oracle. The 2 in the name refers to the patented enhancements by Oracle labs that automatically identify and fix data quality issues resulting in fewer false alarms and more accurate results.

Now take a look at the diagram here, which illustrates the difference between traditional threshold monitoring and multivariate monitoring. You see the sensor for signal at the top of the diagram and it's being monitored as a single signal with a threshold trip. So when the value of the signal exceeds the threshold, then it trips an alarm and notifications happen.

Now in multivariate monitoring as you see down at the bottom, there's six sensors. And it's monitoring all of these in parallel. And those little red triangles are the alarms that are, or the anomalies, that are being detected in those signals by the MSET-2 algorithm.

And the difference between the first anomaly detected and when the threshold trip would occur on SENSOR 4, you can see there's a gap there. And that early warning time difference can be anywhere from days to weeks to months just depending on the data itself.

Now unlike some of the other AI services, OCI anomaly detection is always trained on the customer's data. It's trained using actual historical data with no anomalies, and there can be as many different trained models as needed for different sets of signals.

One of the most obvious applications of this service is for predictive maintenance. Early warning of a problem provides the opportunity to deploy maintenance resources and schedule downtime to minimize disruption to the business.

This table provides a representation of the data expected by OCI anomaly detection. Each row is associated with a timestamp and each signal is a number. Only numerical data is accepted except for the word null. The data has a header row and a timestamp column. Timestamps must follow the ISO 8601 format like shown in the slide.

Each model is trained on a set of related signals. and you can train as many models as you need. The data file itself can be formatted as either CSV or JSON.

Now it's a simple four-step process to prepare a model that can be used for anomaly detection. The first step is to obtain training data from the system to be monitored. The data must contain no anomalies and should cover the normal range of values that would be experienced in a full business cycle.

Second, the training data file is uploaded to an object storage bucket. Third, a data set is created for the training data. So a data set in this context is an object in the OCI anomaly detection service to manage data used for training and testing models.

And fourth, the model is trained. A wizard in the user interface steps the user through the required inputs, such as the training data set and some training parameters like the target false alarm probability.

When training OCI anomaly detection models, the user does not need to specify whether the intended model is for multivariate or univariate data. It does this detection automatically.

For example, if a model is trained with 10 signals and 5 of those signals are determined to be correlated enough for multivariate anomaly detection, it will create an internal multivariate model for those signals. If the other five signals are not correlated with each other, it will create an internal univariate model for each one.

From the user's perspective, the result will be a single OCI anomaly detection model for the 10 signals. But internally, the signals are treated differently based on the training. A user can also train a model on a single signal and it will result in a univariate model.

Training a model requires a single data file with no anomalies that should cover a complete business cycle, which means it should represent all the normal variations in the signal. During training, OCI anomaly detection will use a portion of the data for training and another portion for automated testing. The fraction used for each is specified when the model is trained.

When model training is complete, it's best practice to do another test of the model with a data set containing anomalies to see if the anomalies are detected and if there are any false alarms. Based on the outcome, the user may want to retrain the model and specify a different false alarm probability.

Also called F-A-P or FAP, the FAP is the probability that the model would produce a false alarm. The false alarm probability can be thought of as the sensitivity of the model. The lower the false alarm probability, the less likelihood of it reporting a false alarm, but the less sensitive it will be to detecting anomalies. Selecting the right FAP is a business decision based on the need for sensitive detections balanced by the ability to tolerate false alarms.

Once a model has been trained and the user is satisfied with its detection performance, it can then be used for inferencing. New data is submitted to the model and OCI anomaly detection will respond with anomalies that are detected. The input data must contain the same signals that the model was trained on. So, for example, if the model was trained on signals A, B, C, and D, then for detection inferencing, the same four signals must be provided. No more, no less.

As will be seen in the demo, the training and inferencing features of OCI anomaly detection can be accessed through the OCI console. However, a human-driven interface is not efficient for most business scenarios.

In most cases, automating the detection of anomalies through software is preferred to be able to process hundreds or thousands of signals using many trained models. The service provides multiple software interfaces for this purpose.

Each trained model is accessible through a REST API and an HTTP endpoint. Additionally, programming language-specific SDKs are available for multiple languages, including Python. Using the Python SDK, data scientists can work with OCI anomaly detection for both training and inferencing in an OCI Data Science notebook.

So how does a data scientist take advantage of these capabilities? Well, you can write code against the REST API or use any of the various language SDK shown here. But for data scientists working in OCI Data Science, it makes sense to use Python.

In this lesson, you learned that OCI anomaly detection is the first AI Decision Service with more to follow. Look for OCI forecasting in the near future. OCI anomaly detection is for detecting anomalies in multivariate and univariate time series data and it's always trained with the actual signal data.

7.11 Oracle AI APIs and SDKs

[AUDIO LOGO] Now that you've seen what the services can do, let's look at how to access them in code. So how does a data scientist take advantage of these capabilities? Well, you can write code against the REST API or use any of the various language SDKs shown here.

But for data scientists working in OCI Data Science, it makes sense to use Python. Next, we'll look at what it takes to use the Python SDK in a notebook to be able to use the AI services. You can use a Notebook session in OCI Data Science to invoke the SDK for any of the AI services.

This might be useful to generate new features for a custom model or simply as a way to consume the service using a familiar Python interface. But before you can invoke the SDK, you have to prepare the data science notebook session by supplying it with an API Signing Key.

Signing Key is unique to a particular user and tenancy and authenticates that user to OCI when invoking the SDK. So therefore, you want to make sure you safeguard your Signing Key and never share it with another user. You can obtain an API Signing Key from your user profile in the OCI Console, then you save that key as a file to your local machine.

The API Signing Key also provides commands to be added to a config file that the SDK expects to find in the environment, where the SDK code is executing. The config file then references the key file. Once these files are prepared on your local machine, you can upload them to the Notebook session, where you will execute SDK code for the AI service.

You'll see a demo next that will show you these steps. The API Signing Key and config file can be reused with any of your notebook sessions, and the same files also work for all of the AI services. So the files only need to be created once for each user and tenancy combination.

7.12 Demo: Oracle AI Decision Services

[MUSIC PLAYING] In this demo, I'll show how to invoke the OCI Anomaly Detection API from an OCI Data Science notebook session. In the demo for the previous lesson on the perceptual API services, I showed how to generate the API signing key and configure the notebook session with the key and config file. So I won't repeat that here, but that's still a required step.

Before we go to a notebook, I think it'll be helpful to go through the process of training in OCI Anomaly Detection model in the OCI Console and using it to detect anomalies on some data. Those same steps then will apply when using the API from a notebook. Before we go to OCI Anomaly Detection, we need to have some data already staged in object storage. So I'm going to start there.

I'm going to go to an object my object storage bucket. I've already got the right compartment selected here, and I have a bucket that I previously created. And in that bucket, I've uploaded two files. The one I'm most interested in and will using here in a minute is this training data CSV file. That actually contains the training data that will be used to train the model. And then there's an additional file here, a file containing testing data that can be used as well to test the model.

So now we can go on to anomaly detection. So what we need to do in anomaly detection before you can build a model or do anything is you need to create a project to put that model in. So I'm going to create a new project here. And I'm going to call it console demo.

Now, I'll open the project, and I can jump right into creating a training model, but actually, what I want to do first is create a training-- or excuse me, a data asset for my training data. So I've given it a name, and I want this to be-- I wanted to use object storage for the source of the data. I'm going to pick my bucket. And then I can pick which file that's in that bucket that I want to use.

Now, I'll go ahead and create the training data asset. So that's now ready to go. Now, we can come back and create our model. So it's only had one data asset in my project. It's already selected that one by default. I'm using an existing data set. And now, we'll create our model, and I'm going to call it console model. And I can select here a target alarm, a false alarm probability. I'm going to pick a 0.2 for this, and then we'll leave the training fraction ratio at 70%. S0 I will use 70% of the training data for actual training purposes, and the other 30% I will use for an internal validation test.

Just confirm everything looks good. And we'll go ahead and create. Now, at this point, we see the model here appear in the list. It's in the creating status, and it's going to take a few minutes to create. So we'll use the magic of time lapse to wait until it is complete. Now, the model is active, which means the training is complete, and we can open up the model and use it to detect some anomalies.

So I've got a file here, ready to go. We'll drop it in and do some detection. And what it's showing us here is that they're across the 10 variables that were the 10 signals that were in that data. We've detected anomalies on three of those. And so what we see is the individual anomaly detections as shown by these little red and purple lines. So the yellow is the actual value of the signal, the purple is the estimated value based on the algorithm, and the red is the actual anomaly value.

So in this case, it expected the signal to be up here, but it was since it was actually down here, that's an anomaly. And here, it expected it to be lower and we've got an anomaly. Same for both temperature 2 and 3, very similar looking patterns there. In pressure 2, we get an anomaly down here. It's a little later in time than the temperature readings. And here we see aggregated anomaly based on the anomaly score and then anomaly score poor signal long the times camp. So you can see here a pretty good picture of what's going on across all those signals, and here's where our anomalies are.

Now, we'll go into notebook next and use the same data to generate an anomaly there, or excuse me, to generate a model there. So in this next segment, we'll use Python code in an OCI Data Science notebook to do the same steps we just did manually in the OCI Console. So let's go to our data science notebook. I'm in a OCI Data Science notebook session.

And in this session, I've already got a notebook loaded, called anomaly detection, and what we're going to do is just step through this notebook session. And I'll execute it as we go along. And what it's going to do is it'll create an anomaly detection project, then just like we did in the console, it'll create a anomaly detection data set for the training data, and it will create and train the anomaly detection model, and then we'll use the model to detect some anomalies.

Now, there are some prerequisites here. We need to be using the right conda environment, which I do have loaded already. And we needed to set up our API key files in the notebook session, and that's already been done. I went to a terminal, moved those files into the .OCI folder and confirm they're there. So we're all good on that point.

Now, we're ready to actually start executing some code in the notebook. First step here is just to do some imports. And now, this next cell shows that we need to define some constants. And these are based on the particular environment we're using, and these are going to be used to interact with OCI Anomaly Detection. So we need the location of the config file. We're going to be Reading that in a moment. We need to know what the service endpoint is, that is, each region has its own service endpoint. And you can find that in the documentation. And I put the URL there.

Then we need to know the name space of the bucket, the object storage bucket, that has the data that we're going to use for training, and what the name of the bucket is, and then what the training file is in that bucket, and what the compartment ID is that contains both our anomaly detection project and our bucket. So I've already collected all that information and pasted it into the notebook. Now we can just execute that code. And the other thing it did here, it read in the configuration from the config file that we put in basically this file right here. And then it created the object that we'll use to interact with the anomaly detection service.

Now, in this next step, we're starting to actually do some things in anomaly detection. So we're going to create the project, and I've given it the name demo project. That's completed. Now, we're going to read back the project. And so we can see what was created here. That was the One Project we created with the demo project name. And now, this next step just lists all of the projects that are there in anomaly detection in that compartment. And we've actually got multiple projects there, some I'd created previously. But we see at least our demo project in there.

Now we're moving on to preparing the training data as a data asset. So we'll specify some of the information that we set up in the constants previously for the namespace and the bucket name and the file name. We'll give it the data asset information, then we'll actually create that data asset.

And now we'll go ahead and get that data asset and look at the metadata for it here. That looks all good. Now, we're moving on to creating and training the anomaly detection model. So first, we've got to specify some details about the model like the target, false alarm probability, the training fraction. You may remember that in the console, we can also set these two parameters. And now we got to provide some additional metadata like the model name and description, compartment, and so forth.

And finally, we get to the point where we're going to create the model. We'll go ahead and get the model metadata, and then we're just going to wait in a loop here for that model to become active. So it's read back the model metadata. Every 60 seconds, it's going to print out the current life cycle state of that model. It's going to show creating for a while and it'll eventually get to active. So we won't make you sit through all of that. I'll cut that out, and we'll go right to the point where it becomes active.

Now we see the model is active, and we can move on to the final step of the process here, which is detecting anomalies. So first we're going to load the data from the CSV file. We've got the testing data here in CSV format that I've already uploaded to the notebook, and we're going to read that in into a dataframe. And here's what that data looks like. We've got the time stamp, we've got the five temperature readings, the five pressure readings, and we've got 100 rows of data there. We've also got another method that's commented out here in the notebook if you want to just generate some random data and treat that as a test data. But it's obviously not going to be quite as realistic.

Now we'll create a payload from that dataframe to use to submit to the model. And now let's detect some anomalies. And we get a response back very quickly, and we can see the data here in JSON format, we get the detection results, we get anomaly on temperature 3, another one on temperature 3, then we go, we get some anomaly on temperature 2, another one on temperature 3, pressure 2, pressure 2. So it gives us for each anomaly, we see the actual value of the signal, the estimated value of the signal, the anomaly score, which signal it was, and then the overall score in the time stamp. Same information you saw in the console but now just listed out in JSON form.

Well, that concludes this demo.