# **Dataset: https://data.police.uk/data/open-data/**

# **Debiasing AI Using Amazon SageMaker**

### **Debiasing AI using Amazon SageMaker**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Kesha] Are you curious about machine learning? Maybe you've been wanting to learn about building models using Amazon SageMaker? Or maybe bias in AI is a huge concern for you, and you want to learn ways to mitigate it? If so, this course is for you. In this tactical hands-on course you will learn how to build a crime-fighting machine learning model that is fair, transparent, and explainable using Amazon SageMaker. You will also learn the steps to uncover and remove bias in training data before a model is created. You will learn all of this through a fun crime-fighting case study that integrates Amazon SageMaker, with Rekognition, and the AWS DeepLens camera. Creating a crime-fighting model that can see what's happening in a particular scene. Hello, I'm Kesha Williams. I'm a software engineering manager, author, and international speaker. I've been in the IT industry for over 20 years, and I believe that machine learning is the next wave of transformative technology that will change life in ways that we can't even begin to imagine. So let's get started learning how to build bias-free machine learning models using Amazon SageMaker, AWS DeepLens, and Rekognition.

### **What you should know**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] This course is for you if you want to to know about building machine learning models, using Amazon SageMaker, that are fair, transparent and explainable. Before beginning this course, you should have basic knowledge of Amazon Web Services and a free-tier account. Some exposure to the Python programming language is helpful, but not required. Exposure to machine learning is not required.

### **Predictive policing**

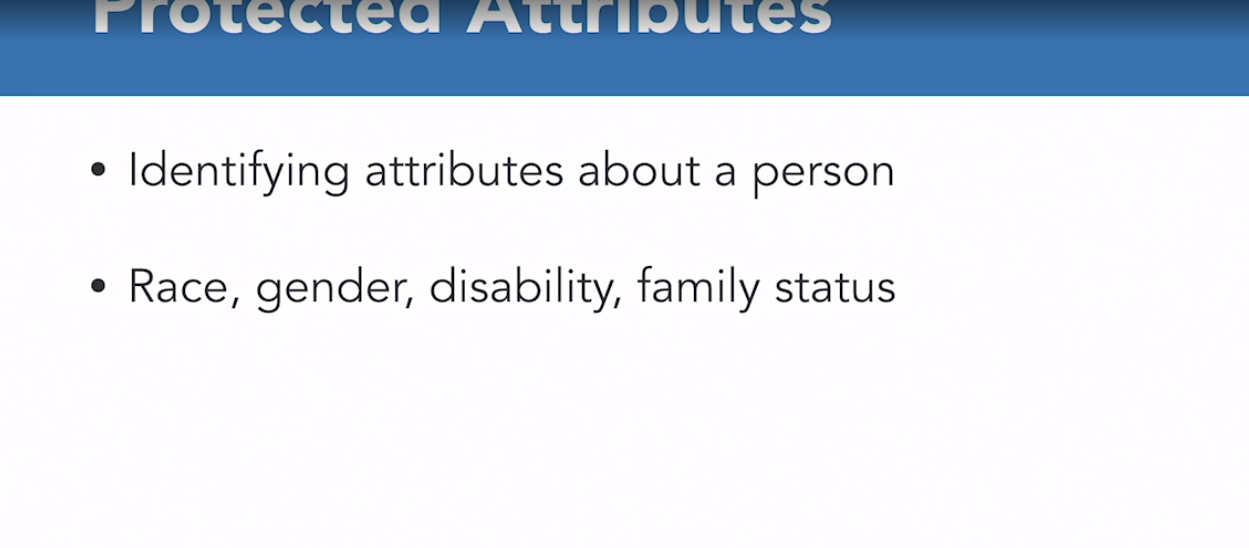
Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Predictive policing may not be a term that you're familiar with, but it is practiced around the globe in well-known cities, like New York, Chicago, Los Angeles, and even London. Some of these cities have spent hundreds of thousands, even millions of dollars to implement predictive policing systems. And the popularity of this technology is growing at a rapid rate. So, what is predictive policing? It describes the process of using artificial intelligence, like machine learning, to predict crime. The technology is used in the courthouse and when officers are out on patrol. Generally, any time machine learning is used in law enforcement, it is called predictive policing. Did you know that, today, some judges use machine learning to predict if someone is going to be a repeat offender or not? Based on what the machine says, a person may or may not receive parole or bail. In research done by the University of Pennsylvania, arraignment decisions made with the assistance of machine learning cut new domestic violence incidents by half. Officers on patrol make use of machine learning by analyzing hotspot maps produced using the technology. The hotspot map shows areas in a city where crime is most likely to occur, and that's where officers patrol. The belief is the heightened police presence in those hot areas reduce crime by making the areas less attractive to criminals. For this course, predicting crime is a great real-world use case for machine learning. Crime is a topic that most everyone can identify with, and the thought of being able to reduce crime using a technology like machine learning is exciting. Predictive policing is also a great case study for bias in machine learning. There are several reports that show certain predictive policing models are biased against minorities, which raises a lot of concern. Machine learning models have a tendency to take on the bias of their creators, either through data or by what the creator tells the machine is important. As creators, we need to tread carefully to ensure our personal biases aren't passed on. During this course, we will address bias and ways to prevent it from creeping into the machine learning models that we create.

### **Overview of crime-fighting case study**

Selecting transcript lines in this section will navigate to timestamp in the video

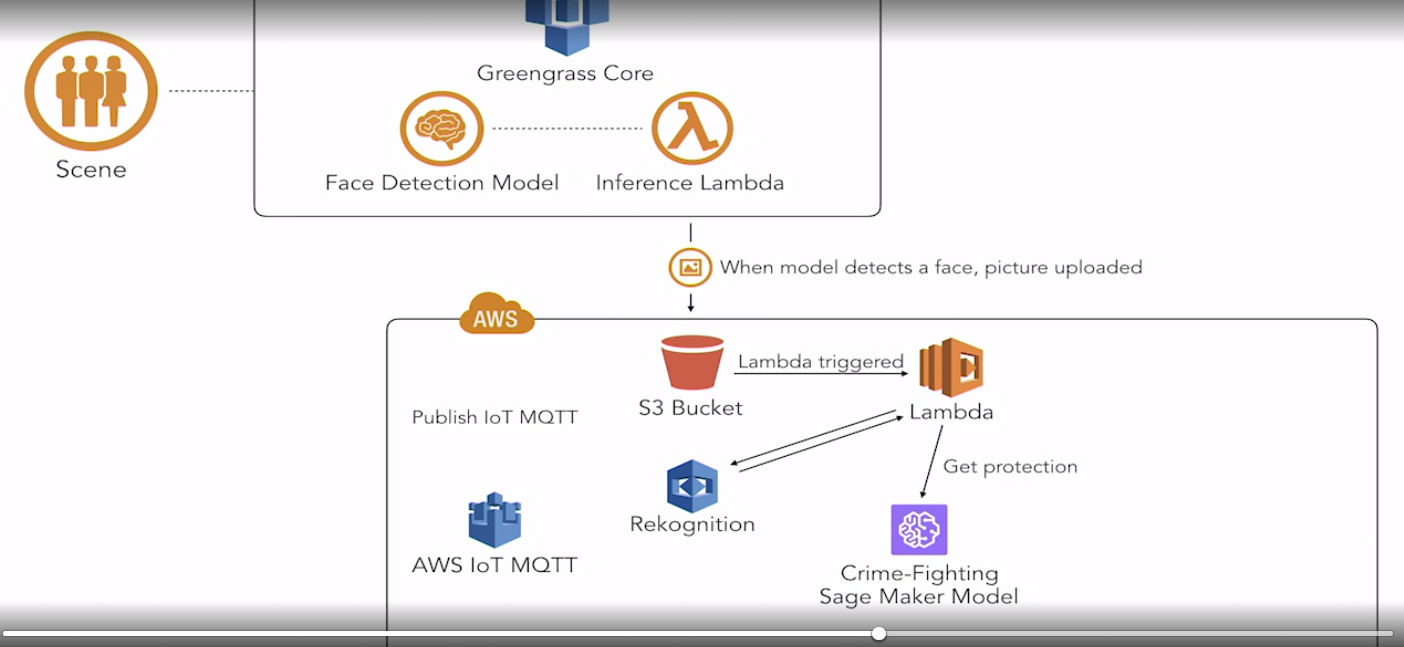
- Every time I hear the term predictive policing, I have flashback to the precrime concept from my favorite science fiction move, Minority Report. In this movie, the precrime division was responsible for arresting criminals at the moment right before they committed a crime. The police officers had foreknowledge of the crime, thanks to psychic technology. I started thinking to myself, what technology do we have today that can predict crime? Machine learning of course. Machine learning is the process a machine goes through to study and learn from past data in order to make predictions about the future. So, I set out to build precrime using Amazon's SageMaker. Minority Report inspired the crime fighting machine learning model we create during this course. I will show you the process I used, step by step. Data, specifically past crime data is key when training or teaching a machine learning model. Now I know I've used the term model several times, but what is it exactly? You could think of the model as an all knowing brain, that grows smarter and smarter, the more data it has access to. In actuality, the model is a mathematical model that holds patterns and trends found in data. Our model specifically holds patterns and trends found in crime data. This model can be consulted or questioned, when new data comes in, in order to determine the likelihood of crime. The answer provided by the model as to whether or not crime is likely, is called the prediction. Now, let's talk about the data we are going to use to train our crime fighting model. In my search for crime data, I found open data about crime and policing in the United Kingdom, which includes reports for England, Wales, and Northern Ireland. So, I downloaded a year's worth of stop and search crime data. The data contains records by date, location, and type. It also contains identifying details about the person that was stopped. The data set that we will use to train our machine learning model contains positive records, or records of people that were stopped, searched, and arrested. It also contains negative records, or records of people that were stopped, searched, and released. Now remember this idea of having both positive and negative records in a single file. I've given you a jumpstart by preparing the data our crime fighting model needs for you. The preparation step can be a bit tedious, but the end result is very important and that's data in a format that a machine can easily read and understand. First, I downloaded data files for each county, as shown here in column B. So examples, Devon Cornwall, Surrey, Essex et cetera, and then combine the separate files into a single file. Next, I converted the last outcome category column which indicates if a person was arrested or not, to a one or zero and we see that here in column A. A one represents the positive record, a person was arrested, and zero, let's scroll down and find a zero, zero indicates the negative record, the person was released. So here's a zero for the county of Kent. Lastly, I took the date time stamp and put it in a more meaningful format, a format that allows for patterns to be easily identified. From the date time stamp, I pulled out the time of day, shown here in column C, the day of the week, shown here in column D, and the month, shown here in column E. As we go through this course, we need to be aware of identifying attributes about a person that are considered to be protected attributes. So attributes, such as race, gender, disability, family status et cetera. Protected is a common term when dealing with sensitive information. If we are not careful, bias can creep in when machine learning models are taught or trained using these attributes. We will keep a close eye on these attributes to ensure our model is fair. Now that we've discussed the inspiration behind the crime fighting model, and the source of the data, let's look at the technical architecture.

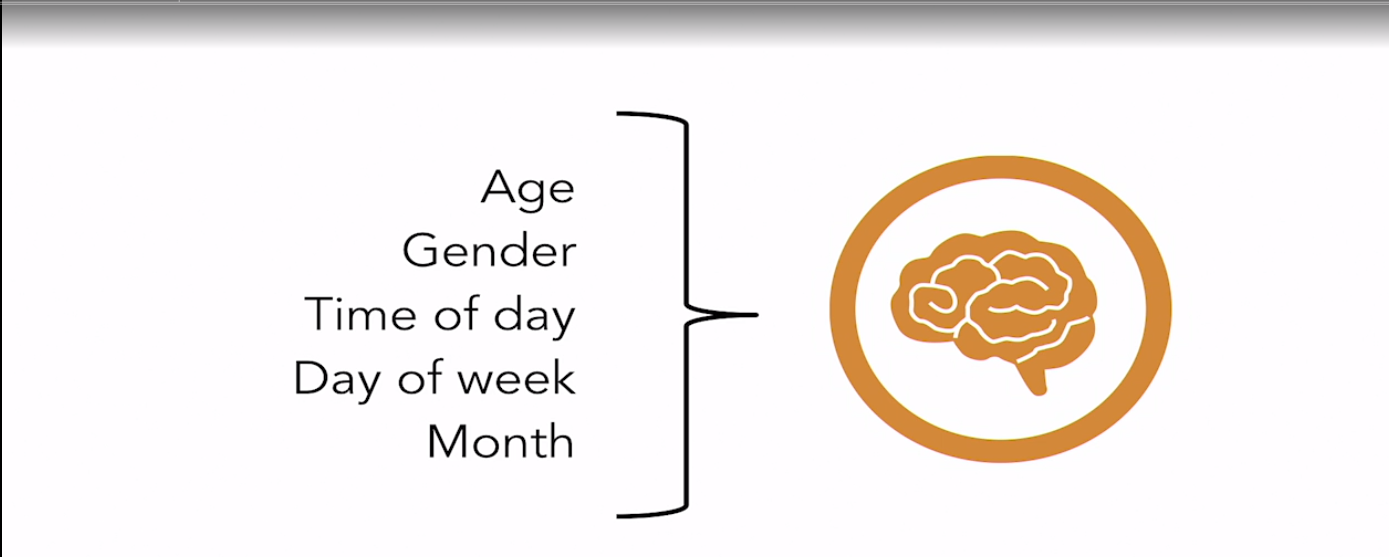


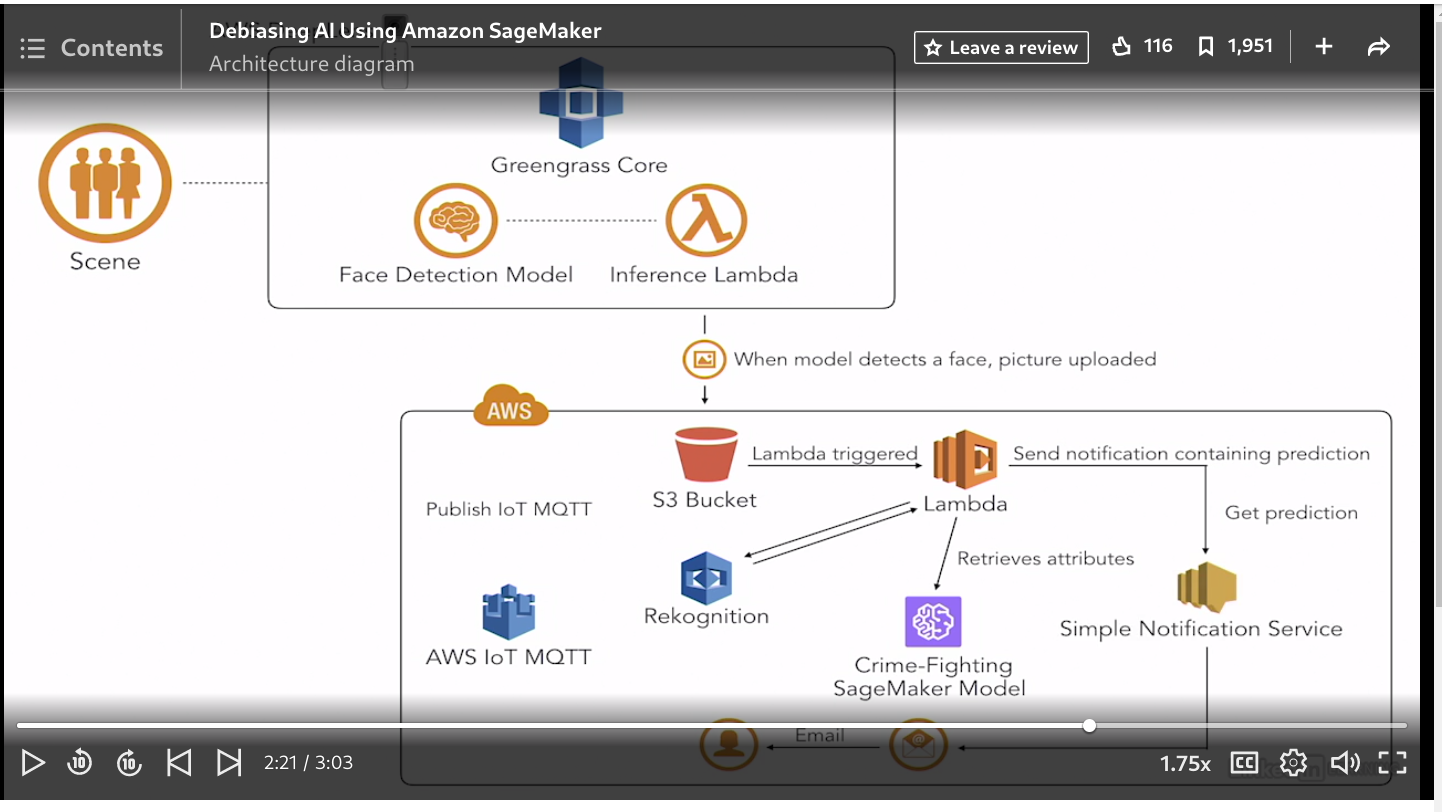
### **Architecture diagram**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Narrator] Let's look at the components that make up the architecture of our crime-fighting case study. First, AWS DeepLens is the video camera that is placed in a location to observe a given scene. DeepLens is not a normal camera, though. It is smart because it is running a face detection machine learning model that I've deployed to it. Thanks to AWS Greengass, a service that extends AWS functionality to IoT devices, I'm able to run lambda code on the video camera. All of this activity happens at the edge directly on the device itself. As soon as a face is detected in the scene, a picture is taken and uploaded to an S3 bucket in the cloud for analysis. S3 is an object storage mechanism and can store objects of many different types. The photo upload to the bucket triggers lambda code that retrieves the photo and sends it to AWS Rekognition. Rekognition is a computer vision service that gives computers a visual understanding of the world around it. Rekognition analyzes the photo and identifies attributes about the people or person in the photo. For our crime-fighting case study, we only care about two attributes returned from the photo via the Rekognition service: gender and age. Gender and age along with other attributes that we'll discuss shortly are then sent to the crime-fighting model. Amazon SageMaker is the tool of choice for training the model on the crime data we previously prepared. SageMaker makes the model accessible via an API endpoint and can be consulted to help us determine the likelihood of crime in our scene. Along with age and gender, the model needs the location, time of day, day of the week, and month in order to make an accurate prediction or inference. After providing the needed data points, a prediction is sent back and delivered to the user via email using the simple notification service. Using DeepLens to kick off the process is really fun, and using Amazon SageMaker to produce the model is simple and straightforward. Additionally, the architecture is super flexible. Let's say you don't have a DeepLens device. Well, as long as you manually upload a photo to the S3 bucket to trigger the process, you can still obtain a crime prediction from the model. Now that you better understand the architecture, let's look at the development cost and ongoing maintenance cost for each component.







### **Tools, services, and costs**

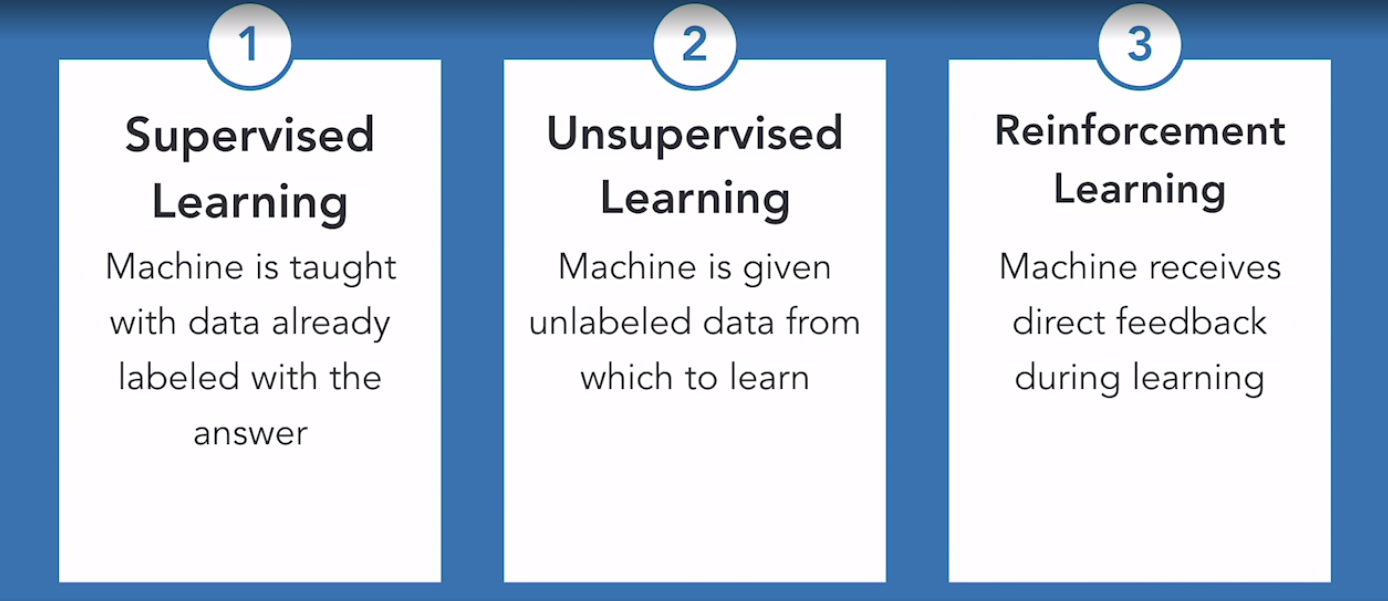
Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Let's discuss the tools needed and services you'll use during this course to build your crime-fighting machine-learning model. I've navigated to the AWS Free Tier page. You will first need an AWS Free Tier account. So this account gives you free access for 12 months to most of the products and services provided by Amazon. When signing up for an account, you will need an email address and a credit card. For machine learning, we'll use Amazon SageMaker. So the free tier includes 250 hours of usage for the first two months only. So although SageMaker is a part of their free tier program, you do not have 12 months free; only two months free. And the hours are used for building, training, and deploying machine-learning models. And as a part of building, we'll use a Jupiter notebook from the SageMaker console to analyze and prepare our data. For object storage, we'll use the Simple Storage Service, or S3. And we'll store our files, or objects, in buckets. If you recall from the architecture diagram, we uploaded a photo to an S3 bucket to kick off the process to obtain a crime prediction. Five gigabytes is included in the free tier. For compute services, we'll use Lambda and the embedded Cloud9 IDE in the Lambda console. We'll use that to author the Python code that is triggered when a photo is uploaded. The free tier includes one million free requests per month. For computer vision, we use Rekognition to analyze the photo of the scene. As a part of the free tier, we can analyze 5,000 images per month. To send our user an email notification containing the crime prediction, we use the Simple Notification Service, or SNS. The free tier includes 1,000 notifications per month. A DeepLens device is used to take a photo of the scene and upload it to the S3 bucket only when a face is detected. The current cost for a DeepLens device is $249. I know that is not cheap. If you don't want to invest in the DeepLens device, then just bypass that step and manually upload a photo to the S3 bucket to trigger the process. I will remind you about this later. For those of you not on a free tier plan, like me, the total cost to build, train, and deploy my crime-fighting model using SageMaker was almost $300. Now yours should not be that expensive, because I went through multiple training cycles, and I also left my endpoint enabled for an entire month. I will show you how to disable everything so that you don't end up with a big bill. Now you are lucky if you're on the free tier, but don't forget. You only have the free tier of SageMaker for two months. Throughout this course, I'll demonstrate all of the concepts we cover, so you can always just watch me and follow along if you don't to recreate the crime-fighting model in your personal AWS account. Now that you've been introduced to the tools, services, and costs, let's talk about a few of the more common machine learning terms you'll hear.

### **Terms and concepts**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Let's talk about machine learning terminology. You should be familiar with the term Model, which is a mathematical representation of trends and patterns found in data. You've also been exposed to the term Training, which is the process of teaching the model by giving it large amounts of data to analyze. The model is the result of the learning or training process, which uses a learning algorithm behind the scenes. SageMaker comes with several built-in learning algorithms, which gives you a big headstart in training your model. We'll look at the out-of-the-box learning algorithms in more detail later. And you may remember inference, the prediction a model provides. How do these terms and concepts fit into the overall machine learning process? Well, there are three ways that machines learn: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is when the machine is taught with data that is already labeled with the answer or target variable you are trying to predict. A single row in that labeled data is called a data point, and each element that goes into making a prediction is called a feature. Unsupervised learning is when the machine is given unlabeled data from which to learn. This type of learning is typically used to cluster data and find hidden patterns and relationships. Reinforcement learning is somewhere in between supervised and unsupervised learning. As the machine goes through the learning process, direct feedback is given. This is used a lot in robotics. Now, let's go a bit deeper on supervised learning, since this is the method used by our case study. With supervised learning, there are several types of questions that a machine can be taught to answer. The first question type is called regression, which provides a numeric answer to a question. For example, what will the temperature be in London tomorrow? Or, how many copies of this book will sell? Or, how many crimes will be committed in this city? The second question type is called classification, which predicts a discrete value. For example, there's multi-class, which predicts one of many, like, is this item a toy, movie, or book? There's also binary classification, which is one of two, typically a yes or no answer, like, is this email spam or not? So, our case study is the perfect example of unsupervised learning, because we provide it with labeled data containing the answer. And it uses binary classification to answer the yes or no question, is a crime about to occur? Now that you understand how machines learn and the different types of questions a machine can be taught to answer, let's see our case study in action.



### **Demo of Amazon SageMaker**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now let's take a look at my crime predicting model so that you can see it in action and then I'll walk you through step by step how to build it yourself later in the course. The first step is to observe the scene from the deep lens camera. Here is an image of my deep lens set up. I have my deep lens connected to a monitor, a keyboard, and a mouse, so that I can quickly see what the device is seeing, the scene it's capturing. So here, in this monitor, that is what the deep lens camera sees. However, this device can connect to Wi-Fi so that you can place it at any location without the need to connect it to a monitor, keyboard or mouse. Once the device is fully configured, you can pull up the scene in a web browser. AWS Deep Lens produces two output streams. What is shown here, is the unprocessed video stream, called the device stream. The video stream processed by the model is called the project stream. Images from the project stream are what get loaded to the S3 bucket. I will show you an image from the project stream momentarily. Now, let's navigate to our AWS Deep Lens console. I've deployed one of deep lens' pre-trained face detection models to the device. Once a face is detected, the inference Lambda running on the device uploads a photo of the scene to S3. So let's navigate to the S3 bucket to see what's been uploaded. Once a face is observed, there is a slight pause, so that we don't have multiple uploads of the same face to the S3 bucket. Let's download an opening photo, which should be of our scene. So let's click on this very last frame photo here. Let's open it. So this photo accurately depicts our scene. The upload triggers a Lambda. The Lambda retrieves the photo from S3 and sends it to recognition to get the age and gender of the person. Additional information like location, time of day, day of week, et cetera, along with age and gender are sent to the crime fighting model. As the Lambda executes, its output is logged using a service called CloudWatch. So here is the CloudWatch log. We see the output here, and notice at the very bottom there is a prediction and it says no crime. Here, we can see the confidence score. So essentially this is saying that there's a 27% probability that a crime is about to occur. Notice this row here. This is the data that is sent to the endpoint for the crime fighting model to make a prediction. The crime fighting model is hosted in Amazon SageMaker. Let's look at the endpoint for the model. So this is the crime endpoint that's hosted in Amazon's SageMaker. It is in service and this is the URL. This what we use in our Lambda code to retrieve a crime prediction. We can see the output log for the execution of this model in CloudWatch as well. So let's take a look at that. Here we see the output of the calls to the endpoint. There are several successes. This log would also be very helpful for failures, as detailed error messages would also be found here. So this just shows an indication of the model. Lastly, by this time, SNS should have alerted the user via email with a crime prediction. Let's see if the crime fighting model thinks I look suspicious. So let's navigate to my email. Here I have an email with the title AWS Notification Message from Amazon AWS. In this email, there is a name of the camera location and the crime prediction of no. Now that we've seen our crime fighting case study in action, let's learn more about building a model via SageMaker.

## **Question 1 of 10**

What's a common way that machine learning models learn to be biased?

* analyzing hotspots
* training  
  Incorrect
* data  
  Correct
* data visualization

## **Question 2 of 10**

Which is one place where predictive policing would NOT be used?

* airport immigration
* courthouse
* police precinct
* restaurant  
  Correct

## **Question 3 of 10**

What does the mathematical model use to make a prediction?

* data  
  Incorrect
* trends and patterns  
  Correct
* educated guess
* positive records

## **Question 4 of 10**

In machine learning, what are identifying attributes about a person called?

* personal data
* sensitive information
* personally identifiable information (PII)  
  Incorrect
* protected  
  Correct

## **Question 5 of 10**

What service extends AWS functionality to IoT devices?

* AWS Deep Lens
* Greengrass  
  Correct
* AWS SageMaker
* AWS IoT Core

## **Question 6 of 10**

Where is a photo of the scene captured by AWS DeepLens uploaded?

* SNS
* S3  
  Correct
* AWS
* AWS DeepLens

## **Question 7 of 10**

SageMaker is free for 12-months.

* TRUE
* FALSE  
  Correct

## **Question 8 of 10**

Which tool is used to analyze and prepare data?

* Excel
* Cloud9 IDE
* Jupyter Notebook  
  Correct
* Lambda  
  Incorrect

## **Question 9 of 10**

What is the term for a prediction returned from a model?

* inference  
  Correct
* analysis
* regression response
* guess

## **Question 10 of 10**

How does our crime-fighting model learn?

* reinforcement
* combination  
  Incorrect
* unsupervised  
  Correct
* supervised

### **What is SageMaker?**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] What is SageMaker? SageMaker is a fully manged machine learning service that caters to everyday programmers and data scientists. There are several universal steps in the machine learning process such as obtaining and preparing data, training a model and then testing and deploying a model via an endpoint. The beauty of SageMaker is that it brings all of these steps under one umbrella which helps to streamline the overall process making it easier. For the data preparation step, SageMaker provides Jupyter Notebooks. In our case study, we'll use a Jupyter Notebook like this to visualize, clean and prepare our crime data stored in S3. For the next step of training a model, SageMaker provides servers, or instances in AWS terminology, that we can use to train our model. During the training process, a learning algorithm is used. SageMaker comes with several built-in learning algorithms that we can use to train our model. The learning algorithm that you select is based on your specific use case, so it's important to understand the business problem you are trying to solve and the answer you are trying to predict before selecting one of these algorithms. So, let's look at a few. Let's look at the K-means algorithm. This algorithm is used for clustering data in an unsupervised learning context. There is also an image classification algorithm. This is used for supervised learning and takes an image as input and classifies it into one of many categories. The BlazingText algorithm provides an implementation of the Word2vec text classification algorithms, so this algorithm is used a lot for natural language processing tasks. The SageMaker documentation is a great reference for all of the built-in algorithms provided. For regression and classification models, SageMaker provides Linear Learner and the XGBoost algorithm. Since our use case is binary classification, we'll use the Linear Learner and set the predictor type hyperparameter to binary classification. We'll see this in action later. A hyperparameter is like a property that you set before training begins to help you configure the algorithm. Binary classifier simply means that the algorithm needs to study the data to figure out how to answer a yes or no question. Linear Learner is represented as a linear equation where making a prediction is simply solving the equation for a specific set of inputs. So, you have a set of input values X that you are trying to predict the output value Y for. Linear Learner includes support for common linear models. Specifically, logistic regression is used for classification problems, and linear regression is used for predicting numeric values. SageMaker also allows you to bring your own algorithm if none of the built-in algorithms fit your needs. If you are already familiar with TensorFlow or Apache MXNet, you can use SageMaker to train a model using your own custom TensorFlow or MXNet code. SageMaker is very flexible and can meet you where you are in your data science journey. Okay, so once you have established your use case and selected the appropriate learning algorithm, for the last step, deploying a model, SageMaker provides the ability to make the model accessible via an API endpoint. This allows other applications to talk to the model and obtain crime predictions. Now that you have an understanding of SageMaker and how it is used to build our crime fighting case study, let's learn more about the machine learning process.

### **Machine learning process**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] There are several standard steps in the machine learning process. Let's talk about each step in more detail. First, you have to find the data you need to train your model with. The data can be yours, like your customer data you've collected over time, or you can obtain the data from machine learning repositories, like the UC Irvine Machine Learning Repository, or from government entities that make data freely available. Once you have your data, typically, you need to clean and prepare it. An example of cleaning data would be taking a date field and splitting it out into month, day of the week, and year. In this case, it makes it easier to find patterns when the components of a date are split out. I cleaned all of the crime data myself using Excel before training the model. **This part is the most time-consuming. Some companies outsource this step using a service like Amazon Mechanical Turk**. Once the data is prepared, we train the model. This is the process where the machine actually learns from the data. The training processes uses a learning algorithm. For our case study, we're using a linear learner algorithm. But what's really happening behind the scenes? The learning algorithm consists of a loss function and an optimization technique. During the training process, several passes are made over the data, trying to find patterns by making optimizations and attempting to minimize loss. Loss is considered the penalty for making an incorrect prediction. And the optimization technique we use is called stochastic gradient descent. Now, SageMaker does allow you to control the optimization process by choosing the optimization technique or algorithm. All of this simply means that, during training, several models are created, tested, and thrown away until a well-performing model is reached. Each iteration through this cycle of creating, testing, and throwing away is called an epoch. We can see an example of the training process by reviewing the log file. So, here, this log file shows the training process. On the first few lines, we see accuracy scores. On the very bottom, we see that the loss was improved, the system updated to provide the best model. Once a model is created, it should be evaluated. During the training process, you should set aside portions, called channels, of your data for training and for validation. The machine learns from the channel set aside for training, and then it tests how well it's performing at predicting crime by using the channel set aside for validation. The Amazon SageMaker linear learner algorithm supports three data channels: train, validation, and test. But of those three channels, the only one that is required is train. The test channel is used for logging metrics, which includes the accuracy score for the final model produced. Once the model's been evaluated, it's ready for deployment. Once deployed, it can be used to make a prediction. Now, deployment is not really the final step, because you can start this process all over again. And, in fact, that's good, because you should start the process over as new data comes in. This process is called retraining, and it makes your model even smarter as it learns from new data and new features. Now that we've reviewed the machine learning process, let's learn how to inspect and visualize our data using a Jupyter notebook.

### **Prepare the data**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Before we use the crime data to train our model, it needs to be cleaned up a bit. For example, there are null or empty values for some of the features, and inconsistent gender records. So let's look at cleaning and preparing the data. Let's first identify all the records that have null or empty values. Click on this code block and click Run. We see there are 159 null gender records, and 272 null average age records. We need to get rid of those, because we don't want our model to learn from empty values. Let's drill down even more to see the specific age records that are null. Click on this code block, and click Run. In this case, where there aren't any transactions, it puts in a value of NaN, which means not a number, as seen here. Now let's explicitly remove those records. So let's scroll down to this code block, click on it. So we're going to remove those records, and then double-check that those records have been removed. So let's execute this code block. So we've removed the records, and then we did a count to see how many records were still null, and the value is zero. So null is gone. Let's do the same thing for gender, and then eliminate the inconsistent gender records. So let's click on this code block and click Run. So we see for gender, we have the NaN and we have other. If we continue to scroll down, this code block here, click Run, will actually remove those. So line two removed it, line five did a count to see, and it returned zero, so it has been removed, so now let's remove other. Click on this code block, let's click Run. And this shows how many records that have gender as Other. So now let's run this code block. On line three we are removing those records. On line six, we're double-checking to see if they have been removed, and the printout here shows no rows that contain other. So let's look at our data frame again, so click on this code block and click Run. And we see it contains good data, so let's scroll down. And now we have 10,543 rows, and still seven columns. Now let's talk about data encoding and transformation. The learning algorithm that we use only works with numerical values, so we need to go through the process of converting our non-numerical data to numbers. So let's look at the data type for each of our features. Click on this code block and click Run. Notice here that all are text objects, except for age and CommittedCrime, so we'll need to convert all features except these two. So let's scroll down to this code block and take a look at what it's doing. I'm using the Find and Replace method to convert text to strings for features that are easily translated to numbers. Here on line five, DayofWeek is easily translated to a valid number. Monday is one, Tuesday is two, Wednesday is three, so on and so forth. The same holds true for month. Month is easily translated to a number. January is one, February is two, March is three, so on and so forth. So let's execute this block of code, make sure it's selected, has the green box around it, and click Run. And we see the output down here. Notice in month we now have numbers, and day of the week, we now have numbers. Next, let's look at a different method called One Hot Encoding. This is for values that can't easily be translated to a numeric value. This process converts each category value into a new column, and assigns a one or zero representing a true or false value to that column. Let's list all of the unique values for gender. Click on this code block and click Run. So that looks great. So let's convert those values. So it's going to go from one column to two columns, representing one or zero for male or female. Click on this code block and click Run. Notice, for gender we now have two distinct columns, one for female and one for male. When you see the number one here, that means this row is male. If you see a one under the female column, that means this row represents a female. This method has the benefit of not weighting a value improperly, but does add more columns to the dataset. Notice here on line two, we are using the Pandas get\_dummies function to make this happen. I will use One Hot Encoding for the rest of the columns to include time of day and county, so I'm going to run that code now. So if I scroll down. So this shows the three unique values for time of day. Let's run this, so now we have three columns for time of day, each one with a zero or one. So now let's do the same thing for county. Click on this row block, click Run. This shows the distinct values of counties. Click this code block and click Run. And then let's scroll down and then scroll over, so that you can see. Let's scroll back up. So notice here, for each county there is a column that has either a zero or one. So this row here represents the Devon Cornwall county. The next step is to split our training data into three sets, training, validation, and test. When you train a model on data that you've prepared, you will reserve some of that data for testing and validating. So the model will use the test data to test how well it does at predicting crime, and it will also use the validation data set for validating and we'll talk about these more in detail. The model will only train on a portion of the data that we provide. This will help prevent us from overfitting the model and allow us to test the model's accuracy on data that it hasn't been trained on. So let's execute this code block, make sure it's selected. So it completed successfully because we did not see any error messages. On lines nine through 11, shown here, it saved three CSV files to our Notebook instance, so we can navigate to our Notebook instance page and see three CSV files, test, train, and validate. So let's go back to our Notebook. The last step is to upload our newly created files to S3s, so that the Sagemaker training process can access those files. So let's execute this code block. It completes successfully because there are no error messages. Lines six through nine show the exact path and the exact folder where we upload the files. Let's navigate to the S3 console to make sure that the files were uploaded. So we created this cleaned folder, and yes we see test, train, validation, and the crime data that was cleaned. Now that we cleaned and prepared the data, let's start the training process to produce the model.

### **Train the model**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now that we have prepared and cleaned our data, let's use it to teach a model about crime. To start that, we need to create a training job, so navigate to the SageMaker homepage and scroll down here on the left hand side, look for training jobs. Click on that. It will bring you to a screen showing you all of your previously-created training jobs. To create a new one, click on create training job. Give your job a name, no spaces allowed in that name. I will call mine crime job. Leave it to the previously-defaulted AIM role that was created. We see here that Amazon SageMaker built-in algorithm is defaulted. Leave that selected and then choose an algorithm. So Amazon SageMaker provides a long list of algorithms. In this case, we are using Linear Learner. Scroll down, make sure your input mode is file because we are using a CSV file. And then you can scroll through and just look at the different metrics that will be published during the training process, and it's published to a service called CloudWatch and I will show you how to access those logs later. So keep scrolling. For the instance type, leave it at extra large, and accept all of the other defaults. So now let's look at the hyperparameters. This portion is very important. So hyperparameters are configuration properties that you set for your training job. For most of these, I'll leave the defaults. Let's take a look at a few. So here, for epochs, there's 15. So that means the pass is over the data. I will manually set this predictor type to binary classifier because we are using binary classification and I will set the feature dim field to 15. That represents the amount of columns in our crime data file. So the next step is to add the training validation and testing channels. So a channel simply represents an input file. So let's scroll down and this is where we will set our channels. So they defaulted a train channel here for us because that one is required. For the input mode, we will select file. For the content type, since we are using a CSV file, we'll simply type in text CSV. All of these, just leave the defaults, and now here, you need to put in the S three location to your file. So mine is in a bucket called LinkedIn hyphen SageMaker in the cleaned folder, and the file is called train.csv. So that's the train channel. Let's add the validation channel. So this one, we call validation, input mode, file, content type, text, CSV. And the location is called validation.csv, and let's add one more channel. This is the test channel, so we'll call it test, input mode, like before, file, text, CSV, and the location, and it's called test.csv. So the next step is to put the output data configuration so once the training job is completed and your model is produced, where will those files be placed? So I've created a separate S three output bucket, so I've called it LinkedIn hyphen SageMaker hyphen output. I'm not using an encryption key, not adding any tags, so then click create training job. And you will come to a screen like this. It will probably take five or more minutes for the training job to finish, so I'll pause now and come back once it's completed. Our training job completed successfully in about four minutes, so let's click on the name. So this just shows all of the details about the training job. You can click on view history to see high-level information about that job. Let's scroll down. As you will see additional details about the train channel, the validation channel, the test channel, the metrics that it's going to publish, so on and so forth. You can view the algorithm metrics by clicking on this link. You can view instance metrics here. Let's view the logs. So this is going to take us to the CloudWatch logs for our training job, so click on this name. And so here, you will see step-by-step details about the training job. So it's a lot of information. Let's search specifically for the binary classification accuracy to see how well our model is performing. So the very last record shows an accuracy score of 0.86. An excellent model has a score close to one. A poor-performing model has a score closer to zero, so our crime-fighting model is doing a pretty good job at predicting crime. Now that we completed training, let's deploy the model so that it can be used by applications that need to obtain a crime prediction.

### **Deploy the model**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now that we've completed training, let's deploy the model via SageMaker hosting so that it can be accessed by systems needing to obtain a crime prediction. So, I'm on the Training Jobs page, click on the training job. In the upper right-hand corner, click Create Model. Let's give our model a name. I'm going to call this crime-job. I'm going to leave the default for most of these options. Let's continue to scroll. Notice this line here. This is the bucket location of our model artifacts, so from the previous training step, the model produced several artifacts that were placed in this bucket. Scroll down. Click Create Model. So, the model was successfully created. Now in order to use that model, we have to create what's called an endpoint. So, click on the model, and click Create Endpoint. So, let's give our endpoint a name. I will call it crime-endpoint. Make sure create a new endpoint configuration is selected. Let's enter the endpoint configuration name. And let's just say crime-job-config. Let's continue to scroll down. Now the endpoint configuration allows you to deploy multiple variants of a model to the same API endpoint. This is useful for testing variations of a model and also for configuring the environment to elastically scale. So, let's click create endpoint configuration. And if you scroll up, you will see a success message. Now let's scroll down and click create endpoint. So, it is probably going to take five or more minutes for this endpoint to be created. Notice here it says Creating. So, I'm going to pause the video. Once it's created, we'll start up again. The endpoint is now in service. Now that we have an API endpoint that can be called by other applications, let's look at the details. So, click on the name. Notice here the URL. So, this shows the URL to the API endpoint, so we'll need to make a note of this because we'll use it in the future. This is what we'll use to obtain a crime prediction. Now that we've created an endpoint, let's configure AWS DeepLens so that we can use our crime fighting model.

## **Question 1 of 12**

Which learning algorithm does our crime-fighting model use to learn?

* Linear Learner  
  Correct
* BlazingText
* Image Classification  
  Incorrect
* XGBoost  
  Incorrect

## **Question 2 of 12**

What is a property that you set before training begins to help configure the learning algorithm?

* classifier
* input value
* hyperparameter  
  Correct
* regression

## **Question 3 of 12**

During the machine learning process, which step is used to clean the data?

* train model
* obtain data
* prepare data  
  Correct
* evaluate model

## **Question 4 of 12**

During the machine learning process, which step allows third parties to obtain a prediction from the model?

* train model
* evaluate model
* prepare data
* deploy model  
  Correct

## **Question 5 of 12**

What does a typical Jupyter Notebook contain?

* Python code and comments  
  Correct
* R code and documentation
* Lambda code and documentation
* Node.js code and comments

## **Question 6 of 12**

What is the data structure that holds the data from a file?

* Dataframe  
  Correct
* histogram
* Numpy
* Pandas

## **Question 7 of 12**

Models that learn from empty values tend to make better predictions.

* TRUE
* FALSE  
  Correct

## **Question 8 of 12**

Which data type does the Linear Learner learning algorithm work with?

* booleans
* integers  
  Correct
* strings
* arrays

## **Question 9 of 12**

Which hyperparameter represents binary classification?

* feature\_dim
* num\_classes  
  Incorrect
* Epoch
* Predictor\_type  
  Correct

## **Question 10 of 12**

Which data channel is mandatory?

* Inference  
  Incorrect
* Train  
  Correct
* Validation
* Test  
  Incorrect

## **Question 11 of 12**

Where are the model artifacts placed after the training process completes?

* API Gateway
* S3 bucket  
  Correct
* API endpoint
* Endpoint URL

## **Question 12 of 12**

What is the endpoint configuration best useful for?

* testing variations of a model  
  Correct
* automatically scales the environment  
  Incorrect
* allows DeepLens to access the model  
  Incorrect
* allows the model to be accessed by systems

### **What is DeepLens?**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] AWS DeepLens is a video camera that can run machine learning models directly on the device. This gives us the power to watch our environment and perform certain actions based on what the video camera sees. The camera can also communicate with services running in the AWS account it's been registered to. In our use case, once a face is detected, a photo of the scene is taken and uploaded to an S3 bucket. How does all of this really work? The AWS Greengrass service turns the DeepLens device into a sophisticated edge device that can process images at the source. Simply put, AWS Greengrass extends AWS cloud capabilities to local devices. Now let's look at the technical specs for the AWS DeepLens device. There is a four megapixel camera that can capture video and has a microphone that can run custom audio models. The CPU is an Intel Atom processor. 18 gig RAM of memory. 16 gig memory, dual-band WiFi, USB and micro HDMI ports, and for graphics, an Intel Gen9 graphics engine. From a software perspective, Ubuntu 16.04 comes preloaded with the Greengrass core, which includes the lambda runtime, device authentication and authorization, message management, and a whole lot more. There's a lot of power in a very small package. If you've purchased your own DeepLens device, after you unbox it, it needs to be registered and connected to your AWS account. You can follow the instructions found here. So once you've connected the device to your account, you can see your registered device. So here it's showing I have one device registered. AWS DeepLens comes with pre-trained image detection and recognition models to choose from. Or you can build and train your own model and deploy it to the device. For our use case, the pre-trained face detection model works perfectly. So that's what we'll use to detect faces. Now that you've been introduced to DeepLens, let's look at the steps necessary to deploy the pre-trained face detection model to the device.

### **Deploy model to AWS DeepLens**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Let's look at the process to deploy a pretrained face recognition model to our DeepLens device. Before you can deploy a pretrained model you first need to register your device. So you would go to this home page and read the detailed instructions on how to register and configure your DeepLens device and get it connected to your AWS account. Now once you've done that you can go to the AWS Management Console page and go to the DeepLens console. So let's do that. And let's expand this. So this is the DeepLens console. On the left hand side you have the option between Projects, Devices, and Models. So my device is already registered with my account, so I can create a new project by clicking on Create project. And we are going to use a project template and so as you scroll down we want to select the Face detection project. So let's keep going. Click on Next. And this is just describing how the process works. And so whenever a face is detected by the DeepLens device there's a Lambda that's running on the actual device and there's a face detection model that's running on the actual device. And so when a face is detected that output is going to be streamed to what is called the device stream and to the project stream. And I'll tell you more about these in a moment. Let's rename this from Face detection to sagemaker face detect. Then we can leave everything else as-is and let's click Create. So it created this project. It's still in the process of creating, so I'm going to pause the video here and once it's finished I'll start again. So the project is finished creating and now make sure that project is selected and select Deploy to device. You select your device and you click Review. So this is showing you the Lambda function that it's going to create. And here it's tell you, this is just a warning, you will probably not see this, this is just saying that there is an existing project on the device that it's going to overwrite. And then here it's saying that there are deployment costs associated with this. Let's click Deploy. So the project is in the process of being deployed. I will pause the video for now and start back up once it's finished. The project deployed successfully. Let's scroll down and take a look at what's in the project. So here, if you click on Project content, we see the name of the Lambda function that's actually running on the device thanks to the Greengrass Core software. And the Lambda function publishes its output to an MQTT topic. MQTT is how an IoT device, like the DeepLens camera, reports its state back to the cloud. So here, if you double-click and copy this, that's the MQTT topic, you can take that to the IoT core console and then you're able to watch in realtime as messages are published from the DeepLens device. Now let's look at the Lambda code that's running on the device. So click on this and it's going to bring you to the Lambda console. Let's scroll down and expand this. So let's go to some of the more important lines, because there is a lot of code here. So first let's look at line 107. Let's scroll there. So on line 107 we are getting a frame from the video stream, so what the camera actually sees, and we are resizing it. Let's scroll down to line 122. On line 122 we are looking at the objects in the frame in the scene and determining the probabilities that we are looking at a face and if a face is detected we are adding a bounding box and we see that here on lines 124 through 127, we are adding a bounding box around the face. And then scroll down to line 146. This is where it's actually publishing a message to the cloud. We will eventually modify this code to upload the photo of our scene to our chosen S3 bucket. So now let's go back here and talk about the video output streams. So AWS DeepLens produces two output streams, the unprocessed video of what the camera is seeing, it's called the device stream, and then the processed video of what the camera sees with the bounding box around the face, that is called the face detection project stream. So let me show you my setup for AWS DeepLens. So this is my setup. I have my DeepLens device, it is connected to a mouse, a monitor, and a keyboard. So on the monitor we see everything that the AWS DeepLens camera sees. So I'm going to run this quick video for you. And so we're able to see in realtime what the camera sees. So this scene is what will be uploaded to the S3 bucket that will kick off the process to detect crime. Now that you've seen the process to deploy a pretrained face detection model to your AWS DeepLens device let's extend the functionality of the DeepLens device to upload a photo of the scene to our chosen S3 bucket.

### **Extend AWS DeepLens**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Let's look at the process to extend the functionality of the DeepLens device so that it uploads a photo of the captured scene to the Cloud. Let's navigate to S3. The first step is to create the S3 bucket to hold the uploaded photo. I've created a bucket called LinkedIn-DeepLens. And this is the bucket where the photos will be uploaded underneath Images here. I've created the bucket using the default settings for the bucket. I've made the bucket public and I've made sure the bucket exists in the same AWS region as the DeepLens device. So this will hold all of the captured photos. And as a side note, bucket names must be unique globally. Since I've already used this bucket name here, you'll need to choose a different name when setting yours up. Also, if you opted not to purchase a DeepLens device and will manually upload your photo to this S3 bucket, you can stop the video now and skip to the next lesson. The next step is to change the inference lambda running on the device. So let's navigate to our AWS DeepLens project to find the lambda name. So click on Projects. Click on your project. Underneath Project Content, you'll see the function name here. So if you click on this link, it will bring you to the source code for that lambda function. And essentially what we're going to do, we are going to replace all of the code that's found here with the code that's in your exercise folder. And the file name is called DeepLens-S3-upload.py. So copy and paste that code here. Now, I've already done that, so I will walk you through what the code is doing. Overall, we've simply modified the code to upload an image to S3 when a face is detected. In addition, I've added a slight pause in countdown once a face is detected and uploaded so that we don't upload duplicate images containing the same face. So here on line 20, this is very important, I've included the name of my bucket, LinkedIn-DeepLens. You will need to update this with your bucket name. I also want to point out down on line 71 and 72, 71 and 72 here, they contain the cool down timers and the countdown timer, so this prevents us from uploading multiple photos. Next, let's look on line 78. So this is where the frame from the camera is being obtained. The frame is then resized down here on line 84 and evaluated using the model here on line 86. The results of the evaluation are stored in parsed\_results here on line 88. And let's scroll down to line 96. Here on line 96, if there's a high probability that the scene contains a face, a bounding box is placed around the face on lines 98 through 101. And then the photo is uploaded down here on line 119. So this is where the photo is uploaded to S3 only if it's within the cool down period. Now, since we've updated this code, the next step is to save the lambda function and publish a new version. So the way we do that, first, you click Save, and then underneath Actions, click on Publish New Version. And you can give it a description, so say, S3 Upload. And then click Publish. And so now, that makes version four of our lambda. So remember that number, we're going to need it later. Next, let's visit the DeepLens project and update the function on the device to this version four. So go back to DeepLens and this is the correct project and click Edit. And let's scroll down. Underneath Function, we are going to use version four. And then scroll down. We'll leave everything else the same. For timeout, let's change this to 600. And then we are going to save. And then click on this and let's redeploy it to the device. Let's select the device, then click Review. And let's scroll down. And here, this is just giving you a warning, saying you've already deployed a project to the device and it's going to be replaced. And click Deploy. So let's give that a few minutes to deploy. I will pause the video and come back once it's finished. Okay, the project deployment completed successfully. The final step is to SSH into your AWS DeepLens device. So remote login to your device and I will show you how to do that. You will remote login using this SSH command here. You will need to use the IP address for your DeepLens device and then you need to install the botocore software. Boto is the AWS SDK, allowing the lambda we just deployed to the DeepLens camera to access AWS services from Python. This is necessary for the photo upload. So the statement down here shows how to install botocore. Now, let's see this in action. Let's navigate back to S3. When the DeepLens camera detects a face in the scene, the photo is uploaded to S3. Let's refresh this to see if a new photo has come in. Yes, so several photos have come in. So now that you've learned how to extend DeepLens to upload a photo to an S3 bucket, let's retrieve attributes from the photo using AWS Rekognition.

### **Retrieve attributes via AWS Rekognition**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now that we've uploaded a photo of the scene to an S three bucket, the next step is to analyze that photo to pull out attributes about the person. We will use the AWS recognition service to obtain these attributes. The first step is to create a brand-new Lambda function responsible for obtaining the photo from S three and sending it to recognition. So navigate to the AWS Lambda console, click on create function, make sure author from scratch is selected, give your function a name, select the runtime as Python three seven. Underneath permissions, create a new role with basic Lambda permissions and then click create function. Because I've already created my function I'm going to click cancel. And so my function is linked in sage maker. You can find the code for this Lambda in the exercises folder and the file name is linked hyphen sage maker dot PY. This Lambda function is triggered when an object or photo is uploaded to the S three bucket. So you can easily set up a trigger by looking on the left-hand side, finding the trigger that you want. So you would select S three and then you would drag it and drop it over here. And then you would click on it to configure it. So let me show you my configuration. So this trigger says, "Every time an object "is created in that S three bucket "and if there's an item that ends in a JPG, "an image that's loaded to this images folder, "then run this Lambda code." So now let's look at the code. So this is the Lambda code. I do want to show you here on line seven and eight, I'm passing in environment variables. So let's scroll down. There's an environment variable called deep lens device name. So this shows the location of where the camera is installed and then the end point name. This is our sage maker crime-fighting model end point, that URL. So let's scroll back up to expand this to make it bigger. So on line seven and eight I'm grabbing the end point name and the deep lens device name. On line nine, I'm getting a client to the sage maker runtime. On line 10, I'm getting a client for recognition. On line 11, I'm getting a client for S three and at S and S on line 12. So on line 17 and 18, when this code is triggered it's passed an event. And the event contains the bucket name and the object key, which is the image file name. So on lines 21 through 26, I'm retrieving that image from the S three bucket. On line 29, I'm sending that image to the recognition service to retrieve attributes about the person. And notice here I'm calling the detect faces function from the recognition service. On line 32, I'm getting an average age. So in the response from recognition there's face details with an age range of low and then also with an age range of high so it gives an estimated range of how old the service thinks the person is. I'm taking those two values and I'm getting the average age. And then I'm also getting the gender. Notice in the face details response there's a gender value. Notice I'm converting the gender. So I created a function called convert gender. Let's look at that. And convert gender is responsible for taking the response and putting it in a format that a machine can understand. So for example, male is returned from recognition. I'm converting that to a zero and a one. Now, remember from our data file we have a column for gender female and a column for gender male. So I'm converting this to say zero for female and one for male. So now it's in a format that a machine can understand. So let's scroll back up. Here I'm getting the current month using date time now, month. On line 41, I'm getting the current week day. I'm using another conversion method to translate the day of the week to a form that a machine can understand. I'm doing the same thing here on line 44 for current time of day. Now, at this point we have all of the attributes needed in order for our crime-fighting model to make a prediction. We've retrieved age and gender from the photo, location from the name of the camera, and additional attributes like month, day, time of day, from the system timestamp. The next step is to send these attributes to the model. Now that we've learned how to retrieve attributes about a person using AWS recognition, let's look at obtaining a crime prediction from our model.

### **Invoke the crime model**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now that we have all of the attributes needed by our crime fighting model to predict crime, let's send them to our model and obtain a crime prediction. Let's navigate to the AWS Lambda console. This is the code that we looked at in the previous lesson, so let's pick up where we left off. On lines 46 through 51, I provided some very useful comments. These comments show examples of the payload, or the data sent to the model. Notice that the payload consists of numbers that are either one or zero. One stands for yes, or true, and zero stands for false, or no. We also have numbers that indicate the day of the week, the month, and the average age. This payload maps back exactly to the CSV file format that was used to train our model with, minus the target, or the answer. In this case, the target comes back from the model. Down here on line 53, I'm creating the payload by calling this create\_payload function. Let's take a look at this function. Scroll down to line 80. On line 87, I'm transforming the attributes to the format needed by the model. Let's scroll over and look at this convert\_camera\_location. Let's take a look at what that's doing. That's here down on line 91. I have a nested if statement that starts on line 94 that determines the camera names and places a one in that location spot and a zero for every other location. So for example, our camera is called the Surrey\_Camera, so zero for all of those and one for Surrey. Once I've converted the attributes to the necessary format on line 53, I'm sending the payload to the model on line 56. Notice I'm calling this invoke\_endpoint function, and I'm storing the prediction from the model in the response variable. Next on line 60, I'm printing out the response. And then on line 64, I'm printing out the body of the response. And then on line 67, I'm retrieving the prediction from the response, and then I'm printing out the prediction. And then here, on line 71, I'm retrieving the confidence score, and I'm printing that out here on line 72. Down here on line 75, I'm setting the predicted label to Crime if the prediction is a crime and No Crime if not. Now let's see this in action. A photo has been uploaded to s3 which triggers the Lambda to obtain a crime prediction. We can go to CloudWatch to see the output from our Lambda function. So let's navigate there now. This is the CloudWatch dashboard. To get to the logs you simply click on logs and it will bring you to this log groups screen. So scroll down and look for the name of your Lambda function, linked-sagemaker, and let's look at one of the output logs. So we can see the output from the crime prediction on these lines here. Notice this is the payload that was sent to the model. This is the response returned. I print the response out. I also print out the prediction. And so the predicted label we see here is zero, which indicates no crime. The confidence score we see here is 0.27. I also print it out here. So this is saying the model has a 27% confidence score that the scene we're looking at contains the likelihood of crime. So that results in a No Crime prediction. Now that you've seen how to invoke the crime model using the attributes we've obtained, let's notify the user of the crime prediction via email.

### **Set up model alerts**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Now that we have our crime prediction let's alert the user to the potential crime going on in the scene via email using the Amazon Simple Notification Service or SNS. Let's navigate to the SNS console. The first step is to create what's called an SNS topic. So click on Topics from the left hand menu and then click Create topic. Give your topic a name, an optional description, leave the other defaults, and click Create topic. Because I've already created my topic I'm going to click Cancel. So let's look at the topic, sagemaker crime alert. This gives you high level information about the topic and notice the topic has one subscription. So let's talk about that. The next step is to actually subscribe to the topic using an email address. So from the left hand menu click on Subscriptions, and then click Create subscription. You select the topic that was just created, in my case, the sagemaker crime alert, yours will be different. For the Protocol select Email. Notice there's also SMS, that's for text messaging. Select Email. And for the Endpoint enter the email address. And then scroll down and click Create subscription. Now I'm going to click Cancel, because I've already created a subscription. Notice here this message, after your subscription is created you must confirm it. Once you create it a confirmation email is going to be sent and the user of that email has to click on a link to confirm. So let's look at an example email confirmation. So I've pulled up my Microsoft Outlook and notice I have an email that says AWS Notification Subscription Confirmation. So this is an example of what it will look like. It shows the topic ARN and then the user has to click Confirm. Now once that is done the user is setup to receive email. The next step is to publish a message to the topic that was just created. This is done via our Lambda function. So let's navigate to our Lambda code. This is the code that we looked at in the previous lesson, so let's pick up where we left off. So we were here at this line, 76, called publish message. Now one thing you have to do, make sure that this Lambda function, that its execution role has access to the SNS service before publishing a topic or you will see an error message when you try to execute this Lambda. In order to do that you go to IAM and give the Lambda basic execution role or whatever role your Lambda is running under access to SNS via the Amazon SNS Full Access policy. So let's look at this publish message function on line 76. So let's scroll down to line 159 to see the implementation. Here on line 161 I'm calling the publish function, which includes the topic ARN. It also includes a Message. In the Message we include the device name and the prediction returned from the model, along with message format information on lines 164 and 165. So once the crime is predicted using the model this code will also publish a message. And so that message is an email that comes to my Outlook. So let's navigate back to Outlook to see a sample email. So this is what the email will look like. The subject is AWS Notification Message. Notice in the message it has the name of the camera and it says, has detected a person. The prediction is No Crime. So that's an example of the email. And like I showed you, you can also send this as an SMS text message. Now that you've seen how to alert a user with the crime prediction using SNS let's discuss testing the model for fairness.

## **Question 1 of 12**

AWS DeepLens has access to AWS services running in you account.

* TRUE  
  Correct
* FALSE

## **Question 2 of 12**

A photo of the scene is taken and uploaded when what occurs?

* DeepLens is turned on
* DeepLens detects a face  
  Correct
* DeepLens detects an object
* the user clicks a button

## **Question 3 of 12**

What is the unprocessed video stream called?

* Video stream  
  Incorrect
* Output stream  
  Incorrect
* Device stream
* Project stream  
  Incorrect



Replay

Review this video

Deploy model to AWS DeepLens

6m 35s

## **Question 4 of 12**

What is the processed video stream called?

* Output stream
* Device stream
* Project stream  
  Correct
* Video stream

## **Question 5 of 12**

S3 Bucket names must be unique.

* TRUE  
  Correct
* FALSE

## **Question 6 of 12**

Once a face is detected, what is placed around it?

* bounding square
* bounding circle
* bounding matrix
* bounding box  
  Correct

## **Question 7 of 12**

Name one attribute retrieved via Rekognition.

* time of day
* age  
  Correct
* hair color
* race

## **Question 8 of 12**

What has to be setup to kickoff the Lambda?

* Alert
* Trigger  
  Correct
* Notification
* Email

## **Question 9 of 12**

What is the data called that is sent to the endpoint?

* Dataset
* File
* Record
* Payload  
  Correct

## **Question 10 of 12**

Which AWS service allows you to see the output of a program execution?

* Cloud Formation
* Cloud Watch  
  Correct
* Cloud Search
* Cloud Trail

## **Question 11 of 12**

Which step sends a message to the topic?

* Publish  
  Correct
* Test
* Subscribe  
  Incorrect
* Notify  
  Incorrect

## **Question 12 of 12**

A user must subscribe to a topic before receiving a message.

* TRUE  
  Correct
* FALSE

### **What is explainable AI (XAI)?**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Complex machine learning models are often considered to be black boxes, which simply means that no one, even the creator of the model, knows why the model made a certain recommendation or prediction. It just simply can't be explained. Explainable AI, or XAI, attempts to rectify the black box problem with machine learning models. The overall goal of XAI is to produce a model that can explain its rationale behind making certain decisions or predictions and call out its own strengths and weaknesses. XAI assists users of the model with knowing what to expect and how the model might perform. The ability to understand why a model chose one path over another path, and the typical errors a model will make, are a huge advancement in machine learning. This level of transparency and explainability helps to build trust in the predictions or outcomes produced by a model. From a legal perspective, in some countries there are now laws that require models to be explainable. Now these laws haven't made it to the United States just yet, but one day they may. The European Union, or EU, put in place on May 25th, 2018, the General Data Protection Regulation, or GDPR, that includes a right to explanation clause. Because of this clause, citizens of the EU affected by a machine learning model have the right to know how and why a certain decision was made about them. This is huge, especially as our reliance on machine learning continues to grow across the globe. The ability to explain certain decisions may mean the difference between a good model versus a bad model, a model that should be used to make predictions and one that should not. XAI is a fairly new discipline. To learn more about it, I recommend checking out the Learning XAI: Explainable Artificial Intelligence course in our library.

### **Trust and transparency issues**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] There is a common misconception that recommendations made by a machine are fair, accurate and trustworthy since a human is removed from the equation. Unfortunately, this is just not true. We've heard the horror stories of artificial intelligence systems, used for recruiting that were biased against women, or the ones used in courthouses by judges, to determine if someone would be a repeat offender that are biased against minorities. Unfortunately, there is a long list of nightmarish AI stories caused by bias, which results in trust and transparency issues. Bias in machine learning is the systematic favoritism of a given group over another. Oftentimes with bias, there is a vicious cycle where historical data just reinforces existing human bias. This cycle surfaces a lot in predictive policing. Because the data can be based on the decisions made by police officers, that could be based on their own personal biases. For example, a police officer visits a neighborhood he considers to be bad, resulting in a lot of arrests in a particular neighborhood. This teaches the model that that particular neighborhood is bad, which results in the model recommending that police officers visit that neighborhood. I do want to point out that bias is not always bad. For example, a model can't predict cancer, without using statistical selection bias. This type of bias allows a model to select something that's outside of the norm. For example, the model should learn what cancerous tumors look like, and become biased towards selecting those kinds of tumors. While there are some things we want model to learn, there're other things that we don't want a model to learn. For example, we don't want a model to learn what a successful computer programmer looks like, and become biased towards selecting those kinds of people. Statistical bias, the selecting tumors example is different from the latter example, which is referred to as machine bias, or algorithmic bias. Machine bias creeps in by the data used to train the model, simply because the data used to calibrate the model are sometimes inefficient. Machine bias also creeps in by what the developer tells the model is important. Sometimes the algorithms can just be poorly designed. So, how do we ensure that we build models that are fair and trustworthy? As developers, it's important for us to be looking for bias in the first place. When working with attributes that are protected by anti-discrimination laws, such as age, gender, race, et cetera, we have to be very careful. In the machine learning world, these attributes are referred to as protected attributes. Protected attributes can reveal and even uncover bias. Let's use our crime fighting case study as an example. In the original crime data set, race was included. However, in the final data set used to train the crime fighting model race was excluded. In order to avoid the possibility, or the appearance of the model using racial profiling, I removed race from the data set. This strategy to remove race is called demographic parity, and is typically used to ensure model fairness by simply excluding the data points, that could cause issues. Demographic parity calls for the decision making process to be independent of a protected attribute. This approach works sometimes, and sometimes it doesn't. It all depends on your use case. Now that we've learned more about bias and trust and transparency issues, let's discuss the questions that should be answered, so that trust can be built within algorithms.

### **Making algorithms explainable**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] If someone tries to sell you a system for predicting crime, loan defaults, or who would be a great employee, but you don't know what data was used to train the system, and they can't explain how the system makes the decisions it does, then it can't be trusted. There are several factors that go into building an explainable model: Finding comprehensive data, and being transparent about the data. Experimenting with different data sets and metrics, and even editing the data to impose fairness. Increasing representation in your technical workforce. Diverse AI teams tend to build fair systems that are less prone to bias. And using external validity testing and auditing by an unbiased third party. Also providing several details about your model to consumers helps as well. For example, who trained the model? What data were used to train the model? What are the accuracy scores determined through a confusion matrix or the area under curve? A confusion matrix provides a visual representation of the performance of the model showing true positives, true negatives, and errors like false positives, when the model said yes but should have said no, and false negatives, when the model said no but should have said yes. When reviewing the confusion matrix, errors should be evenly distributed. If not, then there might be a problem with bias. Area under curve, or AUC, is a metric used to measure the quality of a binary model. The metric has a range from 0.5 to one. The higher the score, the more accurate the model. And finally, was the model tested for bias? Several big companies like Facebook and Accenture provide tutorials and tools to help less-experienced data scientists and developers identify and remove bias from training data. Some tools are capable of determining whether a machine learning algorithm is biased by analyzing the diversity found in the training data, or by evaluating the quality of a recommendation to see if different people are being impacted differently. The tools are stand-alone, and can easily be integrated into an existing data science process. Big companies like Google and Microsoft have formed AI ethics boards and have teams of people looking into the ethics of AI. We've covered several factors that go into building a model that can be explainable and trusted. So now you can assess other models as well as take steps to ensure that you develop sound models.

## **uestion 1 of 6**

An explainable model can tell you where it may make a mistake.

* TRUE  
  Correct
* FALSE

## **Question 2 of 6**

What helps to build trust in a machine learning model?

* a predefined decision tree  
  Incorrect
* a black box solution  
  Incorrect
* an error count  
  Incorrect
* transparency



Replay

Review this video

What is explainable AI (XAI)?

2m 23s

## **Question 3 of 6**

Which type of bias is considered "good" bias?

* statistical selection bias  
  Correct
* algorithmic bias
* machine bias
* gender bias

## **Question 4 of 6**

A machine always makes fair recommendations since humans are removed from the equation.

* TRUE
* FALSE  
  Correct

## **Question 5 of 6**

What is one factor that can help a model be trusted?

* ensuring only one type of person builds, trains, and tests the model
* editing the data to impose fairness  
  Correct
* testing the model with limited data
* ensuring all the data is the same

## **Question 6 of 6**

The lower the Area Under Curve (AUC) the better the binary model.

* TRUE
* FALSE

### **Next steps**

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] This course gave you a great foundation for building fair, transparent, and explainable machine learning models using Amazon SageMaker. I enjoyed taking you through the machine learning process to inspect and visualize data, prepare the data, and train and deploy the model. I hope you enjoyed using our fun crime fighting model to predict crime and integrating it with the AWS DeepLens camera and Rekognition allowing the model to see and understand its surrounding before making a prediction. When building models in the future, it's important that you consider the ways bias can creep in and eliminate those. As developers and innovators using machine learning technology, we have moral and ethical obligations to ensure what we build produces results and recommendations that are fair. To further your learning, I recommend checking out other courses in the library about AWS, machine learning, and explainable AI. Good luck on your machine learning journey and if you have any questions about the course, please feel free to reach out to me, Kesha Williams, on LinkedIn. See you next time.