Machine Learning Engineer Learning Path

A Machine Learning Engineer designs, builds, productionizes, optimizes, operates, and maintains ML systems.

15 activities updated 9 months Managed by Google Cloud

A Machine Learning Engineer designs, builds, productionizes, optimizes, operates, and maintains ML systems. This learning path guides you through a curated collection of on-demand courses, labs, and skill badges that provide you with real-world, hands-on experience using Google Cloud technologies essential to the ML Engineer role. Once you complete the path, check out the Google Cloud Machine Learning Engineer certification to take the next steps in your professional journey.

2 Google Cloud Big Data and Machine Learning Fundamentals

1 Big Data and Machine Learning on Google Cloud

Introduction

Welcome to the first section of the Big Data and Machine Learning course. Here you'll explore the Google infrastructure through compute and storage and see how innovation has enabled big data and machine learning capabilities After that, you'll see the history of big data and ML products, which will help you understand the relevant product categories. And to put it all together, you'll examine an example of a customer who adopted Google Cloud for their big data and machine learning needs. Finally, you'll get hands-on practice using big data tools to analyze a public dataset. Google has been working with data and artificial intelligence since its early days as a company in 1998. Ten years later in 2008 the Google Cloud Platform was launched to provide secure and flexible cloud computing and storage services. You can think of the Google Cloud infrastructure in terms of three layers. At the base layer is networking and security, which lays the foundation to support all of Google's infrastructure and applications. On the next layer sit compute and storage. Google Cloud separates, or decouples, as it's technically called, compute and storage so they can scale independently based on need. And on the top layer sit the big data and machine learning products, which enable you to perform tasks to ingest, store, process, and deliver business insights, data pipelines, and ML models. And thanks to Google Cloud, these tasks can be accomplished without needing to manage and scale the underlying infrastructure. In the videos that follow, we'll focus on the middle layer, compute and storage, and the top layer, big data, and machine learning products. Networking and security fall outside of the focus of this course, but if you're interested in learning more you can explore cloud.google.com/training for more options.

Google Cloud infrastructure

Google Cloud's infrastructure is based in five major geographic location North America, South America, Europe, Asia, and Australia. Having multiple service locations is important because choosing where to locate applications affects qualities like availability, durability, and latency, which measures the time a packet of information takes to travel from its source to its destination. Each of these locations is divided into several different regions and zones. Regions represent independent geographic areas and are composed of zones. For example, London, or europe-west2, is a region that currently contains three different zones. A zone is an area where Google Cloud resources are deployed. For example, let's say you launch a virtual machine using Compute Engine–more about Compute Engine in a bit–it will run in the zone that you specify to ensure resource redundancy. Zonal resources operate within a single zone, which means that if a zone becomes unavailable, the resources won't be available either. Google Cloud lets users specify the geographical locations to run services and resources. In many cases, you can even specify the location on a zonal, regional, or multi-regional level. This is useful for bringing applications closer to users around the world, and also for protection in case there are issues with an entire region, say, due to a natural disaster. A few of Google Cloud's services support placing resources in what we call a multi-region. For example, Cloud Spanner multi-region configurations allow you to replicate the database's data not just in multiple zones, but in multiple zones across multiple regions, as defined by the instance configuration. These additional replicas enable you to read data with low latency from multiple locations close to or within the regions in the configuration, like The Netherlands and Belgium. Google Cloud currently supports 103 zones in 34 regions, though this is increasing all the time. The most up to date info can be found at cloud.google.com/about/locations.

Compute

Let's focus our attention on the middle layer of the Google Cloud infrastructure, compute, and storage. We'll begin with compute. Organizations with growing data needs often require lots of compute power to run big data jobs. And as organizations design for the future, the need for compute power only grows. Google offers a range of computing services. The first is Compute Engine. Compute Engine is an IaaS offering, or infrastructure as a service, which provides compute, storage, and network resources virtually that are similar to physical data centers. You use the virtual compute and storage resources the same as you manage them locally. Compute Engine provides maximum flexibility for those who prefer to manage server instances themselves. The second is Google Kubernetes Engine, or GKE runs containerized applications in a cloud environment, as opposed to on an individual virtual machine, like Compute Engine. A container represents code packaged up with all its dependencies. The third computing service offered by Google is App Engine, a fully managed PaaS offering, or platform as a service. PaaS offerings bind code to libraries that provide access to the infrastructure application needs. This allows more resources to be focused on application logic. Then there is Cloud Functions, which executes code in response to events, like when a new file is uploaded to Cloud Storage. It's a completely serverless execution environment, which means you don't need to install any software locally to run the code and you are free from provisioning and managing servers. Cloud Functions is often referred to as functions as a service. And finally, there is Cloud Run, a fully managed compute platform that enables you to run request or event-driven stateless workloads without having to worry about servers. It abstracts away all infrastructure management so you can focus on writing code. It automatically scales up and down from zero, so you never have to worry about scale configuration. Cloud Run charges you only for the resources you use so you never pay for over provisioned resources. Let's understand how Google Photos relies on the compute capability provided by Google Cloud to implement their video stabilization Google Photos offers a feature called automatic video stabilization. This takes an unstable video, like one captured while riding on the back of a motorbike and stabilizes it to minimize movement. Let's look at an example of a technology that requires a lot of compute power. For video stability to work as intended, you need the proper data. This includes the video itself, which is really a large collection of individual images, along with time series data on the camera's position and orientation from the onboard gyroscope, and motion from the camera lens. A short video can require over a billion data points to feed the ML model to create a stabilized version. As of 2020, roughly 28 billion photos and videos were uploaded to Google Photos every week, with more than four trillion photos in total stored in the service. To ensure that this feature works as intended, and accurately, the Google Photos team needed to develop, train, and serve a high-performing machine learning model on millions of videos. That's a large training dataset! Just as the hardware on a standard personal computer might not be powerful enough to process a big data job for an organization, the hardware on a smartphone is not powerful enough to train sophisticated ML models. That's why Google trains production machine learning models on a vast network of data centers, only to then deploy smaller, trained versions of the models to the smartphone and personal computer hardware But where does all that processing power come from? According to Stanford University's 2019 AI index report, before 2012, artificial intelligence results tracked closely with Moore's Law, with the required computing power used in the largest AI training runs doubling every two years. The report states that, since 2012, the required computing power has been doubling approximately every three and a half months. This means that hardware manufacturers have run up against limitations, and CPUs, which are central processing units, and GPUs, which are graphics processing units, can no longer scale to adequately reach the rapid demand for ML. To help overcome this challenge, in 2016 Google introduced the Tensor Processing Unit, or TPU. TPUs are Google's custom-developed application-specific integrated circuits (ASICs) used to accelerate machine learning workloads. TPUs act as domain-specific hardware, as opposed to general-purpose hardware with CPUs and GPUs. This allows for higher efficiency by tailoring architecture to meet the computation needs in a domain, such as the matrix multiplication in machine learning. TPUs are generally faster than current GPUs and CPUs for AI applications and machine learning. They are also significantly more energy-efficient. Cloud TPUs have been integrated across Google products, making this state-of-the-art hardware and supercomputing technology available to Google Cloud customers.

Storage

Now that we've explored compute and why it's needed for big data and ML jobs, let's now examine storage. For proper scaling capabilities, compute and storage are decoupled. This is one of the major differences between cloud and desktop computing. With cloud computing, processing limitations aren't attached to storage disks. Most applications require a database and storage solution of some kind. With Compute Engine, for example, which was mentioned in the previous video, you can install and run a database on a virtual machine, just as you would do in a data center. Alternatively, Google Cloud offers fully managed database and storage services. These include Cloud Storage Cloud Bigtable Cloud SQL Cloud Spanner Firestore And BigQuery The goal of these products is to reduce the time and effort needed to store data. This means creating an elastic storage bucket directly in a web interface or through a command line for example on Cloud Storage. Google Cloud offers relational and non-relational databases, and worldwide object storage. We'll explore those options in more detail soon. Choosing the right option to store and process data often depends on the data type that needs to be stored and the business need. Let's start with unstructured versus structured data. Unstructured data is information stored in a non-tabular form such as documents, images, and audio files. Unstructured data is usually suited to Cloud Storage, but BigQuery now offers the capability to store unstructured data as well. Cloud Storage is a managed service for storing unstructured data. Cloud Storage is a service for storing your objects in Google Cloud. An object is an immutable piece of data consisting of a file of any format. You store objects in containers called buckets. All buckets are associated with a project, and you can group your projects under an organization. Each project, bucket, and object in Google Cloud is a resource in Google Cloud, as are things such as Compute Engine instances. After you create a project, you can create Cloud Storage buckets, upload objects to your buckets, and download objects from your buckets. A few examples include serving website content, storing data for archival and disaster recovery, and distributing large data objects to end users via Direct Download. Cloud Storage has four primary storage classes. The first is Standard Storage. Standard Storage is considered best for frequently accessed, or "hot," data. It's also great for data that is stored for only brief periods of time. The second storage class is Nearline Storage. This is best for storing infrequently accessed data, like reading or modifying data once per month or less, on average. Examples include data backups, long-tail multimedia content, or data archiving. The third storage class is Coldline Storage. This is also a low-cost option for storing infrequently accessed data. However, as compared to Nearline Storage, Coldline Storage is meant for reading or modifying data, at most, once every 90 days. The fourth storage class is Archive Storage. This is the lowest-cost option, used ideally for data archiving, online backup, and disaster recovery. It's the best choice for data that you plan to access less than once a year, because it has higher costs for data access and operations and a 365-day minimum storage duration. Alternatively, there is structured data, which represents information stored in tables, rows, and columns. Structured data comes in two type transactional workloads and analytical workloads. Transactional workloads stem from Online Transaction Processing systems, which are used when fast data inserts and updates are required to build row-based records. This is usually to maintain a system snapshot. They require relatively standardized queries that impact only a few records. Then there are analytical workloads, which stem from Online Analytical Processing systems, which are used when entire datasets need to be read. They often require complex queries, for example, aggregations. Once you've determined if the workloads are transactional or analytical, you'll need to identify whether the data will be accessed using SQL or not. So, if your data is transactional and you need to access it using SQL, then Cloud SQL and Cloud Spanner are two options. Cloud SQL works best for local to regional scalability, while Cloud Spanner, it best to scale a database globally. If the transactional data will be accessed without SQL, Firestore might be the best option. Firestore is a transactional NoSQL, document-oriented database. If you have analytical workloads that require SQL commands, BigQuery is likely the best option. BigQuery, Google's data warehouse solution, lets you analyze petabyte-scale datasets. Alternatively, Cloud Bigtable provides a scalable NoSQL solution for analytical workloads. It's best for real-time, high-throughput applications that require only millisecond latency.

The history of big data and ML products

The final layer of the Google Cloud infrastructure that is left to explore is big data and machine learning products. In this video, we'll examine the evolution of data processing frameworks through the lens of product development. Understanding the chronology of products can help address typical big data and ML challenges. Historically speaking, Google experienced challenges related to big data quite early– mostly with large datasets, fast-changing data, and varied data. This was the result of needing to index the World Wide Web And as the internet grew, Google needed to invent new data processing methods. So, in 2002, Google released the Google File System, or GFS. GFS was designed to handle data sharing and petabyte storage at scale. It served as the foundation for Cloud Storage and also what would become the managed storage functionality in BigQuery. A challenge that Google was facing around this time was how to index the exploding volume of content on the web. To solve this, in 2004 Google wrote a report that introduced MapReduce. MapReduce was a new style of data processing designed to manage large-scale data processing across big clusters of commodity servers. As Google continued to grow, new challenges arose, specifically with recording and retrieving millions of streaming user actions with high throughput. The solution was the release in 2005 of Cloud Bigtable, a high-performance NoSQL database service for large analytical and operational workloads. With MapReduce available, some developers were restricted by the need to write code to manage their infrastructure, which prevented them from focusing on application logic. As a result, from 2008 to 2010, Google started to move away from MapReduce as the solution to process and query large datasets. So, in 2008, Dremel was introduced. Dremel took a new approach to big-data processing by breaking the data into smaller chunks called shards, and then compressing them. Dremel then uses a query optimizer to share tasks between the many shards of data and the Google data centers, which processed queries and delivered results. The big innovation was that Dremel auto scaled to meet query demands. Dremel became the query engine behind BigQuery. Google continued innovating to solve big data and machine learning challenges. Some of the technology solutions released include colossus, in 2010, which is a cluster-level file system and successor to the Google File System. BigQuery, in 2010 as well, which is a fully-managed, serverless data warehouse that enables scalable analysis over petabytes of data. It is a Platform as a Service (PaaS) that supports querying using ANSI SQL. It also has built-in machine learning capabilities. BigQuery was announced in May 2010 and made generally available in November 2011. Spanner, in 2012, which is a globally available and scalable relational database. Pub/Sub, in 2015, which is a service used for streaming analytics and data integration pipelines to ingest and distribute data. And TensorFlow, also in 2015, which is a free and open-source software library for machine learning and artificial intelligence. 2018 brought the release of the Tensor Processing Unit, or TPU, which you'll recall from earlier, and AutoML, as a suite of machine learning products. The list goes on till Vertex AI, a unified ML platform released in 2021. And it's thanks to these technologies that the big data and machine learning product line is now robust. This include Cloud Storage Dataproc Cloud Bigtable BigQuery Dataflow Firestore Pub/Sub Looker Cloud Spanner AutoML, and Vertex AI, the unified platform These products and services are made available through Google Cloud, and you'll get hands-on practice with some of them as part of this course. As we explored in the last video, Google offers a range of big data and machine learning products. So, how do you know which is best for your business needs? Let's look closer at the list of products, which can be divided into four general categories along the data-to-AI workflow ingestion and process, storage, analytics, and machine learning. Understanding these product categories can help narrow down your choice. The first category is ingestion and process, which include products that are used to digest both real-time and batch data. The list includes Pub/Sub Dataflow Dataproc Cloud Data Fusion You'll explore how Dataflow and Pub/Sub can ingest streaming data later in this course. The second product category is data storage, and you'll recall from earlier that there are five storage product Cloud Storage Cloud SQL Cloud Spanner Cloud Bigtable, and Firestore Cloud SQL and Cloud Spanner are relational databases, while Bigtable and Firestore are NoSQL databases. The third product category is analytics. The major analytics tool is BigQuery. BigQuery is a fully managed data warehouse that can be used to analyze data through SQL commands. In addition to BigQuery, you can analyze data and visualize results using Google Data Studio, and Looker You will explore BigQuery, Looker, and Data Studio in this course. And the final product category is machine learning, or ML. ML products include both the ML development platform and the AI solution the primary product of the ML development platform is Vertex AI, which includes the products and technologic AutoML Vertex AI Workbench, and TensorFlow AI solutions are built on the ML development platform and include state-of-the-art products to meet both horizontal and vertical market needs. These include document AI Contact Center AI Retail Product Discovery, and Healthcare Data Engine These products unlock insights that only large amounts of data can provide. We'll explore the machine learning options and workflow together with these products in greater detail later.

Big data and ML product categories

As we explored in the last video, Google offers a range of big data and machine learning products. So, how do you know which is best for your business needs? Let's look closer at the list of products, which can be divided into four general categories along the data-to-AI workflow ingestion and process, storage, analytics, and machine learning. Understanding these product categories can help narrow down your choice. The first category is ingestion and process, which include products that are used to digest both real-time and batch data. The list includes Pub/Sub Dataflow Dataproc Cloud Data Fusion You'll explore how Dataflow and Pub/Sub can ingest streaming data later in this course. The second product category is data storage, and you'll recall from earlier that there are five storage product Cloud Storage Cloud SQL Cloud Spanner Cloud Bigtable, and Firestore Cloud SQL and Cloud Spanner are relational databases, while Bigtable and Firestore are NoSQL databases. The third product category is analytics. The major analytics tool is BigQuery. BigQuery is a fully managed data warehouse that can be used to analyze data through SQL commands. In addition to BigQuery, you can analyze data and visualize results using Looker, and Looker Studio. You will explore BigQuery, Looker, and Looker Studio in this course. And the final product category is machine learning, or ML. ML products include both the ML development platform and the AI solution he primary product of the ML development platform is Vertex AI, which includes the products and technologic AutoML Vertex AI Workbench, and TensorFlow AI solutions are built on the ML development platform and include state-of-the-art products to meet both horizontal and vertical market needs. These include document AI Contact Center AI Retail Product Discovery, and Healthcare Data Engine These products unlock insights that only large amounts of data can provide. We'll explore the machine learning options and workflow together with these products in greater detail later.

Customer example Gojek

With many big data and machine learning products options available, it can be helpful to see an example of how an organization has leveraged Google Cloud to meet their goals. In this video, you'll learn about a company called Gojek and how they were able to find success through Google Cloud's data engineering and machine learning offerings. This story starts in Jakarta, Indonesia. Traffic congestion is a fact of life for most Indonesian residents. To minimize delays, many rely heavily on motorcycles, including motorcycle taxis, known as Ojeks, to travel to and from work or personal engagements. Founded in 2010 and headquartered in Jakarta, a company called Gojek started as a call center for ojek bookings. The organization has leveraged demand for the service to become one of the few "unicorns" in Southeast Asia. A "unicorn" is a privately held startup business valued at over US$1 billion. Since its inception, Gojek has collected data to understand customer behavior, and in 2015 launched a mobile application that bundled ride-hailing, food delivery, and grocery shopping. They hit hypergrowth very quickly. According to the Q2 2021 Gojek fact sheet the Gojek app has been downloaded over 190 million times, and they have 2 million driver partners and about 900,000 merchant partners. The business has relied heavily on the skills and expertise of its technology team and on selecting the right technologies to grow and to expand into new markets. Gojek chose to run its applications and data in Google Cloud. Gojek's goal is to match the right driver with the right request as quickly as possible. In the early days of the app, a driver would be pinged every 10 seconds, which meant 6 million pings per minute, which turned out to be 8 billion pings per day across their driver partners. They generated around five terabytes of data each day. Leveraging information from this data was vital to meeting their company goals. But Gojek faced challenges along the way. Let's explore two of them to see how Google Cloud was able to solve them.

The first challenge was data latency. When they wanted to scale their big data platform, they found that most reports were produced one day later, so they couldn't identify problems immediately. To help solve this, Gojek migrated their data pipelines to Google Cloud. The team started using Dataflow for streaming data processing and BigQuery for real-time business insights. Another challenge was quickly determining which location had too many, or too few, drivers to meet demand. Gojek was able to use Dataflow to build a streaming event data pipeline. This let driver locations ping Pub/Sub every 30 seconds, and Dataflow would process the data. The pipeline would aggregate the supply pings from the drivers against the booking requests. This would connect to Gojek's notification system to alert drivers where they should go. This process required a system that was able to scale up to handle times of high throughput and then back down again. Dataflow was able to automatically manage the number of workers processing the pipeline to meet demand. The Gojek team was also able to visualize and identify supply and demand issues. They discovered that the areas with the highest discrepancy between supply and demand came from train stations. Often there were far more booking requests than there were available drivers. Since using Google Cloud's big data and machine learning products, the Gojek team has been able actively monitor requests to ensure that drivers are in the areas with the highest demand. This brings faster bookings for riders and more work for the drivers. Now it's time for you to take a break from course videos and get hands-on practice with one of the big data and machine learning products that was introduced earlier– BigQuery. In the lab that follows this video, you'll use BigQuery to explore a public dataset. You'll practice querying a public data set Creating a custom table Loading data into a table and Querying a table. Please note that this exercise involves leaving the current learning platform and opening Qwiklabs. Qwiklabs offers a free, clean, Google Cloud environment for a fixed period of time. You'll have multiple attempts at each lab, so if you don't complete it the first time, or if you want to experiment more with it later on, you can return and start a new instance. This brings us to the end of the first section of the Big Data and Machine Learning course. Before we move forward, let's review what we've covered so far. You began by exploring the Google Cloud infrastructure through three different layers. At the base layer is networking and security, which makes up the foundation to support all of Google's infrastructure and applications. On the next layer sit compute and storage. Google Cloud decouples compute and storage so they can scale independently based on need. And on the top layer sit the big data and machine learning products. In the next section, you learned about the history of big data and ML technologies, and then explored the four major product categories that support the data to AI workflow in the next section, you learned about the history of big data and ML technologies, and then explored the four major product categories that support the data to AI workflow ingestion and process, storage, analytics, and machine learning. After that, you saw an example of how Gojek, the Indonesian on-demand multi-service platform and digital payment technology group, leveraged Google Cloud big data and ML products to expand their business. And finally, you got hands-on practice with BigQuery by analyzing a public dataset.

Summary

This brings us to the end of the first section of the Big Data and Machine Learning course. Before we move forward, let's review what we've covered so far. You began by exploring the Google Cloud infrastructure through three different layers. At the base layer is networking and security, which makes up the foundation to support all of Google's infrastructure and applications. On the next layer sit compute and storage. Google Cloud decouples compute and storage so they can scale independently based on need. And on the top layer sit the big data and machine learning products. In the next section, you learned about the history of big data and ML technologies, and then explored the four major product categories that support the data to AI workflow ingestion and process, storage, analytics, and machine learning. After that, you saw an example of how Gojek, the Indonesian on-demand multi-service platform and digital payment technology group, leveraged Google Cloud big data and ML products to expand their business. And finally, you got hands-on practice with BigQuery by analyzing a public dataset.

2 Data Engineering with Streaming Data

Introduction

In the previous section of this course, you learned about the different layers of the Google Cloud infrastructure, including the categories of big data and machine learning products. In this second section, you'll explore data engineering for streaming data with the goal of building a real-time data solution with Google Cloud products and services. This includes how ingest streaming data using Pub/Sub Process the data with Dataflow and Visualize the results with Looker and Looker Studio. In between data processing with Dataflow and visualization with Looker or Looker Studio, the data is normally saved and analyzed in a data warehouse such as BigQuery. You will learn the details about BigQuery in a later module. Coming up in this section, you'll start by examining some of the big data challenges faced by today's data engineers when setting up and managing pipelines. Next, you'll learn about message-oriented architecture. This includes ways to capture streaming messages globally, reliably, and at scale so they can be fed into a pipeline. From there, you'll see how to design streaming pipelines with Apache Beam, and then implement them with Dataflow. You'll explore how to visualize data insights on a dashboard with Looker and Looker Studio. And finally, you'll get hands-on practice building an end-to-end data pipeline that handles real-time data ingestion with Pub/Sub, processing with Dataflow, and visualization with Looker Studio. Before we get too far, let's take a moment to explain what streaming data is, how it differs from batch processing, and why it's important. Batch processing is when the processing and analysis happens on a set of stored data. An example is Payroll and billing systems that have to be processed on either a weekly or monthly basis. Streaming data is a flow of data records generated by various data sources. The processing of streaming data happens as the data flows through a system. This results in the analysis and reporting of events as they happen. An example would be fraud detection or intrusion detection. Streaming data processing means that the data is analyzed in near real-time and that actions will be taken on the data as quickly as possible. Modern data processing has progressed from legacy batch processing of data toward working with real-time data streams. An example of this is streaming music and movies. No longer is it necessary to download an entire movie or album to a local device. Data streams are a key part in the world of big data.

Big data challenges

Building scalable and reliable pipelines is a core responsibility of data engineers. However, in modern organizations, data engineers and data scientists are facing four major challenges. These are collectively known as the 4 Vs. They are variety, volume, velocity, and veracity. First, data could come in from a variety of different sources and in various formats. Imagine hundreds of thousands of sensors for self-driving cars on roads around the world. The data is returned in various formats such as number, image, or even audio. Now consider point-of-sale data from a thousand different stores. How do we alert our downstream systems of new transactions in an organized way with no duplicates? Next, let's increase the magnitude of the challenge to handle not only an arbitrary variety of input sources, but a volume of data that varies from gigabytes to petabytes. You'll need to know whether your pipeline code and infrastructure can scale with those changes or whether it will grind to a halt or even crash. The third challenge concerns velocity. Data often needs to be processed in near-real time as soon as it reaches the system. You'll probably also need a way to handle data that arrives late, has bad data in the message, or needs to be transformed mid-flight before it is streamed into a data warehouse. And the fourth major challenge is veracity, which refers to the data quality. Because big data involves a multitude of data dimensions resulting from different data types and sources, there's a possibility that gathered data will come with some inconsistencies and uncertainties. Challenges like these are common considerations for pipeline developers. By the end of this section, the goal is for you to better understand the tools available to help successfully build a streaming data pipeline and avoid these challenges.

Message-oriented architecture

One of the early stages in a data pipeline is data ingestion, which is where large amounts of streaming data are received. Data, however, may not always come from a single, structured database. Instead, the data might stream from a thousand, or even a million, different events that are all happening asynchronously. A common example of this is data from IoT, or Internet of Things, applications. These can include sensors on taxis that send out location data every 30 seconds or temperature sensors around a data center to help optimize heating and cooling. These IoT devices present new challenges to data ingestion, which can be summarized in four point the first is that data can be streamed from many different methods and devices, many of which might not talk to each other and might be sending bad or delayed data. The second is that it can be hard to distribute event messages to the right subscribers. Event messages are notifications. A method is needed to collect the streaming messages that come from IoT sensors and broadcast them to the subscribers as needed. The third is that data can arrive quickly and at high volumes. Services must be able to support this. And the fourth challenge is ensuring services are reliable, secure, and perform as expected. Google Cloud has a tool to handle distributed message-oriented architectures at scale, and that is Pub/Sub. The name is short for Publisher/Subscriber or publish messages to subscribers. Pub/Sub is a distributed messaging service that can receive messages from a variety of device streams such as gaming events, IoT devices, and application streams. It ensures at-least-once delivery of received messages to subscribing applications, with no provisioning required. Pub/Sub's APIs are open, the service is global by default, and it offers end-to-end encryption. Let's explore the end-to-end architecture using Pub/Sub. Upstream source data comes in from devices all over the globe and is ingested into Pub/Sub, which is the first point of contact within the system. Pub/Sub reads, stores, broadcasts to any subscribers of this data topic that new messages are available. As a subscriber of Pub/Sub, Dataflow can ingest and transform those messages in an elastic streaming pipeline and output the results into an analytics data warehouse like BigQuery. Finally, you can connect a data visualization tool, like Looker, to visualize and monitor the results of a pipeline, or an AI or ML tool such as Vertex AI to explore the data to uncover business insights or help with predictions A central element of Pub/Sub is the topic. You can think of a topic like a radio antenna. Whether your radio is playing music, or it's turned off, the antenna itself is always there. If music is being broadcast on a frequency that nobody's listening to, the stream of music still exists. Similarly, a publisher can send data to a topic that has no subscriber to receive it. Or a subscriber can be waiting for data from a topic that isn't getting data sent to it, like listening to static from a bad radio frequency. Or you could have a fully operational pipeline where the publisher is sending data to a topic that an application is subscribed to. That means there can be zero, one, or more publishers, and zero, one or more subscribers related to a topic. And they're completely decoupled, so they're free to break without affecting their counterparts. It's helpful to describe this using an example. Say you've got a human resources topic. A new employee joins your company, and several applications across the company need to be updated. Adding a new employee can be an event that generates a notification to the other applications that are subscribed to the topic, and they'll receive the message about the new employee starting. Now, let's assume that there are two different types of employee full-time employee and a contractor. Both sources of employee data could have no knowledge of the other but still publish their events saying "this employee joined" into the Pub/Sub HR topic. After Pub/Sub receives the message, downstream applications like the directory service, facilities system, account provisioning, and badge activation systems can all listen and process their own next steps independent of one another. Pub/Sub is a good solution to buffer changes for lightly coupled architectures, like this one, that have many different publishers and subscribers. Pub/Sub supports many different inputs and outputs, and you can even publish a Pub/Sub event from one topic to another. The next task is to get these messages reliably into our data warehouse, and we'll need a pipeline that can match Pub/Sub's scale and elasticity to do it.

Designing streaming pipelines with Apache Beam

After messages have been captured from the streaming input sources you need a way to pipe that data into a data warehouse for analysis. This is where Dataflow comes in. Dataflow creates a pipeline to process both streaming data and batch data. "Process" in this case refers to the steps to extract, transform, and load data, or ETL. When building a data pipeline, data engineers often encounter challenges related to coding the pipeline design and implementing and serving the pipeline at scale. During the pipeline design phase, there are a few questions to consider will the pipeline code be compatible with both batch and streaming data, or will it need to be refactored? Will the pipeline code software development kit, or SDK, being used have all the transformations, mid-flight aggregations and windowing and be able to handle late data? Are there existing templates or solutions that should be referenced? A popular solution for pipeline design is Apache Beam. It's an open source, unified programming model to define and execute data processing pipelines, including ETL, batch, and stream processing. Apache Beam is unified, which means it uses a single programming model for both batch and streaming data. It's portable, which means it can work on multiple execution environments, like Dataflow and Apache Spark, among others. And it's extensible, which means it allows you to write and share your own connectors and transformation libraries. Apache Beam provides pipeline templates, so you don't need to build a pipeline from nothing. And it can write pipelines in Java, Python, or Go. The Apache Beam software development kit, or SDK, is a collection of software development tools in one installable package. It provides a variety of libraries for transformations and data connectors to sources and sinks. Apache Beam creates a model representation from your code that is portable across many runners. Runners pass off your model for execution on a variety of different possible engines, with Dataflow being a popular choice.

Implementing streaming pipelines on Cloud Dataflow

As covered in the previous video, Apache Beam can be used to create data processing pipelines. The next step is to identify an execution engine to implement those pipelines. When choosing an execution engine for your pipeline code, it might be helpful to consider the following question how much maintenance overhead is involved? Is the infrastructure reliable? How is the pipeline scaling handled? How can the pipeline be monitored? Is the pipeline locked into a specific service provider? This brings us to Dataflow. Dataflow is a fully managed service for executing Apache Beam pipelines within the Google Cloud ecosystem. Dataflow handles much of the complexity relating to infrastructure setup and maintenance and is built on Google's infrastructure. This allows for reliable auto scaling to meet data pipeline demands. Dataflow is serverless and NoOps, which means No Operations. But what does that mean exactly? A NoOps environment is one that doesn't require management from an operations team, because maintenance, monitoring, and scaling are automated. Serverless computing is a cloud computing execution model. This is when Google Cloud, for example, manages infrastructure tasks on behalf of the users. This includes tasks like resource provisioning, performance tuning, and ensuring pipeline reliability. Dataflow means that you can spend more time analyzing the insights from your datasets and less time provisioning resources to ensure that your pipeline will successfully complete its next cycles. It's designed to be low maintenance. Let's explore the tasks Dataflow performs when a job is received. It starts by optimizing a pipeline model's execution graph to remove any inefficiencies. Next, it schedules out distributed work to new workers and scales as needed. After that, it auto-heals any worker faults. From there, it automatically rebalances efforts to use its workers most efficiently. And finally, it outputs data to produce a result. BigQuery is one of many options that data can be outputted to. You'll get some more practice using BigQuery later in this course. So by design, you don't need to monitor all of the compute and storage resources that Dataflow manages, to fit the demand of a streaming data pipeline. Even experienced Java or Python developers will benefit from using Dataflow templates, which cover common use cases across Google Cloud products. The list of templates is continuously growing. They can be broken down into three category treaming templates, batch templates, and utility templates. Streaming templates are for processing continuous, or real-time, data. For example, Pub/Sub to BigQuery Pub/Sub to Cloud Storage Datastream to BigQuery Pub/Sub to MongoDB Batch templates are for processing bulk data, or batch load data. For example, BigQuery to Cloud Storage Bigtable to Cloud Storage Cloud Storage to BigQuery Cloud Spanner to Cloud Storage Finally, utility templates address activities related to bulk compression, deletion, and conversion. For a complete list of templates, please refer to the reading list.

Visualization with Looker

Telling a good story with data through a dashboard can be critical to the success of a data pipeline because data that is difficult to interpret or draw insights from might be useless. After data is in BigQuery a lot of skill and effort can still be required to uncover insights. To help create an environment where stakeholders can easily interact with and visualize data, Google Cloud offers two solutions Looker and Looker Studio. Let's explore both of them, starting with Looker. Looker supports BigQuery, as well as more than 60 different SQL databases. It allows developers to define a semantic modeling layer on top of databases using Looker Modeling Language, or LookML. LookML defines logic and permissions independent from a specific database or a SQL language, which frees a data engineer from interacting with individual databases to focus more on business logic across an organization. The Looker platform is 100% web-based, which makes it easy to integrate into existing workflows and share with multiple teams at an organization. There is also a Looker API, which can be used to embed Looker reports in other applications. Let's explore some of Looker's features, starting with dashboards. Dashboards, like the Business Pulse dashboard, for example, can visualize data in a way that makes insights easy to understand. For a sales organization, it shows figures that many might want to see at the start of the week, like the number of new users acquired, monthly sales trends, and even the number of year-to-date orders. Information like this can help align teams, identify customer frustrations, and maybe even uncover lost revenue. Based on the metrics that are important to your business, you can create Looker dashboards that provide straightforward presentations to help you and your colleagues quickly see a high-level business status. Looker has multiple data visualization options, including area charts, line charts, Sankey diagrams, funnels, and liquid fill gauges. To share a dashboard with your team, you schedule delivery through storage services like Google Drive Slack Dropbox. Let's explore another Looker dashboard, this time one that monitors key metrics related to New York City taxis over a period of time. This dashboard display Total revenue * Total numbers of passengers, and * Total number of rides Looker displays this information through a timeseries to help monitor metrics over time. Looker also lets you plot data on a map to see ride distribution, busy areas, and peak hours. The purpose of these features is to help you draw insights to make business decisions. For more training on Looker, please refer to cloud.google.com/training.

Visualization with Data Studio

Another popular data visualization tool offered by Google is Looker Studio. Looker Studio is integrated into BigQuery, which makes data visualization possible with just a few clicks. This means that leveraging Looker Studio doesn't require support from an administrator to establish a data connection, which is a requirement with Looker. Looker Studio dashboards are widely used across many Google products and applications. For example, Looker Studio is integrated into Google Analytics to help visualize, in this case, a summary of a marketing website. This dashboard visualizes the total number of visitors through a map, compares month-over-month trends, and even displays visitor distribution by age. Another Looker Studio integration is the Google Cloud billing dashboard. You might be familiar with this from your account. Maybe you've already used it to monitor spending, for example. You'll soon have hands-on practice with Looker Studio, but in preparation for the lab, let's explore the three steps needed to create a Looker Studio dashboard. First, choose a template. You can start with either a pre-built template or a blank report. Second, link the dashboard to a data source. This might come from BigQuery, a local file, or a Google application like Google Sheets or Google Analytics–or a combination of any of these sources. And third, explore your dashboard!

Summary

Well done completing the lab on building a data pipeline for streaming data! Before you move on in the course, let's do a quick recap. In this section, you explored the streaming data workflow, from ingestion to visualization. You started by learning about the four common big data challenges, or the 4 volume (data size) Variety (data format) Velocity (data speed), and Veracity (data accuracy). These challenges can be especially common when dealing with streaming data. You then learned how the streaming data workflow provided by Google can help address these challenges. You started with Pub/Sub, which can be used to ingest a large volume of IoT data from diverse resources in various formats. After that, you explored Dataflow, a serverless, NoOps service, to process the data. 'Process' here refers to ETL (extract, transform, and load). And finally, you were introduced to two Google visualization tools, Looker, and Looker Studio.

3 Big Data with BigQuery

Introduction

In the previous section of this course, you explored Dataflow and Pub/Sub, Google Cloud's solutions to processing streaming data. Now let's focus your attention on BigQuery. You'll begin by exploring BigQuery's two main services, storage, and analytics, and then get a demonstration of the BigQuery user interface. After that, you'll see how BigQuery ML provides a data-to-AI lifecycle all within one place. You'll also learn about BigQuery ML project phases, as well as key commands. Finally, you'll get hands-on practice using BigQuery ML to build a custom ML model. Let's get started. BigQuery is a fully managed data warehouse. A data warehouse is a large store, containing terabytes and petabytes of data gathered from a wide range of sources within an organization, that's used to guide management decisions. At this point, it's useful to consider what the main difference is between a data warehouse and a data lake. A data lake is just a pool of raw, unorganized, and unclassified data, which has no specified purpose. A data warehouse on the other hand, contains structured and organized data, which can be used for advanced querying. Being fully managed means that BigQuery takes care of the underlying infrastructure, so you can focus on using SQL queries to answer business questions–without worrying about deployment, scalability, and security. Let's look at some of the key features of BigQuery. BigQuery provides two services in on storage plus analytics. It's a place to store petabytes of data. For reference, 1 petabyte is equivalent to 11,000 movies at 4k quality. BigQuery is also a place to analyze data, with built-in features like machine learning, geospatial analysis, and business intelligence, which we will look at a bit later on. BigQuery is a fully managed serverless solution, meaning that you don't need to worry about provisioning any resources or managing servers in the backend but only focus on using SQL queries to answer your organization's questions in the frontend. If you've never written SQL before, don't worry. This course provides resources and labs to help. BigQuery has a flexible pay-as-you-go pricing model where you pay for the number of bytes of data your query processes and for any permanent table storage. If you prefer to have a fixed bill every month, you can also subscribe to flat-rate pricing where you have a reserved amount of resources for use. Data in BigQuery is encrypted at rest by default without any action required from a customer. By encryption at rest, we mean encryption used to protect data that is stored on a disk, including solid-state drives, or backup media. BigQuery has built-in machine learning features so you can write ML models directly in BigQuery using SQL. Also, if you decide to use other professional tools—such as Vertex AI from Google Cloud—to train your ML models, you can export datasets from BigQuery directly into Vertex AI for a seamless integration across the data-to-AI lifecycle. So, what does a typical data warehouse solution architecture look like? The input data can be either real-time or batch data. If you recall from the last module when we discussed the four challenges of big data, in modern organizations the data can be in any format (variety), any size (volume), any speed (velocity), and possibly inaccurate (veracity). If it's streaming data, which can be either structured or unstructured, high speed, and large volume, Pub/Sub is needed to digest the data. If it's batch data, it can be directly uploaded to Cloud Storage. After that, both pipelines lead to Dataflow to process the data. That's the place we ETL – extract, transform, and load – the data if needed. BigQuery sits in the middle to link data processes using Dataflow and data access through analytics, AI, and ML tools. The job of the analytics engine of BigQuery at the end of a data pipeline is to ingest all the processed data after ETL, store and analyze it, and possibly output it for further use such as data visualization and machine learning.

BigQuery outputs usually feed into two bucket business intelligence tools and AI/ML tools. If you're a business analyst or data analyst, you can connect to visualization tools like Looker, Looker Studio, Tableau, or other BI tools. If you prefer to work in spreadsheets, you can query both small or large BigQuery datasets directly from Google Sheets and even perform common operations like pivot tables. Alternatively, if you're a data scientist or machine learning engineer, you can directly call the data from BigQuery through AutoML or Workbench. These AI/ML tools are part of Vertex AI, Google's unified ML platform. BigQuery is like a common staging area for data analytics workloads. When your data is there, business analysts, BI developers, data scientists, and machine learning engineers can be granted access to your data for their own insights.

Storage and analytics

BigQuery provides two services in one. It's both a fully managed storage facility to load and store datasets, and also a fast SQL-based analytical engine. The two services are connected by Google's high-speed internal network. It's this super-fast network that allows BigQuery to scale both storage and compute independently, based on demand. Let's take a look at how BigQuery manages the storage and metadata for datasets. BigQuery can ingest datasets from a variety of different sources, including internal data, which is data saved directly in BigQuery, external data, multi-cloud data, and public datasets. After the data is stored in BigQuery, it's fully managed by BigQuery and is automatically replicated, backed up, and set up to auto scale. BigQuery also offers the option to query external data sources–like data stored in other Google Cloud storage services like Cloud Storage, or in other Google Cloud database services like Spanner or Cloud SQL–and bypass BigQuery managed storage. That means a raw CSV file in Cloud Storage, or a Google Sheet can be used to write a query without being ingested by BigQuery first. One thing to note he inconsistency might result from saving and processing data separately. To avoid that risk, consider using Dataflow to build a streaming data pipeline into BigQuery. In addition to internal/native and external data sources, BigQuery can also ingest data for multi-cloud data, which is data stored in multiple cloud services, such as AWS or Azure. A public dataset. If you don't have data of your own, you can analyze any of the datasets available in the public dataset marketplace. There are three basic patterns to load data into BigQuery. The first is a batch load, where source data is loaded into a BigQuery table in a single batch operation. This can be a one-time operation, or it can be automated to occur on a schedule. A batch load operation can create a new table or append data into an existing table. The second is streaming, where smaller batches of data are streamed continuously so that the data is available for querying in near-real time. And the third is generated data, where SQL statements are used to insert rows into an existing table or to write the results of a query to a table. Of course, the purpose of BigQuery is not just to save data; it's for analyzing data and helping to make business decisions. BigQuery is optimized for running analytic queries over large datasets. It can perform queries on terabytes of data in seconds and on petabytes in minutes. This performance lets you analyze large datasets efficiently and get insights in near real time. Let's look at the analytics features that are available in BigQuery. BigQuery support ad hoc analysis using Standard SQL, the BigQuery SQL dialect. Geospatial analytics using geography data types and Standard SQL geography functions. Building machine learning models using BigQuery ML. Building rich, interactive business intelligence dashboards using BigQuery BI Engine. By default, BigQuery runs interactive queries, which means that the queries are executed as needed. BigQuery also offers batch queries, where each query is queued on your behalf and the query starts when idle resources are available, usually within a few minutes. Up next, you'll see a demonstration in BigQuery. Please note that you might notice a slightly different user interface.

Introduction to BigQuery ML

Although BigQuery started out solely as a data warehouse, over time it has evolved to provide features that support the data-to-AI lifecycle. In this section of the course, we'll explore BigQuery's capabilities for building machine learning models and the ML project phases and walk you through the key ML commands in SQL. If you've worked with ML models before, you know that building and training them can be very time-intensive. You must first export data from your datastore into an IDE (integrated development environment) such as Jupyter Notebook or Google Colab and then transform the data and perform all your feature engineering steps before you can feed it into a training model. Then finally, you need to build the model in TensorFlow, or a similar library, and train it locally on your computer or on a virtual machine. To improve the model performance, you also need to go back and forth to get more data and create new features. This process will need to be repeated, but it's so time-intensive that you'll probably stop after a few iterations. Also, we just mentioned TensorFlow and feature engineering; in the past, if you weren't familiar with these technologies, ML was left to the data scientists on your team and was not available to you. Now you can create and execute machine learning models on your structured datasets in BigQuery in just a few minutes using SQL queries. There are two steps needed to start step create a model with a SQL statement. Here we use the bikeshare dataset as an example. Step rite a SQL prediction query and invoke ml. Predict. And that's it! You now have a model and can view the results. Additional steps might include activities like evaluating the model, but if you know basic SQL, you can now implement ML; pretty cool! BigQuery ML was designed to be simple, like building a model in two steps. That simplicity extends to defining the machine learning hyperparameters, which let you tune the model to achieve the best training result. Hyperparameters are the settings applied to a model before the training starts, like the learning rate. With BigQuery ML, you can either manually control the hyperparameters or hand it to BigQuery starting with a default hyperparameter setting and then automatic tuning. When using a structured dataset in BigQuery ML, you need to choose the appropriate model type. Choosing which type of ML model depends on your business goal and the datasets. BigQuery supports supervised models and unsupervised models. Supervised models are task-driven and identify a goal. Alternatively, unsupervised models are data-driven and identify a pattern. Within a supervised model, if your goal is to classify data, like whether an email is spam, use logistic regression. If your goal is to predict a number, like shoe sales for the next three months, use linear regression. Within an unsupervised model, if your goal is to identify patterns or clusters and then determine the best way to group them, like grouping random pictures of flowers into categories, you should use cluster analysis. Once you have your problem outlined, it's time to decide on the best model. Categories include classification and regression models. There are also other model options to choose from, along with ML ops. Logistic regression is an example of a classification model, and linear regression is an example of a regression model. We recommend that you start with these options and use the results to benchmark to compare against more complex models such as DNN (deep neural networks), which may take more time and computing resources to train and deploy. In addition to providing different types of machine learning models, BigQuery ML supports features to deploy, monitor, and manage the ML production, called ML Ops, which is short for machine learning operations. Options include importing TensorFlow models for batch prediction Exporting models from BigQuery ML for online prediction And hyperparameter tuning using Vertex AI Vizier We discuss ML Ops in more detail later in this course.

Using BigQuery ML to predict customer lifetime value

Now that you’re familiar with the types of ML models available to choose from, high-quality data must be used to teach the models what they need to learn. The best way to learn the key concepts of machine learning on structured datasets is through an example. In this scenario, we’ll predict customer lifetime value with a model. Lifetime value, or LTV, is a common metric in marketing… …used to estimate how much revenue

or profit you can expect from a customer given their history and customers with similar patterns. We’ll use a Google Analytics ecommerce dataset from Google’s own merchandise store that sells branded items like t-shirts and jackets. The goal is to identify high-value customers and bring them to our store with special promotions and incentives. Having explored the available fields, you may find some useful in determining whether a customer is high value based on their behavior on our website. These fields include customer lifetime pageviews, total visits, average time spent on the site, total revenue brought in, and ecommerce transactions on the site. Remember that in machine learning, we feed in columns of data and let the model figure out the relationship to best predict the label. It may even turn out that some of the columns weren’t useful at all to the model in predicting the outcome -- we'll see later how to determine this. Now that we have some data, we can prepare to feed it into the model. Incidentally, to keep this example simple, we’re only using seven records, but we’d need tens of thousands of records to train a model effectively. Before we feed the data into the model, we first need to define our data and columns in the language that data scientists and other ML professionals use. Using the Google Merchandise Store example, a record or row in the dataset is called an example, an observation, or an instance. A label is a correct answer, and you know it’s correct because it comes from historical data. This is what you need to train the model on in order to predict future data. Depending on what you want to predict, a label can be either a numeric variable, which requires a linear regression model, or a categorical variable, which requires a logistic regression model. For example, if we know that a customer who has made transactions in the past and spends a lot of time on our website often turns out to have high lifetime revenue, we could use revenue as the label and predict the same for newer customers with that same spending trajectory. This means forecasting a number, so we can use a linear regression as a starting point to model. Labels could also be categorical variables like binary values, such as High Value Customer or not. To predict a categorical variable, if you recall from the previous section, you need to use a logistic regression model. Knowing what you’re trying to predict, such as a class or a number, will greatly influence the type of model you’ll use. But what do we call all the other data columns in the data table? Those columns are called features, or at least potential features. Each column of data is like a cooking ingredient you can use from the kitchen pantry. But keep in mind that using too many ingredients can ruin a dish! The process of sifting through data can be time consuming. Understanding the quality of the data in each column and working with teams to get more features or more history is often the hardest part of any ML project. You can even combine or transform feature columns in a process called feature engineering. If you’ve ever created calculated fields in SQL, you’ve already executed the basics of feature engineering. Also, BigQuery ML does much of the hard work for you, like automatically one-hot encoding categorical values. One-hot encoding is a method of converting categorical data to numeric data to prepare it for model training. From there, BigQuery ML automatically splits the dataset into training data and evaluation data. And finally, there is predicting on future data. Let’s say new data comes in that you don’t have a label for, so you don’t know whether it is for a high-value customer. You do, however, have a rich history of labeled examples for you to train a model on.

So, if we train a model on the known historical data, and are happy with the performance, then we can use it to predict on future datasets!

BigQuery ML project phases

Let’s walk through the key phases of a machine learning project. In phase 1, you extract, transform, and load data into BigQuery, if it isn’t there already. If you’re already using other Google products, like YouTube for example, look out for easy connectors to get that data into BigQuery before you build your own pipeline. You can enrich your existing data warehouse with other data sources by using SQL joins. In phase 2, you select and preprocess features. You can use SQL to create the training dataset for the model to learn from. You’ll recall that BigQuery ML does some of the preprocessing for you, like one-hot encoding of your categorical variables. One-hot encoding converts your categorical data into numeric data that is required by a training model. In phase 3, you create the model inside BigQuery. This is done by using the “CREATE MODEL” command. Give it a name, specify the model type, and pass in a SQL query with your training dataset. From there, you can run the query. In phase 4, after your model is trained, you can execute an ML.EVALUATE query to evaluate the performance of the trained model on your evaluation dataset. It’s here that you can analyze loss metrics like a Root Mean Squared Error for forecasting models and area-under-the-curve, accuracy, precision, and recall, for classification models. We’ll explore these metrics later in this course. In phase 5, the final phase, when you’re happy with your model performance, you can then use it to make predictions. To do so, invoke the ml. PREDICT command on your newly trained model to return with predictions and the model’s confidence in those predictions. With the results, your label field will have “predicted” added to the field name. This is your model’s prediction for that label. Now that you’re familiar with the key phases of an ML project, let’s explore some of the key commands of BigQuery ML. You’ll remember from an earlier video that you can create a model with just the CREATE MODEL command. Models have OPTIONS, which you can specify. The most important, and the only one required, is the model type. If you want to overwrite an existing model, use the CREATE OR REPLACE MODEL command. You can inspect what a model learned with the ML.WEIGHTS command and filtering on an input column. The output of ML.WEIGHTS is a numerical value, and each feature has a weight from -1 to 1. That value indicates how important the feature is for predicting the result, or label. If the number is closer to 0, the feature isn't important for the prediction. However, if the number is closer to -1 or 1, then the feature is more important for predicting the result. To evaluate the model's performance, you can run an ML.EVALUATE command against a trained model. You get different performance metrics depending on the model type you chose. And if you want to make batch predictions, you can use the ML.PREDICT command on a trained model and pass through the dataset you want to make the prediction on. Now let’s explore a consolidated list of BigQuery ML commands for supervised models. First in BigQuery ML, you need a field in your training dataset titled LABEL, or you need to specify which field or fields are your labels using the input_label_cols in your model OPTIONS. Second, your model features are the data columns that are part of your SELECT statement after your CREATE MODEL statement. After a model is trained, you can use the ML.FEATURE_INFO command to get statistics and metrics about that column for additional analysis. Next is the model object itself. This is an object created in BigQuery that resides in your BigQuery dataset. You train many different models, which will all be objects stored under your BigQuery dataset, much like your tables and views. Model objects can display information for when it was last updated or how many training runs it completed. Creating a new model is as easy as writing CREATE MODEL, choosing a type, and passing in a training dataset. Again, if you’re predicting on a numeric field, such as next year's sales, consider linear regression for forecasting. If it's a discrete class like high, medium, low, or spam/not-spam, consider using logistic regression for classification. While the model is running, and even after it's complete, you can view training progress with ML.TRAINING_INFO. As mentioned earlier, you can inspect weights to see what the model learned about the importance of each feature as it relates to the label you're predicting. The importance is indicated by the weight of each feature. You can see how well the model performed against its evaluation dataset by using ML.EVALUATE. And lastly, getting predictions is as simple as writing ML.PREDICT and referencing your model name and prediction dataset. Now it's time to get some hands-on practice building a machine learning model in BigQuery. In the lab that follows this video, you'll use ecommerce data from the Google Merchandise Store website http://shop.googlemerchandisestore.com/ The site's visitor and order data has been loaded into BigQuery, and you'll build a machine learning model to predict whether a visitor will return for more purchases later. You'll get practice loading data into BigQuery from a public dataset. . Querying and exploring the ecommerce dataset. Creating a training and evaluation dataset to be used for batch prediction. Creating a classification (logistic regression) model in BQML. Evaluating the performance of your machine learning model. And Predicting and ranking the probability that a visitor will make a purchase

BigQuery ML key commands

Now that you're familiar with the key phases of an ML project, let's explore some of the key commands of BigQuery ML. You'll remember from an earlier video that you can create a model with just the CREATE MODEL command. Models have OPTIONS, which you can specify. The most important, and the only one required, is the model type. If you want to overwrite an existing model, use the CREATE OR REPLACE MODEL command. You can inspect what a model learned with the ML.WEIGHTS command and filtering on an input column. The output of ML.WEIGHTS is a numerical value, and each feature has a weight from -1 to 1. That value indicates how important the feature is for predicting the result, or label. If the number is closer to 0, the feature isn't important for the prediction. However, if the number is closer to -1 or 1, then the feature is more important for predicting the result. To evaluate the model's performance, you can run an ML.EVALUATE command against a trained model. You get different performance metrics depending on the model type you chose. And if you want to make batch predictions, you can use the ML.PREDICT command on a trained model and pass through the dataset you want to make the prediction on. Now let's explore a consolidated list of BigQuery ML commands for supervised models. First in BigQuery ML, you need a field in your training dataset titled LABEL, or you need to specify which field or fields are your labels using the input_label_cols in your model OPTIONS. Second, your model features are the data columns that are part of your SELECT statement after your CREATE MODEL statement. After a model is trained, you can use the ML.FEATURE_INFO command to get statistics and metrics about that column for additional analysis. Next is the model object itself. This is an object created in BigQuery that resides in your BigQuery dataset. You train many different models, which will all be objects stored under your BigQuery dataset, much like your tables and views. Model objects can display information for when it was last updated or how many training runs it completed. Creating a new model is as easy as writing CREATE MODEL, choosing a type, and passing in a training dataset. Again, if you're predicting on a numeric field, such as next year's sales, consider linear regression for forecasting. If it's a discrete class like high, medium, low, or spam/not-spam, consider using logistic regression for classification. While the model is running, and even after it's complete, you can view training progress with ML.TRAINING_INFO. As mentioned earlier, you can inspect weights to see what the model learned about the importance of each feature as it relates to the label you're predicting. The importance is indicated by the weight of each feature. You can see how well the model performed against its evaluation dataset by using ML.EVALUATE. And lastly, getting predictions is as simple as writing ML.PREDICT and referencing your model name and prediction dataset.

Summary

Well done on completing another lab! Hopefully you now feel more comfortable building custom machine learning models with BigQuery ML! Let's review what we explored in this section of the course. Our focus was on BigQuery, the data warehouse that provides two services in one. It's a fully-managed storage facility for datasets, and a fast SQL-based analytical engine. BigQuery sits between data processes and data uses, like a common staging area. It gets data from ingestion and processing and outputs data to BI tools such as Looker and Looker Studio and ML tools such as Vertex AI. After the data is in BigQuery, business analysts, BI developers, data scientists, and machine learning engineers can be granted access to the data for their own insights. In addition to traditional data warehouses, BigQuery offers machine learning features. This means you can use BigQuery to directly build ML models in five key phases. In phase 1, you extract, transform, and load data into BigQuery, if it isn't there already. In phase 2, you select and preprocess features. You can use SQL to create the training dataset for the model to learn from. In phase 3, you create the ML model inside BigQuery. In phase 4, after your model is trained, you can execute an ML.EVALUATE query to evaluate the performance of the trained model on your evaluation dataset. In phase 5, the final phase, when you're happy with your model performance, you can then use it to make predictions.

4 Machine Learning Options on Google Cloud

Introduction

In previous sections of this course, you learned about many data engineering tools available from Google Cloud. Now let's switch our focus to machine learning. In this section, we'll explore the different options Google Cloud offers for building machine learning models. Additionally, we will explain how a product called Vertex AI can help solve machine learning challenges. So you might be wondering, "Why should I trust Google for artificial intelligence and machine learning?" Google is an AI-first company and is recognized as a leader across industries because of its contributions in the fields of artificial intelligence and machine learning. In 2022 Google was recognized as a leader in the Gartner Magic Quadrant for Cloud AI Developer services, and in recent years has also received recognition in numerous annual industry awards and reports. And at Google we've been implementing artificial intelligence for over ten years into many of our critical products, systems, and services. For example, have you ever noticed how Gmail automatically suggests three responses to a received message? This feature is called Smart Reply, which uses artificial intelligence to predict how you might respond. Behind this intelligence is AI technology known as natural language processing, which is just one example of an impressive list of technologies that Google scientists and engineers are working on. We'll explore these in more depth later in the course. The goal of these technologies is not for exclusive use to only benefit Google customers. The goal is to enable every company to be an AI company by reducing the challenges of AI model creation to only the steps that require human judgment or creativity. So for workers in the travel and hospitality field, this might mean using AI and ML to improve aircraft scheduling or provide customers with dynamic pricing options. For retail-sector employees, it might mean using AI and ML to leverage predictive inventory planning. The potential solutions are endless. What are the problems in your business that artificial intelligence and machine learning might help you solve? Take a moment to think about this question before continuing to the next video.

Options to build ML models

Google Cloud offers four options for building machine learning models. The first option is BigQuery ML. You'll remember from an earlier section of this course that BigQuery ML is a tool for using SQL queries to create and execute machine learning models in BigQuery. If you already have your data in BigQuery and your problems fit the pre-defined ML models, this could be your choice. The second option is to use pre-built APIs, which are application programming interfaces. This option lets you leverage machine learning models that have already been built and trained by Google, so you don't have to build your own machine learning models if you don't have enough training data or sufficient machine learning expertise in-house. The third option is AutoML, which is a no-code solution, so you can build your own machine learning models on Vertex AI through a point-and-click interface. And finally, there is custom training, through which you can code your very own machine learning environment, the training, and the deployment, which gives you flexibility and provides the control over the ML pipeline. Let's compare the four options to help you decide which one to use for building your ML model. Please note that the technologies change constantly, and this is only a brief guideline. Data type BigQuery ML only supports tabular data while the other three support tabular, image, text, and video. Training data si re-built APIs do not require any training data, while BigQuery ML and custom training require a large amount of data. Machine learning and coding expertise re-Built APIs and AutoML are user friendly with low requirements, while Custom training has the highest requirement and BigQuery ML requires you to understand SQL. Flexibility to tune the hyperparameter at the moment, you can't tune the hyperparameters with Pre-built APIs or AutoML, however, however, you can experiment with hyperparameters using BigQuery ML and custom training. Time to train the mod re-built APIs require no time to train a model because they directly use pre-built models from Google. The time to train a model for the other three options depends on the specific project. Normally, custom training takes the longest time because it builds the ML model from scratch, unlike AutoML and BigQuery ML. Selecting the best option will depend on your business needs and ML expertise. If your data engineers, data scientists, and data analysts are familiar with SQL and already have your data in BigQuery, BigQuery ML lets you develop SQL-based models. If your business users or developers have little ML experience, using pre-built APIs is likely the best choice. Pre-built APIs address common perceptual tasks such as vision, video, and natural language. They are ready to use without any ML expertise or model development effort. If your developers and data scientists want to build custom models with your own training data while spending minimal time coding, then AutoML is your choice. AutoML provides a code-less solution to enable you to focus on business problems instead of the underlying model architecture and ML provisioning. If your ML engineers and data scientists want full control of ML workflow, Vertex AI custom training lets you train and serve custom models with code on Vertex Workbench. We've already explored BigQuery ML, so in the videos that follow, we'll explore the other three options in more detail.

Pre-built APIs

Good Machine Learning models require lots of high-quality training data. You should aim for hundreds of thousands of records to train a custom model. If you don't have that kind of data, pre-built APIs are a great place to start. Pre-built APIs are offered as services. In many cases they can act as building blocks to create the application you want without expense or complexity of creating your own models. They save the time and effort of building, curating, and training a new dataset so you can just jump right ahead to predictions. So, what are some of the pre-built APIs? Let's explore a short list. The Speech-to-Text API converts audio to text for data processing. The Cloud Natural Language API recognizes parts of speech called entities and sentiment. The Cloud Translation API converts text from one language to another. The Text-to-Speech API converts text into high quality voice audio. The Vision API works with and recognizes content in static images. And the Video Intelligence API recognizes motion and action in video. And Google has already done a lot of work to train these models using Google datasets. For example, the Vision API is based on Google's image datasets, the Speech-to-Text API is trained on YouTube captions, and the Translation API is built on Google's neural machine translation technology. You'll recall that how well a model is trained depends on how much data is available to train it. As you might expect, Google has a lot of images, text, and ML researchers to train it's pre-built models. This means less work for you. Let's take a minute and try out the Vision API in a browser. Start by navigating to cloud.google.com/vision in Chrome, and then Scroll down to try the API by uploading an image. You can actually experiment with each of the ML APIs in a browser. When you're ready to build a production model, you'll need to pass a JSON object request to the API and parse what it returns.

AutoML

To understand AutoML, which is short for automated machine learning, let's briefly look at how it was built. If you've worked with ML models before, you know that training and deploying ML models can be extremely time consuming, because you need to repeatedly add new data and features, try different models, and tune parameters to achieve the best result. To solve this problem, when AutoML was first announced in January of 2018, the goal was to automate machine learning pipelines to save data scientists from manual work, such as tuning hyperparameters and comparing against multiple models. But how could this be done? Well, machine learning is similar to human learning. It all starts with gathering the right information. For AutoML, two technologies are vital. The first is known as transfer learning. With transfer learning, you build a knowledge base in the field. You can think of this like gathering lots of books to create a library. Transfer learning is a powerful technique that lets people with smaller datasets, or less computational power, achieve state-of-the-art results by taking advantage of pre-trained models that have been trained on similar, larger data sets. Because the model learns via transfer learning, it doesn't have to learn from scratch, so it can generally reach higher accuracy with much less data and computation time than models that don't use transfer learning. The second technology is neural architecture search. The goal of neural architecture search is to find the optimal model for the relevant project. Think of this like finding the best book in the library to help you learn what you need to. AutoML is powered by the latest machine-learning research, so although a model performs training, the AutoML platform actually trains and evaluates multiple models and compares them to each other. This neural architecture search produces an ensemble of ML models and chooses the best one. Leveraging these technologies has produced a tool that can significantly benefit data scientists. One of the biggest benefits is that it's a no-code solution. That means it can train high-quality custom machine learning models with minimal effort and requires little machine learning expertise. This allows data scientists to focus their time on tasks like defining business problems or evaluating and improving model results. Others might find AutoML useful as a tool to quickly prototype models and explore new datasets before investing in development. This might mean using it to identify the best features in a dataset, for example. So, how does AutoML work exactly? AutoML supports four types of da mage, tabular, text, and video. For each data type, AutoML solves different types of problems, called objectives. To get started, upload your data into AutoML. It can come from Cloud Storage, BigQuery, or even your local machine. From there, inform AutoML of the problems you want to solve. Some problems may sound similar to those mentioned in pre-built APIs. However, the major difference is that pre-built APIs use pre-built machine learning models, while AutoML uses custom-built models. In AutoML, you use your own data to train the machine learning model and then apply the trained model to predict your goal. For image data you can use a classification model to analyze image data and return a list of content categories that apply to the image. For example, you could train a model that classifies images as containing a dog or not containing a dog, or you could train a model to classify images of dogs by breed. You can also use an object detection model to analyze your image data and return annotations that consist of a label and bounding box location for each object found in an image. For example, you could train a model to find the location of the dogs in image data. For tabular data you can use a regression model to analyze tabular data and return a numeric value. For example, you could train a model to estimate a house’s value or rental price based on a set of factors such as location, size of the house, and number of bedrooms. You can use a classification model to analyze tabular data and return a list of categories. For example, you could train a model to classify different types of land into high, median, and low potentials for commercial real estate. And a forecasting model can use multiple rows of time-dependent tabular data from the past to predict a series of numeric values in the future. For example, you could use the historical plus the economic data to predict what the housing market will look like in the next five years. For text data you can use a classification model to analyze text data and return a list of categories that apply to the text found in the data. For example, you can classify customer questions and comments to different categories and then redirect them to corresponding departments. An entity extraction model can be used to inspect text data for known entities referenced in the data and label those entities in the text. For example, you can label a social media post in terms of predefined entities such as time, location, and topic, etc. This can help with online search, similar to the concept of a hashtag, but created by machine. And a sentiment analysis model can be used to inspect text data and identify the prevailing emotional opinion within it, especially to determine a writer's comment as positive, negative, or neutral. And finally, for video data you can use a classification model to analyze video data and return a list of categorized shots and segments. For example, you could train a model that analyzes video data to identify whether the video is of a soccer, baseball, basketball, or football game. You can use an object tracking model to analyze video data and return a list of shots and segments where these objects were detected. For example, you could train a model that analyzes video data from soccer games to identify and track the ball. And an action recognition model can be used to analyze video data and return a list of categorized actions with the moments the actions happened. For example, you could train a model that analyzes video data to identify the action moments involving a soccer goal, a golf swing, a touchdown, or a high five. In reality, you may not be restricted to just one data type and one objective but rather need to combine multiple data types and different objectives to solve a business problem. AutoML is a powerful tool that can help across these different data types and objectives.

Custom training

We’ve explored the options Google Cloud provides to build machine learning models using BigQuery ML, pre-built APIs, and AutoML. Now let's take a look at the last option, custom training. If you want to code your machine learning model, you can use this option by building a custom training solution with Vertex AI Workbench. Workbench is a single development environment for the entire data science workflow, from exploring, to training, and then deploying a machine learning model with code. Before any coding begins, you need to determine what environment you want your ML training code to use. There are two options pre-built container or a custom container. Imagine that a container is a kitchen. A pre-built container would represent a fully furnished room with cabinets and appliances (which represent the dependencies) that includes all the cookware (which represents the libraries) you need to make a meal. So, if your ML training needs a platform like TensorFlow, Pytorch, Scikit-learn, or XGboost, and Python code to work with the platform, a pre-built container is probably your best solution. A custom container, alternatively, is like an empty room with no cabinets, appliances, or cookware. You define the exact tools that you need to complete the job.

Vertex AI

For years now, Google has invested time and resources into developing big data and AI. Google had developed key technologies and products, from its roots in the development of Scikit Learn back in 2007 to Vertex AI today. As an AI-first company, Google has applied AI technologies to many of its products and services, like Gmail, Google Maps, Google Photos, and Google Translate, just to name a few. But developing these technologies doesn’t come without challenges, especially when it involves developing machine learning models and putting them into production. Some traditional challenges include determining how to handle large quantities of data, determining the right machine learning model to train the data, and harnessing the required amount of computing power. Then there are challenges around getting ML models into production. Production challenges requires scalability, monitoring, and continuous integration and continuous delivery or deployment. In fact, according to Gartner, only half of enterprise ML projects get past the pilot phase. And finally, there are ease-of-use challenges. Many tools on the market require advanced coding skills, which can take a data scientist’s focus away from model configuration. And without a unified workflow, data scientists often have difficulties finding tools. Google’s solution to many of the production and ease-of-use challenges is Vertex AI, a unified platform that brings all the components of the machine learning ecosystem and workflow together. So, what exactly does a unified platform mean? In the case of Vertex AI, it means having one digital experience to create, deploy, and manage models over time, and at scale. For example, During the data readiness stage, users can upload data from wherever it’s stored– Cloud Storage, BigQuery, or a local machine. Then, during the feature readiness stage, users can create features, which are the processed data that will be put into the model, and then share them with others using the feature store. After that, it’s time for Training and Hyperparameter tuning. This means that when the data is ready, users can experiment with different models and adjust hyperparameters. And finally, during deployment and model monitoring, users can set up the pipeline to transform the model into production by automatically monitoring and performing continuous improvements. And to refer back to the different options we explored earlier, Vertex AI allows users to build machine learning models with either AutoML, a code-less solution or Custom Training, a code-based solution. AutoML is easy to use and lets data scientists spend more time turning business problems into ML solutions, while custom training enables data scientists to have full control over the development environment and process. Being able to perform such a wide range of tasks in one unified platform has many benefits. This can be summarized with four t’s seamless. Vertex

AI provides a smooth user experience from uploading and preparing data all the way to model training and production. It's scalable. The Machine Learning Operations (MLOps) provided by Vertex AI helps to monitor and manage the ML production and therefore scale the storage and computing power automatically. It's sustainable. All of the artifacts and features created using Vertex AI can be reused and shared. And it's speedy. Vertex AI produces models that have 80% fewer lines of code than competitors.

AI Solutions

Now that you've explored the four different options available to create machine learning models with Google Cloud, let's take a few minutes to explore Google Cloud's artificial intelligence solution portfolio. It can be visualized with three layers. The bottom layer is the AI foundation and includes the Google Cloud infrastructure and data. The middle layer represents the AI development platform, which includes the four ML options you just learned about AutoML and custom training, which are offered through Vertex AI, and pre-built APIs and BigQuery ML. The top layer represents AI solutions, for which there are two groups, horizontal solutions, and industry solutions. Horizontal solutions usually apply to any industry that would like to solve the same problem. Examples include Document AI and CCAI. Document AI uses computer vision and optical character recognition, along with natural language processing, to create pretrained models to extract information from documents. The goal is to increase the speed and accuracy of document processing to help organizations make better decisions faster, while reducing costs. Another example of a horizontal solution is Contact Center AI, or CCAI. The goal of CCAI is to improve customer service in contact centers through the use of artificial intelligence. It can help automate simple interactions, assist human agents, unlock caller insights, and provide information to answer customer questions. And the second group is vertical, or industry solutions. These represent solutions that are relevant to specific industries. Examples include detail Product Discovery, which gives retailers the ability to provide Google-quality search and recommendations on their own digital properties, helping to increase conversions and reduce search abandonment, Google Cloud Healthcare Data Engine, which generates healthcare insights and analytics with one end-to-end solution, and Lending DocAI, which aims to transform the home loan experience for borrowers and lenders by automating mortgage document processing. You can learn more about Google Cloud's growing list of AI solutions at cloud.google.com/solutions/ai.

Summary

We've covered a lot of information in this section of the course. Let's do a quick recap. To start, we explored Google's history as an AI-first company. From there, we looked at the four options Google Cloud offers to build machine learning models. This includes BigQuery ML, pre-built APIs, AutoML, and Custom Training. Next, we introduced Vertex AI, a tool that combines the functionality of AutoML, which is codeless, and custom training, which is code-based, to solve production and ease-of-use problems. You'll recall that selecting the best ML option will depend on your business needs and ML expertise. If you are data engineers, data scientists, and data analysts are familiar with SQL and already have your data in BigQuery, BigQuery ML lets you develop SQL-based models. If your business users or developers have little ML experience, using pre-built APIs is likely the best choice. Pre-built APIs address common perceptual tasks such as vision, video, and natural language. They are ready to use without any ML expertise or model development effort. If your developers and data scientists want to build custom models with your own training data while spending minimal time coding, then AutoML is your choice.

AutoML provides a code-less solution to enable you to focus on business problems instead of the underlying model architecture and ML provisioning. If your ML engineers and data scientists want full control of the ML workflow, Vertex AI custom training lets you train and serve custom models with code on Vertex Workbench. Using pre-built containers, you can leverage popular ML libraries, such as Tensorflow and PyTorch. Alternatively, you can build a custom container from scratch. And finally, we introduced Google Cloud AI solutions. The solutions are built on top of the four ML development options to meet both horizontal and vertical market needs. The solutions are built on top of the four ML development options to meet both horizontal and vertical market needs.

5 The Machine Learning Workflow with Vertex AI

Introduction

In the previous section of this course, you explored the machine learning options available on Google Cloud. Now let's switch our focus to the machine learning Workflow with Vertex AI – from data preparation to model training, and finally, model deployment. Vertex AI, Google's AI platform, provides developers and data scientists one unified environment to build custom ML models. This process is actually not too different from serving food in a restaurant–starting with preparing raw ingredients through to serving dishes on the table. Later in this section, you'll get hands-on practice building a machine-learning model end-to-end using AutoML on Vertex AI. But before we get into the details, let's look at the basic differences between machine learning and traditional programming. In traditional programming, simply put, one plus one equals two (1+1=2). Data, plus rules–otherwise known as algorithms–lead to answers. And with traditional programming, a computer can only follow the algorithms that a human has set up. But what if we're just too lazy to figure out the algorithms? Or what if the algorithms are too complex to figure out? This is where machine learning comes in. With machine learning, we feed a machine a large amount of data, along with answers that we would expect a model to conclude from that data. Then, we show the machine a learning method by selecting a machine learning model. From there, we expect the machine to learn from the provided data and examples to solve the puzzle on its own. So, instead of telling a machine how to do addition, we give it pairs of numbers and the answers. For example, 1,1, and 2, and 2, 3, and 5. We then ask it to figure out how to do addition on its own. But how is it possible that a machine can actually learn to solve puzzles? For machine learning to be successful, you'll need lots of storage, like what's available with Cloud Storage, and the ability to make fast calculations, like with cloud computing. There are many practical examples this capability. For example, by feeding Google Photos lots of pictures with tags, we can teach the software to associate and then automatically attach tags to new pictures. Tags can then be used for search functionality, or even to automatically create photo albums. Can you come up with any other examples to apply machine learning capabilities? Take a moment to think about it. There are three key stages to this learning process. The first stage is data preparation, which includes two steps data uploading and feature engineering. You will be introduced to feature engineering in the next lesson. A model needs a large amount of data to learn from. Data used in machine learning can be either real-time streaming data or batch data, and it can be either structured, which is numbers and text normally saved in tables, or unstructured, which is data that can't be put into tables, like images and videos. The second stage is model training. A model needs a tremendous amount of iterative training. This is when training and evaluation form a cycle to train a model, then evaluate the model, and then train the model some more. The third and final stage is model serving. A model needs to actually be used in order to predict results. This is when the machine learning model is deployed, monitored, and managed. If you don’t move an ML model into production, it has no use and remains only a theoretical model. We mentioned at the start that the machine learning workflow on Vertex AI is not too different from serving food in a restaurant. So, if you compare these steps to running a restaurant, data preparation is when you prepare the raw ingredients, model training is when you experiment with different recipes, and model serving is when you finalize the menu to serve the meal to lots of hungry customers. Now it’s important to note that an ML workflow isn’t linear, it’s iterative. For example, during model training, you may need to return to dig into the raw data and generate more useful features to feed the model.

Data preparation

Now let’s look closer at an AutoML workflow. The first stage of the AutoML workflow is data preparation. During this stage, you must upload data and then prepare the data for model training with feature engineering. When you upload a dataset in the Vertex AI user interface, you’ll need to provide a meaningful name for the data and then select the data type and objective. AutoML allows four types of da mage, tabular, text, and video. To select the correct data type and objective, you shou tart by checking data requirements. We’ve included a link to these requirements in the resources section of this course. Next, you’ll need to add labels to the data if you haven’t already. A label is a training target. So, if you want a model to distinguish a cat from a dog, you must first provide sample images that are tagged—or labeled—either cat or dog. A label can be manually added, or it can be added by using Google’s paid label service via the Vertex console. These human labelers will manually generate accurate labels for you. The final step is to upload the data. Data can be uploaded from a local source, BigQuery, or Cloud Storage. You will practice these steps in the lab. After your data is uploaded to AutoML, the next step is preparing the data for model training with feature engineering. Imagine you’re in the kitchen preparing a meal. Your data is like your ingredients, such as carrots, onions, and tomatoes. Before you start cooking, you'll need to peel the carrots, chop the onions, and rinse the tomatoes. This is what feature engineering is li he data must be processed before the model starts training. A feature, as we discussed in the BigQuery module, refers to a factor that contributes to the prediction. It’s an independent variable in statistics or a column in a table. Preparing features can be both challenging and tedious. To help, Vertex AI has a function called Feature Store. Feature Store is a centralized repository to organize, store, and serve machine learning features. It aggregates all the different features from different sources and updates them to make them available from a central repository. Then, when engineers need to model something, they can use the features available in the Feature Store dictionary to build a dataset. Vertex AI automates the feature aggregation to scale the process. So, what are the benefits of Vertex AI Feature Store? First, features are shareable for training or serving tasks. Features are managed and served from a central repository, which helps maintain consistency across your organization. Second, features are reusable. This helps save time and reduces duplicative efforts, especially for high-value features. Third, features are scalable. Features automatically scale to provide low-latency serving, so you can focus on developing the logic to create the features without worrying about deployment. And fourth, features are easy to use. Feature Store is built on an easy-to-navigate user interface.

Model training

Now that our data is ready, which, if we return to the cooking analogy is our ingredients, it's time to train the model. This is like experimenting with some recipes. This stage involves two step model training, which would be like cooking the recipe, and model evaluation, which is when we taste how good the meal is. This process might be iterative. Before we get into more details about this stage, let's pause to clarify two terms artificial intelligence and machine learning. Artificial intelligence, or AI, is an umbrella term that includes anything related to computers mimicking human intelligence. For example, in an online word processor, robots performing human actions all the way down to spell check. Machine learning is a subset of AI that mainly refers to supervised and unsupervised learning. You might also hear the term deep learning, or deep neural networks. It's a subset of machine learning that adds layers in between input data and output results to make a machine learn at more depth. So, what's the difference between supervised and unsupervised learning? Supervised learning is task-driven and identifies a goal. Unsupervised learning, however, is data-driven and identifies a pattern. An easy way to distinguish between the two is that supervised learning provides each data point with a label, or an answer, while unsupervised does not. For example, if we were given sales data from an online retailer, we could use supervised learning to predict the sales trend for the next couple of months and use unsupervised learning to group customers together based on common characteristics. There are two major types of supervised learning the first is classification, which predicts a categorical variable, like using an image to tell the difference between a cat and a dog. The second type is a regression model, which predicts a continuous number, like using past sales of an item to predict a future trend. There are three major types of unsupervised learning: the first is clustering, which groups together data points with similar characteristics and assigns them to "clusters," like using customer demographics to determine customer segmentation. The second is association, which identifies underlying relationships, like a correlation between two products to place them closer together in a grocery store for a promotion. And the third is dimensionality reduction, which reduces the number of dimensions, or features, in a dataset to improve the efficiency of a model. For example, combining customer characteristics like age, driving violation history, or car type, to create an insurance quote. If too many dimensions are included, it can consume too many compute resources, which might make the model inefficient. Although Google Cloud provides four machine learning options, with AutoML and pre-built APIs you don't need to specify a machine learning model. Instead, you'll define your objective, such as text translation or image detection. Then on the backend, Google will select the best model to meet your business goal. With the other two options, BigQuery ML and custom training, you'll need to specify which model you want to train your data on and assign something called hyperparameters. You can think of hyperparameters as user-defined knobs in a machine that helps guide the machine learning process. For example, one parameter is a learning rate, which is how fast you want the machine to learn With AutoML, you don't need to worry about adjusting these hyperparameter knobs because the tuning happens automatically in the back end. This is largely done by a neural architect search, which finds the best fit model by comparing the performance against thousands of other models.

Model evaluation

While we are experimenting with a recipe, we need to keep tasting it constantly to make sure it meets our expectations. This is the model evaluation portion of the model training stage. Vertex AI provides extensive evaluation metrics to help determine a model's performance. Among the metrics are two sets of measurements. The first is based on the confusion matrix, for example recall and precision. The second is based on feature importance, which we'll explore later in this section of the course. A confusion matrix is a specific performance measurement for machine learning classification problems. It's a table with combinations of predicted and actual values. To keep things simple, we assume the output includes only two classes. Let's explore an example of a confusion matrix. The first is a true positive combination, which can be interpreted as, "The model predicted positive, and that's true." The model predicted that this is an image of a cat, and it actually is. The opposite of that is a true negative combination, which can be interpreted as, "The model predicted negative, and that's true." The model predicted that a dog is not a cat, and it actually isn't. Then there is a false positive combination, otherwise known as a Type 1 Error, which can be interpreted as, "The model predicted positive, and that's false." The model predicted that a dog is a cat, but it actually isn't. Finally, there is the false negative combination, otherwise known as a Type 2 Error, which can be interpreted as, "The model predicted negative, and that's false." The model predicted that a cat is not a cat, but it actually is. A confusion matrix is the foundation for many other metrics used to evaluate the performance of a machine learning model. Let's take a look at the two popular metrics, recall and precision, that you will encounter in the lab. Recall refers to all the positive cases and looks at how many were predicted correctly. This means that recall is equal to the true positives, divided by the sum of the true positive and false negatives. Precision refers to all the cases predicted as positive, and how many are actually positive. This means that precision is equal to the true positives, divided by the sum of the true positive and false positives. Imagine you're fishing with a net. Using a wide net, you caught both fish and roc 0 fish out of 100 total fish in the lake, plus 80 rocks. The recall in this case is 80%, which is calculated by the number of fish caught, 80, divided by the total number of fish in the lake, 100. The precision is 50%, which is calculated by taking the number of fish caught, 80, and dividing it by the number of fish and rocks collected, 160. Let's say you wanted to improve the precision, so you switched to a smaller net. This time you caught 20 fish and 0 rocks. The recall becomes 20% (20 out of 100 fish collected) and the precision becomes 100% (20 out of 20 total fish and rocks collected). Precision and recall are often a trade-off. Depending on your use case, you may need to optimize for one or the other. Consider a classification model where Gmail separates emails into two categoric spam and not-spam. If the goal is to catch as many potential spam emails as possible, Gmail may want to prioritize recall. In contrast, if the goal is to only catch the messages that are definitely spam without blocking other emails, Gmail may want to prioritize precision. In Vertex AI, the platform visualizes the precision and the recall curve so they can be adjusted based on the problem that needs solving. You'll get the opportunity to practice adjusting precision and recall in the AutoML lab. In addition to the confusion matrix and the metrics generated to measure model effectiveness such as recall and precision, the other useful measurement is featuring importance. In Vertex AI, feature importance is displayed through a bar chart to illustrate how each feature contributes to a prediction. The longer the bar, or the larger the numerical value associated with a feature, the more important it is. This information helps decide which features are included in a machine learning model to predict the goal. You will observe the feature importance chart in the lab as well. Feature importance is just one example of Vertex AI's comprehensive machine learning functionality called Explainable AI. Explainable AI is a set of tools and frameworks to help understand and interpret predictions made by machine learning models.

Model deployment and monitoring

The recipes are ready and now it's time to serve the meal! This represents the final stage of the machine learning workflow, model serving. Model serving consists of two steps: first, model deployment, which we can compare to serving the meal to a hungry customer, and second, model monitoring, which we can compare to checking with the waitstaff to ensure that the restaurant is operating efficiently. It's important to note that model management exists throughout this whole workflow to manage the underlying machine learning infrastructure. This lets data scientists focus on what to do, rather than how to do it. Machine learning operations, or MLOps, play a big role. MLOps combines machine learning development with operations and applies similar principles from DevOps to machine learning models, which is short for development and operations. MLOps aims to solve production challenges related to machine learning. In this case, this refers to building an integrated machine learning system and operating it in production. These are considered to be some of the biggest pain points by the ML practitioners' community, because both data and code are constantly evolving in machine learning. Practicing MLOps means advocating for automation and monitoring at each step of the ML system construction. This means adopting a process to enable continuous integration, continuous training, and continuous delivery. What does MLOps have to do with model serving? Well, let's start with model deployment, which is the exciting time when a model is implemented. In our restaurant analogy, it's when the food is put on the table for the customer to eat! MLOps provides a set of best practices on the backend to automate this process. There are three options to deploy a machine learning model. The first option is to deploy to an endpoint. This option is best when immediate results with low latency are needed, such as making instant recommendations based on a user's browsing habits whenever they're online. A model must be deployed to an endpoint before that model can be used to serve real-time predictions. The second option is to deploy using batch prediction. This option is best when no immediate response is required, and accumulated data should be processed with a single request. For example, sending out new ads every other week based on the user's recent purchasing behavior and what's currently popular on the market. And the final option is to deploy using offline prediction. This option is best when the model should be deployed in a specific environment off the cloud. In the lab, you will practice predicting with an endpoint. Now let's shift our focus to model monitoring. The backbone of MLOps on Vertex AI is a tool called Vertex AI Pipelines. It automates, monitors, and governs machine learning systems by orchestrating the workflow in a serverless manner. Imagine you're in a production control room, and Vertex AI Pipelines is displaying the production data onscreen. If something goes wrong, it automatically triggers warnings based on a predefined threshold. With Vertex AI Workbench, which is a notebook tool, you can define your own pipeline. You can do this with prebuilt pipeline components, which means that you primarily need to specify how the pipeline is put together using components as building blocks. And it's with these final two steps, model deployment and model monitoring, that we complete our exploration of the machine learning workflow. The restaurant is open and operating smoothly – Bon appetite!

Lab recap predicting loan risk with AutoML

Well done on completing the AutoML lab! You've now had a chance to use Vertex AI to build a machine learning model without writing a single line of code. Let's take a few moments to review the results of the lab, starting with the confusion matrix. But before that begins, please pause to consider the matrix results for yourself. The true positives were 100%. This represents the percentage of people the model predicted would repay their loan who actually did pay it back. The true negatives were 87%. This represents the percentage of people the model predicted would not repay their loan who indeed did not pay it back. The false negatives were 0%. This represents the percentage of people the model predicted would not repay their loan, but who actually did pay it back. And finally, the false positives were 13%. This represents the percentage of people the model predicted would repay their loan, but who actually did not pay it back. As a general principle, it's good to have high true positives and true negatives, and low false positives and false negatives. However, how high or low they need to be really depended on the business goals you're looking to achieve. There are different ways to improve the performance of a model, which might include using a more _accurate data source, using a larger dataset, choosing a different type of ML model, or tuning the hyperparameters. Let's also review the precision-recall curve from the AutoML lab. The confidence threshold determines how a machine learning model counts the positive cases. A higher threshold increases the precision but decreases recall. A lower threshold decreases the precision but increases recall. Moving the threshold to zero produces the highest recall of 100%, and the lowest precision of 50%. So, what does that mean? That means the model predicts that 100% of loan applicants will be able to repay a loan they take out. However, in actuality, only 50% of people were able to repay the loan. Using this threshold to identify the default cases in this example can be risky, because it means that you're only likely to get half of the loan investment back. Now let's move to the other extreme by moving the threshold to 1. This will produce the highest precision of 100% with the lowest recall of 1%. What does this mean? It means that of all the people who were predicted to repay the loan, 100% of them actually did. However, you rejected 99% of loan applicants by only offering loans to 1% of them. That's a pretty big loss of business for your company. These are both extreme examples, but it's important that you always try to set an appropriate threshold for your model.

Summary

Before we finish this section of the course, let's quickly review the three stages of the machine learning workflow–with the help of our restaurant analogy. In stage one, data preparation, we uploaded data and applied feature engineering. This translated to gathering our ingredients and then chopping and prepping them in the kitchen. In stage two, model training, the model was trained and evaluated. This is where we experimented with our recipes and tasted the meal to make sure it turned out as expected. And in the final stage, model serving, the model was deployed and monitored. This translates to serving the meal to the hungry customers and adjusting the menu as more people tried the dish.

Course summary

Congratulations on completing the Big Data and Machine Learning Fundamentals course! We hope that you learned some valuable information from this course that will help advance your career. Throughout the course, we introduced a number of products and technologies to support Google's data-to-AI lifecycle. Let's do a final review of the main concepts presented. In the first section of the course, you were introduced to the Google Cloud infrastructure and Google's big data and machine learning products. Of the three layers of the Google Cloud infrastructure, you explored the middle and top layers. On the middle layer sit compute and storage. Google Cloud decouples compute and storage so they can scale independently based on need. And on the top layer sit the big data and machine learning products, which enable you to perform tasks to ingest, store, process, and deliver business insights, data pipelines, and machine learning models. In the second section of the course, you explored data engineering for streaming data. This included how to build a streaming data pipeline– from ingestion with Pub/Sub, to processing with Dataflow, to visualization using Looker and Looker Studio. After that, in the third section of the course, you were introduced to BigQuery, which is Google's fully-managed data warehouse. BigQuery provides two services in on storage plus analytics. You also learned about BigQuery ML; the machine learning tool used for developing machine learning models directly in BigQuery. In the fourth section of the course, you explored the options available to build and deploy machine learning models with Google Cloud. If you're familiar with SQL and already have data in BigQuery, you can use BigQuery ML to develop machine learning models. If you have little ML experience, using pre-built APIs is likely the best choice. Pre-built APIs address common perceptual tasks such as vision, video, and natural language. They're ready to use without any ML expertise or model development effort. If you want to build custom models with your own training data while spending minimal time coding, then AutoML is a great choice. AutoML provides a code-less solution to enable you to focus on business problems instead of the underlying model architecture and ML provisioning. And if you want full control of the machine learning workflow, Vertex AI custom training lets you train and serve custom models with code on Vertex Workbench. Using pre-built containers, you can leverage popular ML libraries, such as Tensorflow and PyTorch. Alternatively, you can build a custom container from scratch. In the final section of the course, you learned about the machine learning workflow using Vertex AI, a unified platform that brings all the components of the machine learning ecosystem and workflow together. The machine learning workflow comprises three stages. In stage one, data preparation, data is uploaded, and feature engineering is applied. In stage two, model training, the model is trained and evaluated. And in stage three, model serving, the model is deployed and monitored. We hope that this course is just the beginning of your big data and machine learning journey. For more training and hands-on practice with data engineering and analytics, please refer to cloud.google.com/training/data-engineering-and-analytics. And for more training with machine learning and AI, please explore the options available at cloud.google.com/training/machinelearning-ai. And if you're interested in validating your expertise and showcasing your ability to transform businesses with Google Cloud technology, you might consider working toward a a Google Cloud certification. You can learn more about Google Cloud's certification offerings at cloud.google.com/certifications. Thanks for completing this course. We'll see you next time!