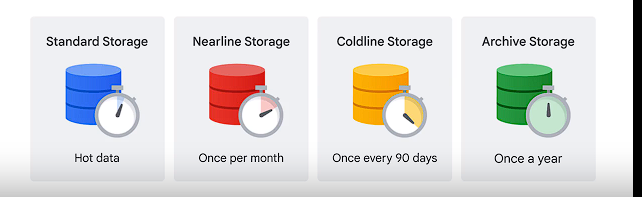
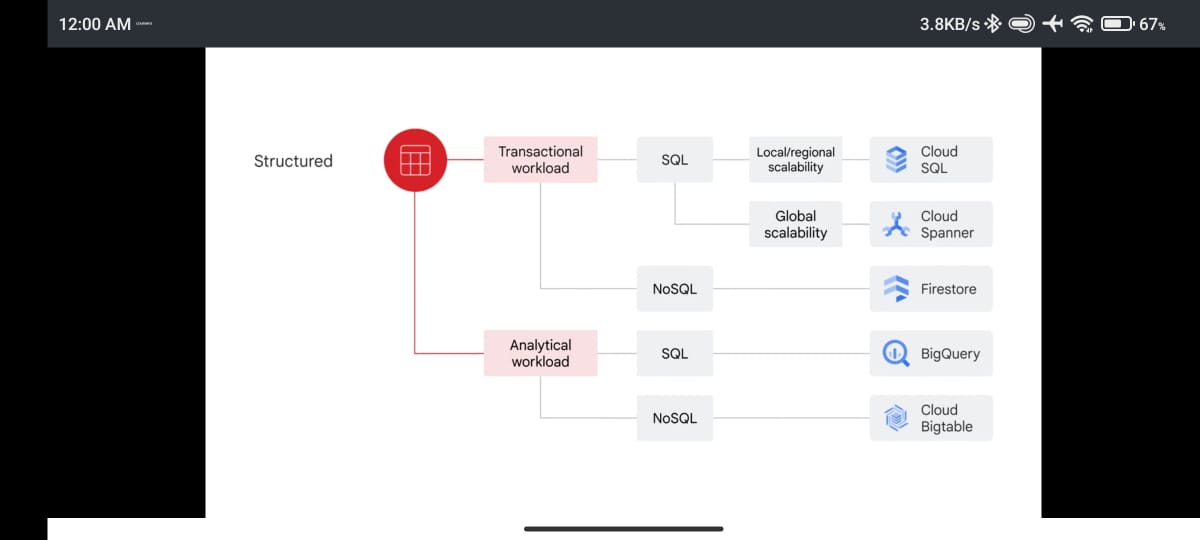


* **Google Cloud Infrastructure:**
  + Based in five major geographic locations: North America, South America, Europe, Asia, and Australia.
  + Multiple service locations crucial for qualities like availability, durability, and latency.
  + Latency measures packet travel time from source to destination.
* **Geographic Structure:**
  + Each major location divided into several regions and zones.
  + **Regions:** Independent geographic areas composed of zones.
  + **Zones:** Specific areas where Google Cloud resources are deployed.
  + Example: London (Europe West 2) region has multiple zones.
* **Resource Deployment and Zones:**
  + Launching resources (e.g., virtual machines) specifies the zone for resource redundancy.
  + Zonal resources operate within a single zone; if a zone goes down, resources within that zone are affected.
* **Control Over Geographic Locations:**
  + Users can specify geographic locations for running services/resources.
  + Flexibility to choose zonal, regional, or multiregional levels.
  + Benefits: Bringing applications closer to users worldwide and ensuring protection during regional issues (e.g., natural disasters).
* **Multiregional Support in Services:**
  + Some services support multiregion configurations.
  + Example: Cloud Spanner allows data replication across multiple zones and regions defined by the instance configuration.
  + Enables low-latency data access from various locations within the configured regions.
* **Google Cloud's Reach:**
  + Currently supports 103 zones in 34 regions.
  + Constant expansion and addition of new zones and regions.
  + For updated information, visit cloud.google.com/about location
* **Compute Services in Google Cloud:**
  + **Compute Engine (IaaS):** Offers virtual compute, storage, and network similar to physical data centers. Provides flexibility for managing server instances directly.
  + **Google Kubernetes Engine (GKE):** Runs containerized applications in a Cloud environment, emphasizing container deployment over individual virtual machines.
  + **App Engine (PaaS):** Fully managed platform binding code to required libraries, allowing focus on application logic rather than infrastructure.
  + **Cloud Functions (Functions as a Service - FaaS):** Executes code in response to events without local software installation or server management. Completely serverless execution environment.
  + **Cloud Run:** Fully managed compute platform for event-driven stateless workloads. Abstracts away infrastructure management, automatically scales, and charges only for resources used.
* **Google Photos and Compute Capability:**
  + **Video Stabilization Feature:**
    - Utilizes compute capability for automatic video stabilization.
    - Requires various data components: video frames, camera position, orientation from gyroscope, and lens motion.
    - Processing billions of data points from videos to feed the ML model for stabilization.
    - Google Photos stored trillions of photos and videos, necessitating robust machine learning models for accurate stabilization.
* **Machine Learning and Processing Power:**
  + **Hardware Limitations:**
    - Standard hardware on personal computers or smartphones insufficient for complex ML model training.
    - Training production ML models occurs in vast data centers; smaller versions deployed to personal devices.
  + **Rapid Demand for Processing Power:**
    - Stanford University's AI Index Report noted exponential growth in computing power demand for AI since 2012.
    - CPUs and GPUs reached limitations in scaling, unable to meet the rapid demand for machine learning.
* **Tensor Processing Unit (TPU):**
  + **Introduction in 2016 by Google:**
    - Custom-developed application-specific integrated circuits.
    - Designed specifically for accelerating machine learning workloads.
    - Domain-specific hardware tailored for efficient matrix multiplication in machine learning.
    - Faster and significantly more energy-efficient than CPUs and GPUs for AI applications.
* **Integration of TPUs:**
  + **Usage Across Google Products:**
    - Integrated into various Google services, providing state-of-the-art hardware for supercomputing technology.
* **Decoupling Compute and Storage in Cloud Computing:**
  + Unlike desktop computing, cloud computing separates processing limitations from storage disks.
  + Applications typically require a database and storage solution.
* **Google Cloud's Storage Solutions:**
  + Compute Engine allows running databases on virtual machines or opt for fully managed database and storage services.
  + Services include:
    - Cloud Storage
    - Cloud Bigtable
    - Cloud SQL
    - Cloud Spanner
    - Firestore
    - BigQuery
* **Purpose of Storage Solutions:**
  + Reduce time and effort in data storage.
  + Offers elastic storage creation through web interfaces or command lines.
* **Types of Data: Unstructured vs. Structured:**
  + **Unstructured Data:**
    - Non-tabular data (documents, images, audio).
    - Suited for Cloud Storage, now also possible in BigQuery.
* **Overview of Cloud Storage:**
  + Managed service for storing unstructured data.
  + Objects stored in buckets within a project; buckets can be grouped under an organization.
  + Objects, buckets, and projects are resources in Google Cloud.
  + Use cases: serving website content, data archival, disaster recovery, large data distribution.
* **Storage Classes in Cloud Storage:**
  + **Standard Storage:** Best for frequently accessed or short-term data.
  + **Nearline Storage:** Ideal for infrequently accessed data.
  + **Coldline Storage:** For data accessed at most once every 90 days.
  + **Archive Storage:** Lowest-cost option for rarely accessed data (< once a year).



* **Structured Data:**
  + Stored in tables, rows, and columns.
  + Two types: transactional and analytical workloads.
* **Transactional Workloads:**
  + **Online Transaction Processing (OLTP) systems:**
    - Fast data inserts/updates for row-based records.
    - Standardized queries affecting few records.
  + **Suitable Options:**
    - Cloud SQL for local-regional scalability.
    - Cloud Spanner for global scalability.
* **Analytical Workloads:**
  + **Online Analytical Processing (OLAP) systems:**
    - Reading entire datasets.
    - Complex queries, e.g., aggregations.
  + **Suitable Options:**
    - BigQuery for SQL-based petabyte-scale analysis.
    - Cloud Bigtable for scalable NoSQL for high-throughput, low-latency applications.

Top of Form

* **Google's Early Big Data Challenges:**
  + Handling large datasets, fast-changing data, and diverse data types due to the necessity of indexing the expanding World Wide Web.
* **Introduction of Google File System (GFS) - 2002:**
  + Purpose: Address data sharing and petabyte-scale storage needs.
  + Foundation for Cloud Storage and managed storage in BigQuery.
* **MapReduce Introduction - 2004:**
  + Developed to manage the growing web content volume.
  + Designed for large-scale data processing across commodity server clusters.
* **Addressing Streaming User Actions - 2005:**
  + Solution: Release of Cloud Bigtable, a high-performance NoSQL database for extensive analytical and operational workloads.
* **Transition from MapReduce to Dremel (2008 - 2010):**
  + Development of Dremel, breaking data into smaller shards and introducing autoscaling.
  + Dremel became BigQuery's query engine, revolutionizing big-data processing.
* **Further Technological Solutions:**
  + **Colossus (2010):** Successor to GFS, a cluster-level file system.
  + **BigQuery (2010):** Fully-managed, serverless data warehouse supporting ANSI SQL and machine learning capabilities.
  + **Spanner (2012):** Globally available, scalable relational database.
  + **Pub/Sub (2015):** Service for streaming analytics and data integration pipelines.
  + **TensorFlow (2015):** Open-source software library for machine learning and AI.
* **2018 Releases:**
  + **Tensor Processing Unit (TPU):** Specialized hardware for machine learning.
  + **AutoML:** Suite of machine learning products.
* **Innovations Leading to Vertex AI (2021):**
  + **Vertex AI:** Unified ML platform.
* **Impact on Big Data and Machine Learning Product Line:**
  + **Google Cloud Services (available products):** Cloud Storage, Dataproc, Cloud Bigtable, BigQuery, Dataflow, Firestore, Pub/Sub, Looker, Cloud Spanner, AutoML, and Vertex AI.
* **Course Integration:**
  + Practical exposure to these products and services through the course.
* **Ingestion and Process:**
  + **Pub/Sub:** Real-time and batch data ingestion.
  + **Dataflow:** Data pipeline management for processing.
  + **Dataproc:** Processes data using Apache Hadoop and Apache Spark.
  + **Cloud Data Fusion:** Simplifies data integration.
* **Data Storage:**
  + **Cloud Storage:** Object storage service.
  + **Cloud SQL:** Relational database service.
  + **Cloud Spanner:** Globally distributed, scalable relational database.
  + **Cloud Bigtable:** Scalable NoSQL database.
  + **Firestore:** NoSQL document-oriented database.
* **Analytics:**
  + **BigQuery:** Fully managed data warehouse for SQL-based data analysis.
  + **Looker:** Tool for data analysis and visualization.
* **Machine Learning (ML):**
  + **ML Development Platform (Vertex AI):**
    - **AutoML:** Automated ML model development.
    - **Vertex AI Workbench:** Collaboration and experimentation platform.
    - **TensorFlow:** Open-source ML library.
  + **AI Solutions:**
    - **Document AI:** Analyzes and extracts information from documents.
    - **Contact Center AI:** Enhances customer support with AI.
    - **Retail Product Discovery:** Improves product discovery for retailers.
    - **Healthcare Data Engine:** Provides insights in healthcare.

Understanding these categories helps in selecting the right products based on specific business needs and stages of the data-to-AI workflow. In the course, you'll delve deeper into these products and explore their functionalities for effective utilization in various scenarios.

1. **Google Cloud Infrastructure Overview:**
   * Explored layers of Google Cloud, including big data and machine learning products.
2. **Data Engineering for Streaming Data:**
   * Objective: Building real-time data solutions using Google Cloud services.
   * Processes:
     + Ingest streaming data via Pub/Sub.
     + Data processing with Dataflow.
     + Visualization of results using Looker and Looker Studio.
   * Intermediary step: Data stored and analyzed in a data warehouse like BigQuery.
3. **Section Contents:**
   * Challenges faced by data engineers in pipeline setup and management.
   * Learning about message-oriented architecture for global, reliable streaming message capture.
   * Designing streaming pipelines with Apache Beam and implementing them through Dataflow.
   * Visualizing data insights on Looker and Looker Studio dashboard.
   * Hands-on practice: Building an end-to-end data pipeline handling real-time data ingestion with Pub/Sub, processing via Dataflow, and visualization using Looker Studio.
4. **Understanding Streaming Data:**
   * Differentiation from batch processing: Streaming involves continuous flow and real-time analysis.
   * Examples: Fraud detection, intrusion detection.
   * Importance: Enables near real-time analysis and immediate actions on data.
   * Evolution from batch processing to real-time streaming in modern data processing.
   * Analogous to streaming music/movies: No need to download entire content.
5. **Batch Processing:**
   * Definition: Processing and analysis on a set of stored data.
   * Examples: Payroll and billing systems processed weekly or monthly.
   * Contrast with streaming data: Batch processing operates on stored data chunks, while streaming involves continuous flow and real-time analysis.
6. **Significance of Data Streams:**
   * Essential component in the world of big data.

challenges faced by data engineers and data scientists, known as the 4Vs (variety, volume, velocity, and veracity), along with their implications:

1. **Variety:**
   * **Challenge:** Data arrives from diverse sources in various formats (e.g., numbers, images, audio).
   * **Example:** Sensors in self-driving cars or point-of-sale data from multiple stores.
   * **Implication:** Organizing and alerting downstream systems about new transactions without duplicates becomes complex.
2. **Volume:**
   * **Challenge:** Handling varying data volumes, ranging from gigabytes to petabytes.
   * **Example:** Scaling pipeline code and infrastructure to manage huge data volumes.
   * **Implication:** Ensuring scalability to prevent system slowdowns or crashes.
3. **Velocity:**
   * **Challenge:** Processing data in near real-time as it arrives in the system.
   * **Example:** Handling late-arriving data, data with errors, or the need for on-the-fly transformation in streaming into a data warehouse.
   * **Implication:** Need for real-time processing capabilities and mechanisms to handle data inconsistencies.
4. **Veracity:**
   * **Challenge:** Ensuring data quality due to inconsistencies and uncertainties in big data from various sources and types.
   * **Implication:** Dealing with data inconsistencies and uncertainties to maintain data quality standards.

These challenges pose significant considerations for pipeline developers in creating scalable, reliable, and efficient data pipelines. The goal of understanding these challenges is to explore tools and solutions available to build successful streaming data pipelines while mitigating these complexities.

challenges and the use of Pub/Sub in data ingestion within a data pipeline:

**Challenges in Data Ingestion for IoT Devices:**

1. **Diverse Data Sources:**
   * Data streams from various devices/methods asynchronously.
   * Challenges with different data formats and quality (bad/delayed data).
2. **Distributing Event Messages:**
   * Difficulty in efficiently distributing event messages to the relevant subscribers.
   * Need for a method to collect and broadcast streaming messages from IoT sensors.
3. **High Volume and Velocity:**
   * Data arrives rapidly and in high volumes, requiring systems to handle the load efficiently.
4. **Reliability, Security, and Performance:**
   * Services need to be reliable, secure, and perform as expected despite varying data inputs.

**Google Cloud's Solution: Pub/Sub:**

* **What is Pub/Sub?**
  + Short for Publisher-Subscriber.
  + A distributed messaging service handling various device streams like IoT, gaming events, and application streams.
  + Guarantees at least once delivery of received messages with no provisioning required.
  + Offers open APIs, global service, and end-to-end encryption.

**Pub/Sub in Data Pipeline Architecture:**

1. **Data Ingestion:**
   * Incoming data from global devices ingested into Pub/Sub as the primary point.
   * Pub/Sub stores and broadcasts new data messages to subscribers of specific topics.
2. **Processing with Dataflow:**
   * Subscribers utilize Dataflow to ingest, transform, and output messages into an analytics data warehouse like BigQuery.
   * Enables elastic streaming pipelines for data processing.
3. **Visualization and Analysis:**
   * Data visualization tools like Looker monitor and visualize pipeline results.
   * AI/ML tools like Vertex AI leverage the data for business insights or predictions.
4. **Pub/Sub's Topic Model:**
   * Topics act as channels for data streams, akin to radio antennas always broadcasting.
   * Decoupled structure allows zero to multiple publishers and subscribers per topic, ensuring independence.
5. **Example: HR Topic for Employee Updates:**
   * Illustrates how different applications subscribe to an HR topic for employee-related notifications.
   * Publishers (e.g., full-time, contractor employee data sources) send events to the Pub/Sub HR topic.
   * Downstream applications independently process received messages for their respective tasks.
6. **Pub/Sub as a Solution:**
   * Suitable for handling changes in loosely coupled architectures with numerous publishers and subscribers.
   * Supports diverse inputs/outputs, even facilitating event publication from one topic to another.
7. **Scalable Data Warehouse Ingestion:**
   * Need for a scalable pipeline to reliably transfer messages from Pub/Sub to the data warehouse.

The next phase involves developing a pipeline that matches Pub/Sub's scalability and elasticity for reliable data transfer into the data warehouse.

**Dataflow for Pipeline Creation:**

1. **Purpose of Dataflow:**
   * Tool for creating pipelines handling both streaming and batch data.
   * Executes Extract, Transform, Load (ETL) processes for data.
2. **Challenges in Pipeline Development:**
   * Coding Pipeline Design: Challenges in designing and implementing the pipeline at scale.
   * Compatibility: Ensuring the pipeline code is compatible with both batch and streaming data or if refactoring is necessary.
3. **Pipeline Design Considerations:**
   * SDK Capabilities: Assessing if the software development kit (SDK) supports essential transformations, aggregations, windowing, and late data handling.
   * Leveraging Templates or Solutions: Using existing templates or solutions for efficient pipeline design.

**Apache Beam as a Solution:**

1. **Overview of Apache Beam:**
   * Open-source, unified programming model for defining and executing data processing pipelines (ETL, batch, stream).
   * **Unified Model**: Supports a single programming model for both batch and streaming data.
   * **Portability:** Works across various execution environments like Dataflow, Apache Spark, etc.
   * Extensibility: Allows developers to create and share **custom connectors and transformation libraries.**
   * Template Availability: Provides pre-built templates for initiating pipelines.
   * Language Support: Allows pipeline creation in Java, Python, or Go.
2. **Apache Beam SDK:**
   * Collection of development tools bundled together in an installable package.
   * Offers libraries for transformations and data connectors to handle data **sources and sinks.**
   * Generates a model representation from the code that's portable across different execution engines.

**Execution with Dataflow:**

1. **Dataflow Implementation:**
   * Dataflow serves as one of the engines executing Apache Beam pipelines.
   * Utilizes the model representation generated by Apache Beam for pipeline execution.

**Summary:** Dataflow and Apache Beam are key components in building scalable and versatile data pipelines. Apache Beam's unified and extensible nature, coupled with Dataflow's execution capabilities, streamlines the design, implementation, and execution of data processing pipelines capable of handling both streaming and batch data

**Choosing an Execution Engine:**

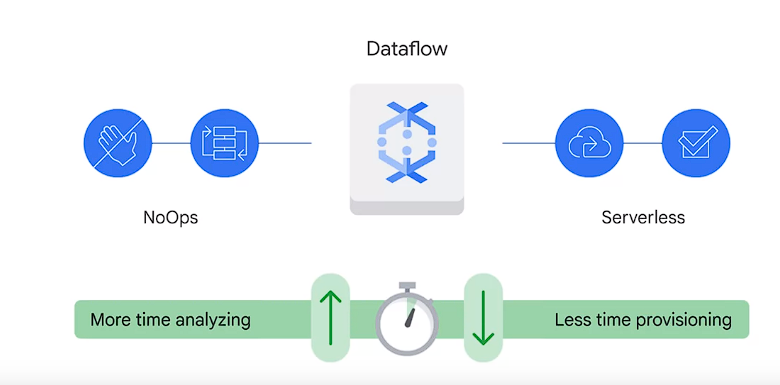
1. **Considerations:**
   * Maintenance Overhead: How much maintenance is involved?
   * Infrastructure Reliability: Is the infrastructure reliable?
   * Scalability of the Pipeline: How is pipeline scaling handled?
   * Monitoring Capabilities: How can the pipeline be monitored?
   * Vendor Lock-in: Is the pipeline locked to a specific service provider?

**Dataflow as an Execution Engine:**

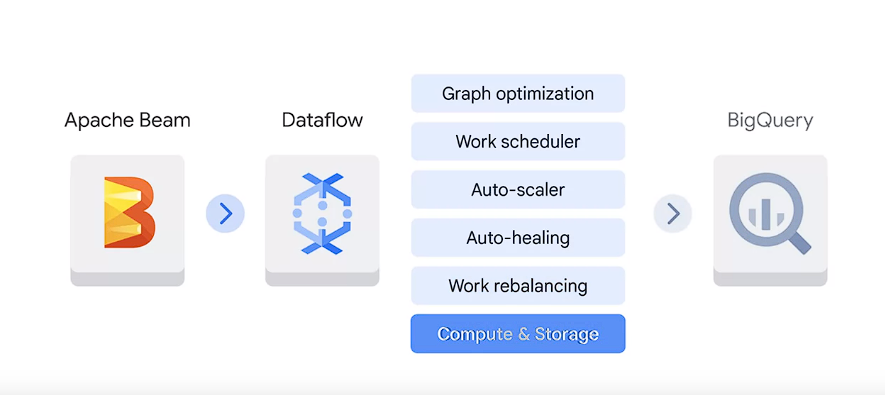
1. **Introduction to Dataflow:**
   * Fully managed service in the Google Cloud ecosystem for executing Apache Beam pipelines.
   * Handles complexities related to infrastructure setup and maintenance.
   * Built on Google's infrastructure, ensuring reliability and auto-scaling to meet pipeline demands.



1. **Serverless and NoOps:**
   * **No Operations (NoOps**): Doesn't require manual management by an **operations team because Maintenance monitoring and scaling are automatic**
   * Serverless Computing: Google Cloud manages infrastructure tasks on behalf of users (e.g., resource provisioning, performance tuning, reliability).

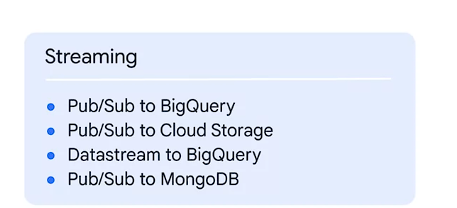


1. **Benefits of Dataflow:**
   * Enables focus on data analysis by automating maintenance, monitoring, and scaling.
   * Designed to be low maintenance, reducing the need for manual resource provisioning.
2. **Dataflow's Tasks during Job Execution:**

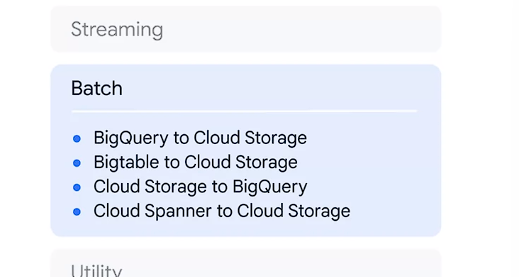


* + Optimizes pipeline model execution graph to remove inefficiencies.
  + Schedules distributed work to new workers and dynamically scales based on needs.
  + Auto-heals worker faults and efficiently rebalances efforts among workers.
  + Outputs data, for example, to destinations like BigQuery among others.

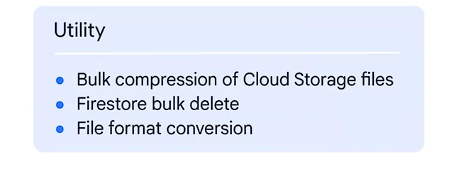
1. **Dataflow Templates:**
   * Useful for various use cases across Google Cloud products, catering to Java or Python developers.
   * Categories:
     + Streaming Templates: Process continuous/real-time data (e.g., Pub/Sub to BigQuery/Cloud Storage/MonogDB).

DataStream to BigQuery,  


* + - Batch Templates: Handle bulk or batch data (e.g., BigQuery to Cloud Storage).



* + - Utility Templates: Perform activities like compression, deletion, or conversion.



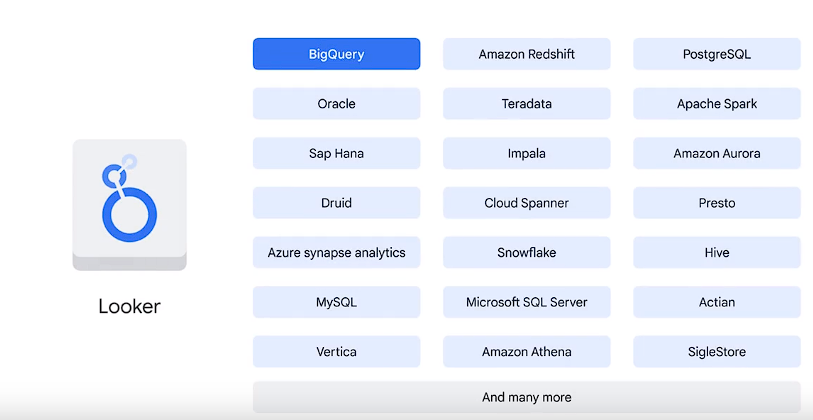
1. **Expanding Templates:**
   * Constantly growing list of templates catering to different data processing needs.
   * Covering common use cases and continuously updated with new functionalities.

By utilizing Dataflow as an execution engine and leveraging its capabilities, users can focus more on data analysis and less on the operational aspects of managing infrastructure, scaling, and monitoring within their Apache Beam pipelines. The availability of templates streamlines the development process for various use cases.

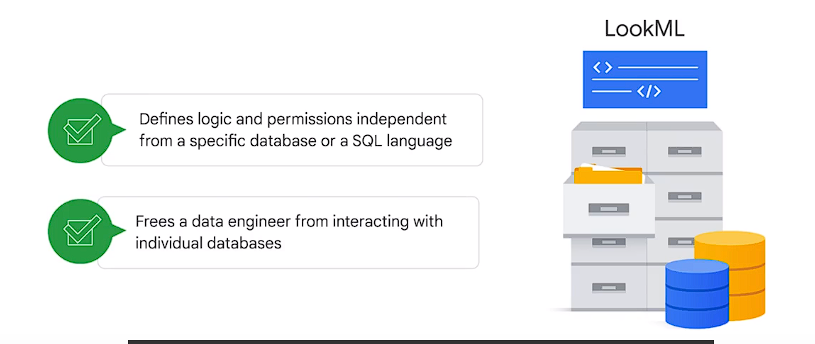
Telling a good story with data through a dashboard can be critical to the success of a data pipeline because **data that is difficult to interpret or draw insights from might be useless**. After data is in BigQuery a lot of skill and effort can still be required to uncover insights. **To help create an environment where stakeholders can easily interact with and visualize data, Google Cloud offers two solutions Looker and Looker Studio**

**Looker:**

1. **Support and Integration:**
   * Supports BigQuery and over 60 SQL databases

.

* + Defines a semantic modeling layer on top of database using **LookML**(Looker Modeling Language), enabling logic and permissions definition independent of specific databases or SQL languages.
  + Enables focusing on business logic across the organization rather than individual database interactions.

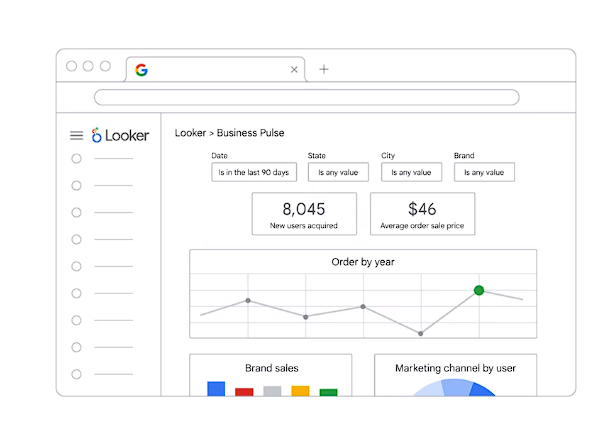


1. **Platform Overview:**
   * 100% web-based platform, easily integrable into existing workflows.

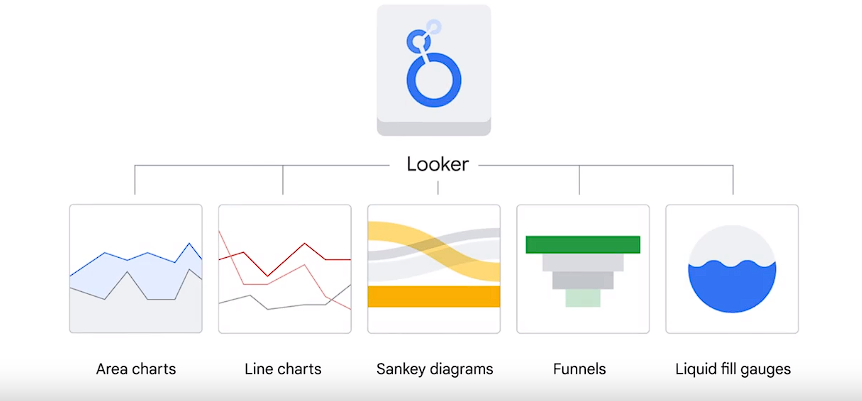


* + Allows sharing across multiple teams within an organization.
  + Offers an API for embedding Looker reports into other applications.

1. **Features and Capabilities:**
   * **Dashboards:** Provides visual representations like the Business Pulse dashboard, simplifying data interpretation.
     + For sales organizations, displays crucial figures like new user acquisitions, monthly sales trends, year-to-date orders, fostering team alignment and revenue insights.



* + **Visualization Options:** Offers diverse visualization choices such as area charts, line charts, Sankey diagrams, funnels, and liquid fill gauges.



* + **Sharing and Delivery:** Allows scheduling delivery through storage services like **Google Drive, Slack, and Dropbox** for team collaboration.

**Looker Dashboard Examples:**

1. **Business Pulse Dashboard:**
   * Visualizes key sales metrics at the start of a week (new users, sales trends, year-to-date orders) for quick insights.
   * Aids in aligning teams, identifying customer issues, and uncovering potential revenue losses.
2. **New York City Taxis Monitoring Dashboard:**
   * Displays metrics over time, including total revenue, passenger count, and ride count.
   * Utilizes time-series representations for monitoring metrics.
   * Enables plotting data on maps to visualize ride distribution, busy areas, and peak hours.

**Looker Studio:**

1. **Purpose and Functionality:**
   * Designed to assist in drawing insights and facilitating business decisions.
   * Offers various visualization tools and techniques, enabling data-driven decision-making.

Overall, Looker and Looker Studio provide a robust environment for interacting with data, creating intuitive dashboards, and presenting insights crucial for driving business decisions across multiple teams within an organization.

Top of Form

**Looker Studio:**

1. **Integration and Data Visualization:**
   * Integrated into BigQuery, enabling easy data visualization with minimal setup.
   * No administrator support required for establishing data connections **compared to Looker.**
   * Widely used across various Google products and applications.
2. **Use Cases and Integrations:**
   * **Google Analytics Integration:** Utilized for visualizing marketing website summaries.
     + Visualizes total visitor counts on a map, month-over-month trends, and visitor distribution by age.
   * **Google Cloud Billing Dashboard Integration:** Familiar dashboard for monitoring spending within Google Cloud accounts.

**Creating a Looker Studio Dashboard:**

1. **Dashboard Creation Steps:**
   * **Step 1: Choose a Template:** Begin with a pre-built template or a blank report.
   * **Step 2: Link Dashboard to Data Source:** Connect to data sources like BigQuery, local files, or Google applications (e.g., Google Sheets, Google Analytics) or combinations of any three sources.
   * **Step 3: Explore the Dashboard:** Once linked, explore and interact with the dashboard to visualize data insights.

**Hands-On Practice and Preparation:**

1. **Lab Preparation:**
   * In anticipation of the lab exercise, understanding the three primary steps required to create a Looker Studio dashboard.
   * Selection of templates, linking to data sources from various platforms (BigQuery, local files, Google applications), and exploring resulting dashboards.

**Conclusion:** Looker Studio simplifies data visualization within BigQuery, offering an intuitive interface to create interactive dashboards. By choosing templates, linking to diverse data sources, and exploring resulting dashboards, users can extract valuable insights for decision-making without extensive administrative setup or support.

BigData and BigQuery

1. **Focus on BigQuery:**
   * It's introduced as Google Cloud's fully managed data warehouse solution.
   * It's a repository for terabytes and petabytes of data from various sources within an organization.
2. **Difference Between Data Warehouse and Data Lake:**
   * BigQuery is contrasted with a data lake, explaining that a data warehouse contains structured and organized data for advanced querying, while a data lake stores raw, unorganized **data without a specific purpose.**

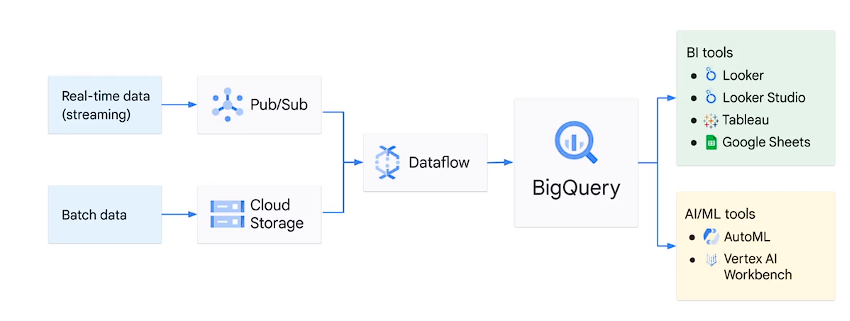
**Being fully managed means that BigQuery takes care of the underlying infrastructure, so you can focus on using SQL queries to answer business questions–without worrying about deployment, scalability, and security.**

1. **Key Features of BigQuery:**
   * It combines **storage and analytics**, providing a platform to store and analyze massive amounts of data.



* + Offers built-in features for machine learning, geospatial analysis, and business intelligence.
  + It's a fully managed serverless solution, eliminating concerns about infrastructure management free from provisioning resources or managing servers but focus on SQL queries.

1. **Pricing Model and Security:**
   * BigQuery operates on a flexible pay-as-you-go pricing model based on data query processing and permanent table storage.
   * Data in BigQuery is encrypted at rest by default, ensuring security without additional action required from customers.
2. **Machine Learning in BigQuery:**
   * It has built-in machine learning capabilities allowing the creation of ML models directly using SQL.
   * Integration with **Vertex AI** enables seamless data transfer for ML model training using professional tools.
3. **Data Warehouse Solution Architecture:**
   * Explains the architecture, starting with input data (real-time or batch), highlighting the challenges of big data (variety, volume, velocity, veracity).
   * Describes the role of **Pub/Sub for streaming** data and Cloud Storage for batch data.
   * Dataflow processes the data, performing ETL (Extract, Transform, Load), with BigQuery serving as the link between data processes and access through analytics and ML tools.
4. **Integration with Visualization and ML Tools:**
   * Business analysts can use BI tools like Looker, Tableau, or Google Sheets for data visualization.
   * Data scientists and ML engineers can access data directly from BigQuery through AutoML or Workbench, part of Vertex AI.

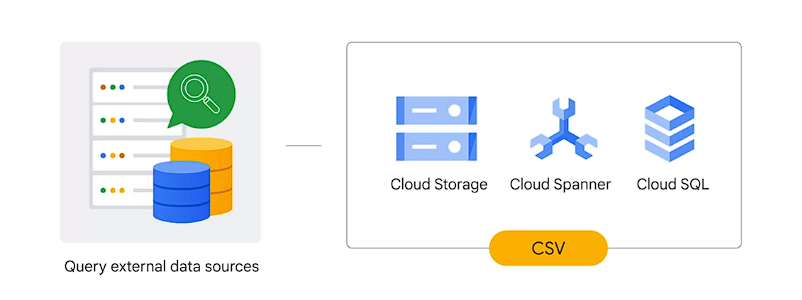


1. **BigQuery's Role in Data Analytics:**
   * Serves as a central area for various stakeholders (business analysts, BI developers, data scientists, ML engineers) to access and derive insights from data.

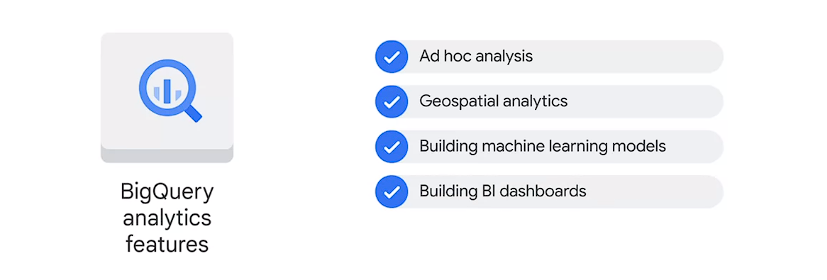
This portion delves deeper into BigQuery's functionalities, specifically focusing on its capabilities as both a storage facility and a powerful analytical engine:

1. **Dual Service of Storage and Analytics:**
   * BigQuery serves as both a managed storage solution for datasets and a high-speed SQL-based analytical engine.
   * It employs Google's internal network for high-speed connections, allowing independent scaling of storage and compute based on demand.
2. **Data Management in BigQuery:**
   * Ingests data from various sources: internal, external, multi-Cloud, and public datasets.



* + Automatically handles replication, backup, and auto-scaling for managed datasets.
  + 
  + Allows querying of external data sources without ingestion, minimizing inconsistencies but bearing the risk of separate data saving and processing.
  + inconsistency might result from saving and processing data separately. To avoid that risk, consider using data flow to build a streaming data pipeline into BigQuery. In addition to internal or native and external data sources, BigQuery can also ingest data from multi-Cloud data, which is data stored in multiple Cloud services, such as AWS or Azure, or a public data set. If you don't have any data of your own, you can analyze any of the datasets available in the public data set marketplace.

1. **Data Loading Patterns:**
   * Three primary patterns for loading data:
     + Batch load: Single or scheduled operation to load source data into BigQuery tables.(can create a new table or append data to an existing table)
     + Streaming: Continuous streaming of smaller data batches for near real-time availability.
     + Generated data: Using SQL statements to insert rows or write query results into tables(existing).
2. **Analytical Capabilities:**
   * BigQuery's optimization for analytical queries enables rapid analysis of massive datasets.
   * Offers near real-time insights by handling terabytes of data within seconds and petabytes within minutes.
3. **Analytics Features in BigQuery:**



* + Supports **ad hoc** analysis using standard SQL and its dialect, along with geospatial analytics using geography data types and functions.
  + Facilitates building machine learning models through BigQuery ML and creating interactive BI dashboards using BigQuery BI Engine.

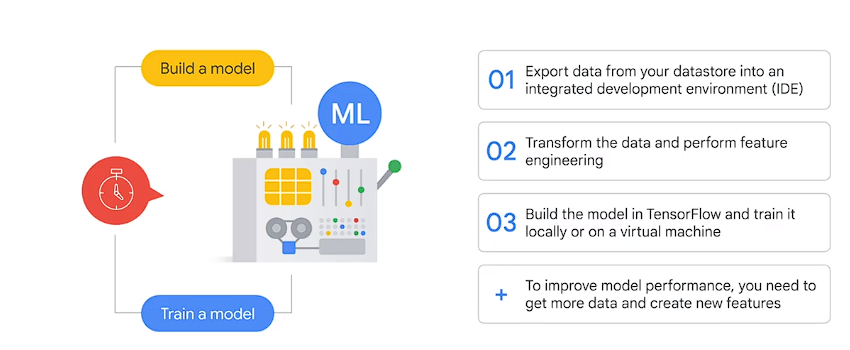
1. **Query Execution in BigQuery:**
   * **Executes interactive queries by default** and also provides batch **queries that are queued and executed when idle resources are available.**
2. **Upcoming Demonstration:**
   * An impending demonstration in BigQuery is mentioned, indicating a slightly different user interface might be observed.

This part covers BigQuery's capacity to manage, process, and analyze data from various sources while emphasizing its prowess in handling large-scale analytical queries efficiently. It also introduces upcoming insights through a demonstration of the platform, hinting at potential differences in the user interface.

Top of Form

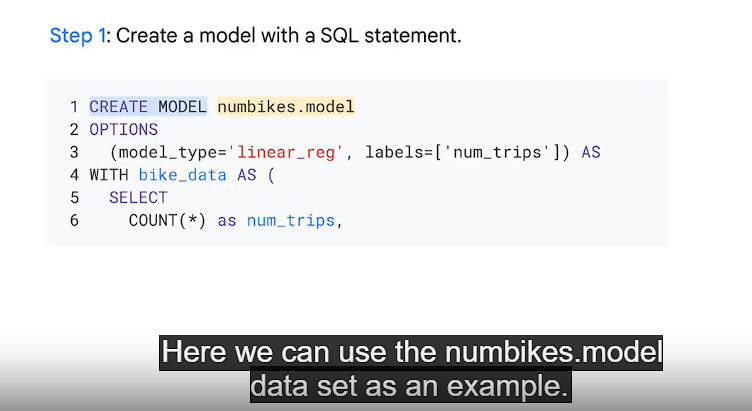
**BigQuery** has evolved from a data warehouse into a platform supporting the entire data-to-AI lifecycle, especially focusing on machine learning (ML) capabilities:

1. **Traditional ML Model Building Process:**

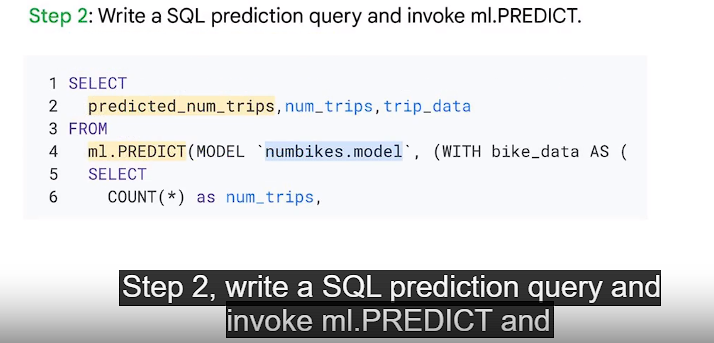


* + Traditional model building involved exporting data into IDEs, performing transformations, feature engineering, and training models using frameworks like TensorFlow.

1. **Simplification with BigQuery ML:**
   * BigQuery ML simplifies this process, enabling users to create and execute ML models directly on structured datasets using SQL queries, reducing time and complexity.
2. **Two-Step Model Creation in BigQuery ML:**
   * Step 1: Create a model with a SQL statement.



* + Step 2: Write a SQL prediction query and invoke ml.PREDICT to obtain results, demonstrating the simplicity of creating and utilizing ML models within BigQuery.

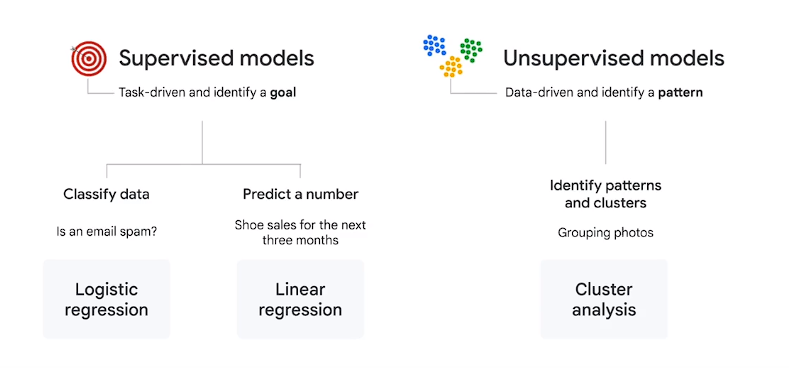


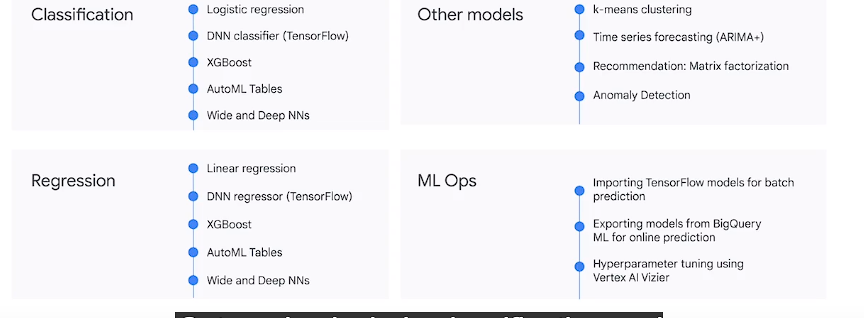
1. **Hyperparameter Control and Model Types**

**Hyperparamters are the settings apply to a model before the trainings starts like learning rates**

* + Users can control hyperparameters for model tuning or opt for automatic tuning.
  + Choosing the appropriate model type (supervised or unsupervised) in BigQuery ML depends on business goals and datasets.

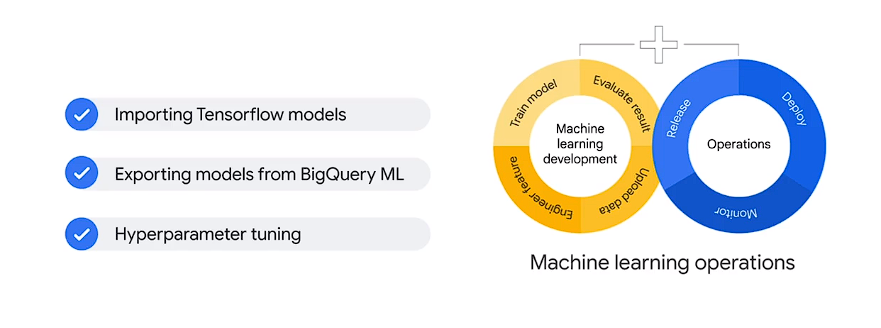
1. **Supervised and Unsupervised Models:**
   * Supervised models (task-driven) aim to identify a goal, e.g., classification (e.g., logistic regression) or prediction (e.g., linear regression).
   * Unsupervised models (data-driven) aim to identify patterns or clusters, e.g., cluster analysis for grouping photos.





1. **Choosing ML Models:**
   * Recommendations to start with basic models like logistic regression and linear regression and benchmark against more complex ones (e.g., Deep Neural Networks - DNNs).
2. **Introduction to ML Ops in BigQuery:**
   * BigQuery ML supports ML Ops (machine learning operations) for deploying, monitoring, and managing ML models in production.
   * Ops include importing/exporting models, hyperparameter tuning using Cloud AI Vizier, and will be explored in more detail later in the course.

This section emphasizes the transformation in the approach to ML model development, enabling users, even with basic SQL knowledge, to leverage BigQuery's capabilities to create and execute ML models efficiently. It also introduces the notion of ML Ops for managing ML models in production, highlighting the platform's broader support beyond model creation.



Absolutely, let's break down each point to ensure we cover all the details:

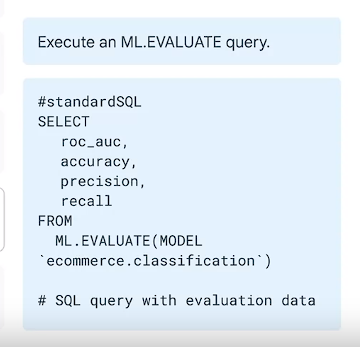
1. **Customer Lifetime Value (LTV):**
   * LTV estimates potential revenue/profit from customer behavior, crucial in marketing strategy.
   * Utilizes Google Analytics ecommerce dataset for the example, focusing on identifying high-value customers.
2. **Relevant Data Fields:**
   * Fields include customer lifetime pageviews, total visits, average time spent, total revenue, and ecommerce transactions.
   * These fields help in evaluating customer value based on behavior on the website.
3. **Preparing for Model Training:**
   * Emphasizes the importance of substantial data volume for effective model training.
   * Definitions: Examples/rows in the dataset, labels as correct answers derived from historical data for training future predictions.
4. **Labels and Model Types:**
   * Labels can be numeric (suitable for linear regression) or categorical (suitable for logistic regression).
   * Explains scenarios where revenue prediction might use linear regression while High Value Customer classification might use logistic regression.
5. **Understanding Features in ML:**
   * Data columns called features, similar to ingredients in a recipe, but too many can negatively impact the model.
   * Challenges in data sifting, assessing quality, collaborating for more features, and the role of feature engineering.
6. **BigQuery ML and Model Prep:**
   * Highlights BigQuery ML's role in automating tasks like one-hot encoding for categorical data.
   * Explains the critical step of dividing the dataset into training and evaluation subsets for model development.
7. **Predicting on Future Data:**
   * Shows how a model trained on historical data can predict outcomes for new, unlabeled data.
   * Illustrates informed decision-making based on model predictions for future datasets.

Absolutely, the phases of a machine learning project help streamline the process from data preparation to model deployment:

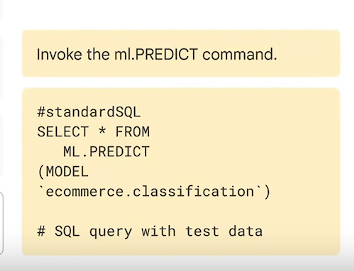
1. **Phase 1: Data Extraction, Transformation, and Loading (ETL):**
   * Involves extracting, transforming, and loading data into BigQuery, either from external sources or existing Google products like YouTube.
   * Emphasizes utilizing connectors for seamless data ingestion and enriching the data warehouse through SQL joins with additional data sources.
2. **Phase 2: Feature Selection and Preprocessing:**
   * Selects and preprocesses features, creating the training dataset using SQL to facilitate the model's learning process.
   * Highlights BigQuery ML's role in automating preprocessing tasks like one-hot encoding for categorical variables.(converts them into numerical data)
3. **Phase 3: Model Creation in BigQuery:**
   * Involves creating the model within BigQuery using the **CREATE MODEL** command, specifying its name, type, and providing a SQL query with the training dataset.
   * Executing the query initiates the model creation process.



1. **Phase 4: Model Evaluation:**
   * Evaluates the trained model's performance using the **ML.EVALUATE** query on an evaluation dataset.
   * Metrics like root mean squared error (for forecasting) and accuracy, precision, recall (for classification) are analyzed to assess model performance.



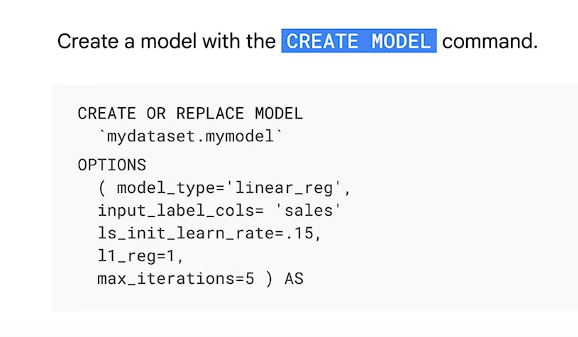
1. **Phase 5: Model Deployment and Prediction:**
   * Upon satisfactory model performance evaluation, the model is deployed for predictions.
   * Invoking the **ML.PREDICT** command on the trained model generates predictions with associated confidence levels, adding "predicted" to the field name in the label field.



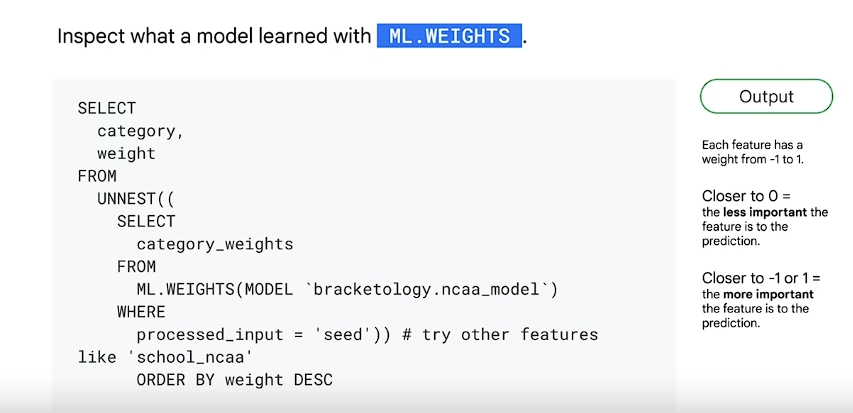
These phases represent a structured approach to a machine learning project, guiding the process from data preparation and model creation to evaluation and deployment for making predictions. Each phase focuses on a specific aspect, streamlining the workflow and ensuring the model's readiness for effective predictions.

Absolutely, here's a concise rundown of the key commands and aspects involved in using BigQuery ML for supervised machine learning models:

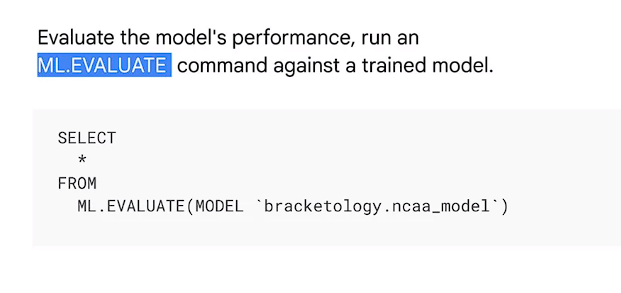
1. **Model Creation and Management:**

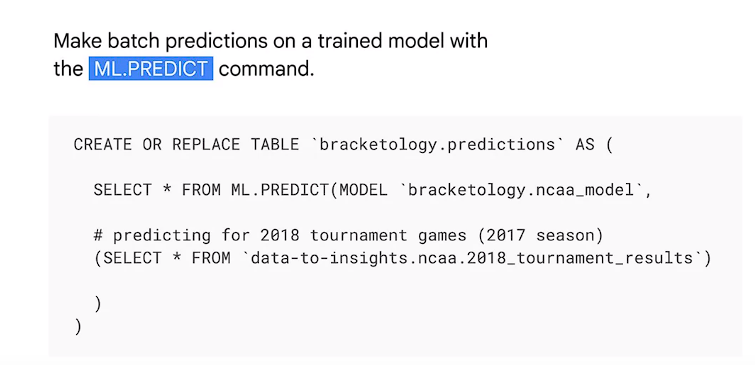


* + **CREATE MODEL**: Command to create a new model, specifying the model type and training dataset.
  + **CREATE OR REPLACE MODEL**: Overwrites an existing model with updated settings.
  + **ML.WEIGHTS**: Inspects what the model learned, showcasing feature importance via numerical values (-1 to 1).

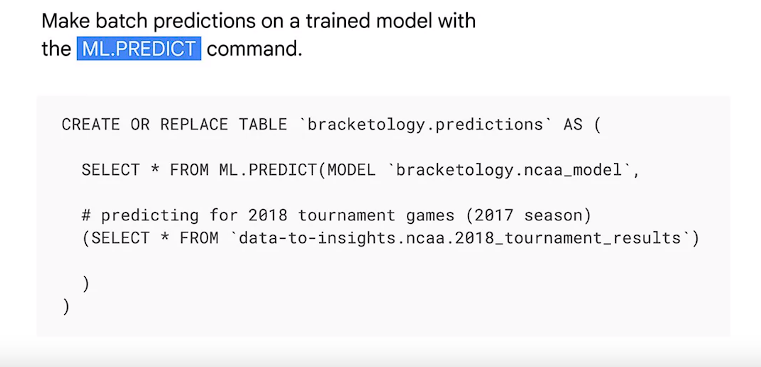


* + **ML.EVALUATE**: Evaluates model performance against an evaluation dataset, offering different metrics based on the model type.

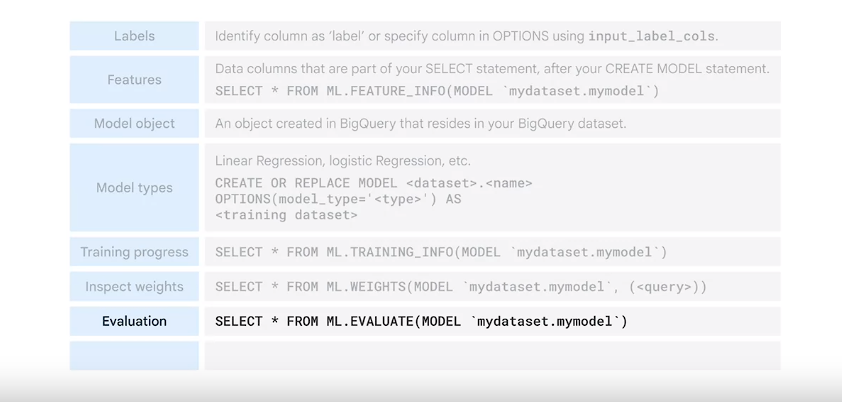


* + **ML.PREDICT**: Generates predictions using a trained model on a specific dataset.
  + 

1. **Model Components and Information:**
   * **label**: Field in the training dataset indicating the predicted outcome (required for supervised models).
   * **model features**: Data columns selected after the **CREATE MODEL** statement that constitute the features for model learning.
   * **ML.FEATURE\_INFO**: Provides statistics and metrics about columns for additional analysis.
   * **Model Objects**: Represent trained models in BigQuery, residing in the dataset. Displays information like last update and training run count.
2. **Choosing Model Types:**
   * **Linear Regression**: Suitable for numeric predictions (e.g., sales forecasting).
   * **Logistic Regression**: Apt for discrete classifications (e.g., high/medium/low or spam/not spam).
3. **Monitoring and Inspection:**
   * **ml.training\_info**: Tracks training progress, offers insights during and after model training.
   * **Inspecting Weights**: Reviewing the learned feature importance related to the predicted label.
4. **Prediction and Usage:**
   * **ml.predict**: Generates predictions from the trained model, referencing the model name and prediction dataset.



These commands and considerations cover the essential aspects of working with BigQuery ML, from model creation and evaluation to understanding feature importance and making predictions. They provide a comprehensive toolkit for leveraging supervised machine learning within BigQuery effectively.



Google Cloud's offerings for machine learning and introduces Vertex AI as a solution for ML challenges:

1. **Focus on Machine Learning Tools:**
   * Shifts attention from data engineering tools to machine learning options available within Google Cloud.
2. **Introduction to Vertex AI:**
   * Highlights Vertex AI as a solution to address machine learning challenges, hinting at its capabilities for building ML models.
3. **Google's AI Leadership:**
   * Establishes Google as an AI-first company, recognized as a leader in AI and ML across industries.
   * Cites Google's recognition in Gartner Magic Quadrant for Cloud AI Developer services in 2022 and other industry awards, showcasing its AI prowess.
4. **Google's AI Implementation:**
   * Illustrates Google's extensive implementation of AI technology in critical products and services over the past decade.
   * Provides an example of Smart Reply in Gmail, powered by natural language processing (NLP) as one of Google's AI-driven features.
5. **Democratization of AI:**
   * Emphasizes Google's goal to democratize AI, making advanced AI capabilities accessible to every company.
   * Highlights the aim to simplify AI model creation, leaving only steps requiring human judgment or creativity.
6. **Application of AI in Various Sectors:**
   * Encourages businesses in different sectors (e.g., travel, hospitality, retail) to consider AI and ML for addressing industry-specific challenges.
   * Suggests potential applications like aircraft scheduling, dynamic pricing for customers, and predictive inventory planning in respective fields.
7. **Prompt for Reflection:**
   * Encourages viewers to ponder how AI and ML might help solve problems within their businesses before proceeding to the next video.

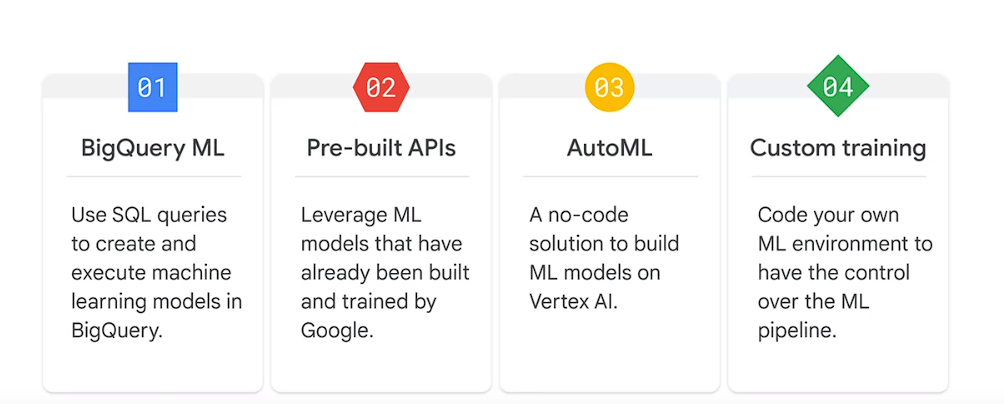
This section provides an overview of Google's AI leadership, its commitment to democratizing AI, and the potential of AI/ML applications across various industries. It primes the audience to consider how these technologies could address specific business challenges before delving deeper into Google Cloud's machine learning offerings.

Here's a breakdown comparing the four options offered by Google Cloud for building machine learning models:

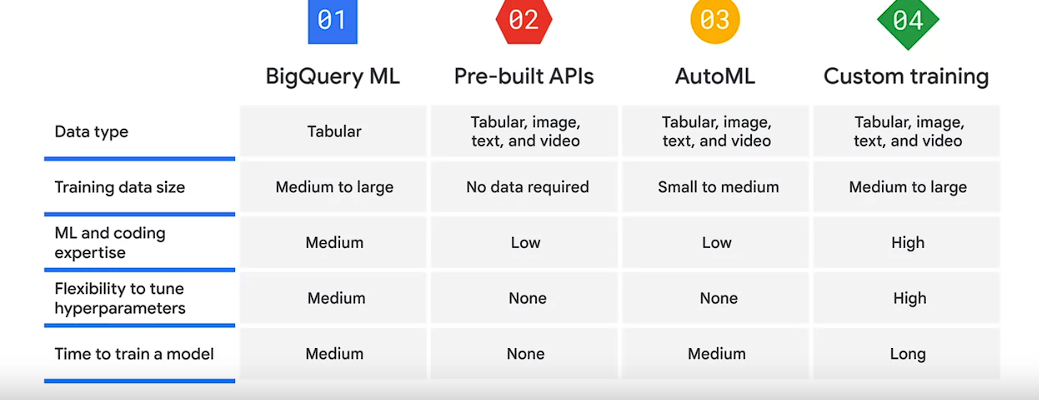
1. **BigQuery ML:**
   * **Data Type:** Supports tabular data.
   * **Training Data Size:** Requires a large amount of data.
   * **ML & Coding Expertise:** Requires understanding of SQL for model creation.
   * **Hyperparameter Tuning:** Allows experimentation with hyperparameters using SQL.
   * **Time to Train:** Depends on the project but utilizes SQL-based model development.
2. **Pre-built APIs:**
   * **Data Type:** Support for tabular, image, text, and video data.
   * **Training Data Size:** Doesn't require training data; utilizes pre-built models from Google.
   * **ML & Coding Expertise:** User-friendly, requires minimal ML expertise or model development effort.
   * **Hyperparameter Tuning:** Doesn't support hyperparameter tuning; uses pre-built models directly.
   * **Time to Train:** No training time; ready-to-use pre-built models for common tasks like vision, video, and natural language.
3. **AutoML (Vertex AI):**
   * **Data Type:** Supports tabular, image, text, and video data.
   * **Training Data Size:** Provides a no-code solution, allowing users to build their own models.
   * **ML & Coding Expertise:** Codeless solution, minimal coding required; focuses on business problems.
   * **Hyperparameter Tuning:** Automated model architecture and ML provisioning; limited control over tuning.
   * **Time to Train:** Offers a faster solution than custom training but more time-consuming than pre-built APIs.
4. **Custom Training (Vertex AI):**
   * **Data Type:** Supports tabular, image, text, and video data.
   * **Training Data Size:** Full control over model training and deployment, requires substantial coding.
   * **ML & Coding Expertise:** Requires significant ML expertise, full control over ML workflow.
   * **Hyperparameter Tuning:** Allows comprehensive control over hyperparameters and model architecture.
   * **Time to Train:** Takes the longest time to train models from scratch, offers full control over the ML pipeline.

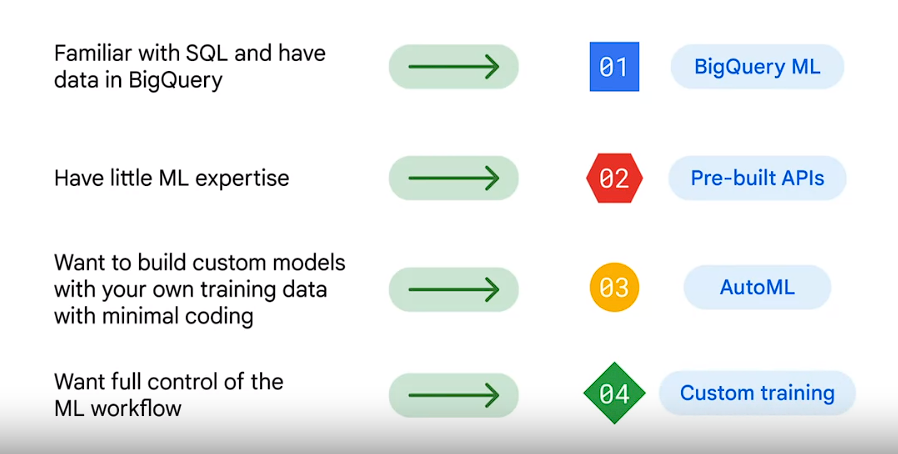
Choosing the best option depends on your business needs, available training data, and the level of ML expertise within your team. BigQuery ML suits those comfortable with SQL and existing data in BigQuery. Pre-built APIs cater to users with little ML experience, offering ready-to-use models. AutoML is for those seeking a codeless solution, focusing on business problems. Custom training on Vertex AI is ideal for ML engineers desiring complete control and flexibility in the ML workflow. The decision hinges on factors like data type, expertise, time constraints, and control over the ML process.

Subsequent videos will delve deeper into the details of the other three options to help users understand and choose the most suitable approach for their specific ML requirements.



Top of Form



Top of Form

Pre-built APIs provided by Google Cloud offer ready-to-use services, acting as valuable resources especially when substantial training data is not readily available. Here's a breakdown of some notable pre-built APIs:

1. **Speech-to-Text API:** Converts audio into text, facilitating data processing.
2. **Cloud Natural Language API:** Recognizes parts of speech, such as entities, and analyzes sentiment within text.
3. **Cloud Translation API:** Translates text from one language to another.
4. **Text-to-Speech API:** Converts text into high-quality voice audio.
5. **Vision API:** Recognizes and works with content in static images.
6. **Video Intelligence API:** Identifies motion and action within videos.

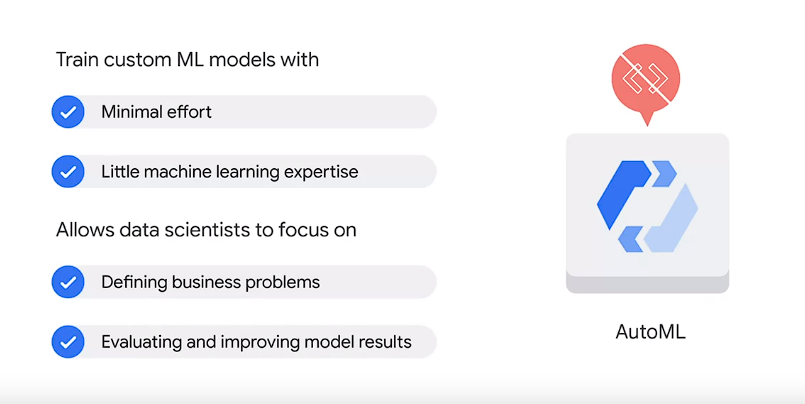
These APIs leverage models already trained by Google, utilizing vast datasets like Google's image datasets, YouTube captions, and Google's neural machine translation technology for training. The advantage lies in Google's extensive data resources and ML expertise, minimizing the effort required to build and train models from scratch.

Exploring these APIs can be done directly in a browser, allowing users to experiment with functionalities. For instance, navigating to cloud.google.com/vision in Chrome provides the opportunity to try out the Vision API by uploading an image. Similar experimentation is possible for the other ML APIs.

However, when moving towards building production models, using these APIs would entail passing JSON object requests to the API and parsing the returned responses to integrate the functionalities into the intended applications or workflows.

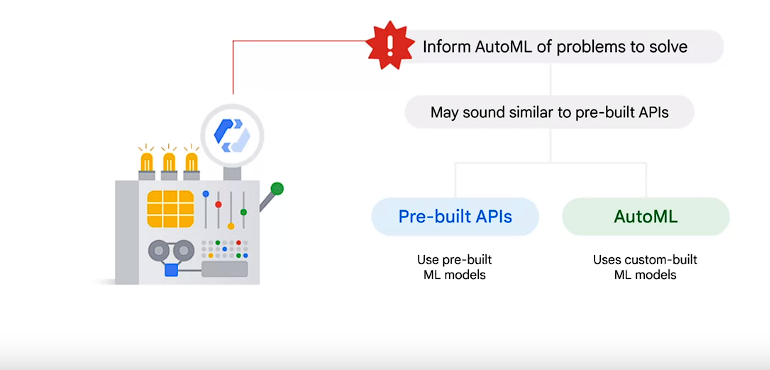
AutoML, or automated machine learning, is designed to streamline the creation of machine learning models by automating various steps involved in the process. It simplifies training and deploying ML models by utilizing technologies like transfer learning and neural architecture search. Here's a comprehensive overview of how AutoML operates and its capabilities across various data types and objectives:

1. **Technologies behind AutoML:**
   * **Transfer Learning:** Similar to building a library, transfer learning accumulates knowledge by leveraging pre-trained models trained on larger datasets. This technique enables achieving superior accuracy with less data and computation time.
   * **Neural Architecture Search:** This technology aims to find the most optimal model for a specific project, analogous to selecting the best book in a library for learning purposes.
2. **No-code Solution and Benefits:**
   * AutoML eliminates the need for extensive machine learning expertise by automating tasks such as hyperparameter tuning and model comparison.



* + It enables data scientists to focus on critical tasks like defining business problems and improving model results.

1. **Working of AutoML:**
   * Supports four types of data: **image, tabular, text, and video, a**ddressing different objectives.
   * Users can upload data from various sources **like Cloud Storage, BigQuery, or local machines.**

****

1. **Objectives and Solutions across Data Types:**

**Image Data:**

* + **Classification Model:** Identifies image content categories like detecting the presence of dogs or classifying dog breeds.
  + **Object Detection Model:** Locates objects within images with labels and bounding box annotations.

**Tabular Data:**

* + **Regression Model:** **Estimates numeric values like house prices** based on factors such as location and size. Rental p
  + **Classification Model:** Categorizes data into different levels of potential, such as for commercial real estate.
  + **Forecasting Model:** Predicts future numeric values using time-dependent historical data.

**Text Data:**

* + **Classification Model:** Sorts text into categories like redirecting customer queries to relevant departments.
  + **Entity Extraction Model:** Labels predefined entities like time and location within text. Label a social media post #tag
  + **Sentiment Analysis Model:** Identifies emotional tones in text as positive, negative, or neutral.

**Video Data:**

* + **Classification Model:** Categorizes video segments, like identifying different sports games.
  + **Object Tracking Model:** Detects and tracks objects in video segments, such as tracking a ball in soccer games.
  + **Action Recognition Model:** Identifies specific actions in video footage, like recognizing goals in soccer or touchdowns in football.

1. **Application and Flexibility:**
   * In real-world scenarios, combining multiple data types and objectives might be necessary to address complex business problems, and AutoML supports this flexibility.

AutoML empowers users to leverage machine learning effectively across diverse data types and objectives, providing a robust toolset for building custom machine learning models without extensive coding or ML expertise.

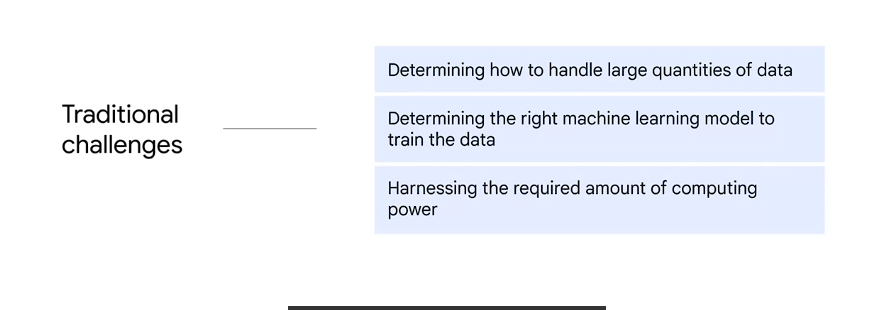
In the context of building machine learning models using Google Cloud's Vertex AI, custom training through Vertex AI Workbench is one of the options available. Here's a breakdown:

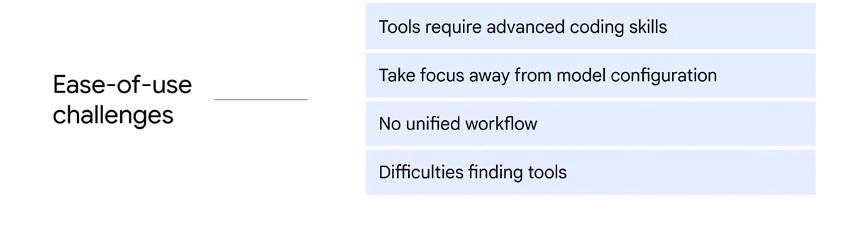
1. **Vertex AI Workbench:**
   * It's an integrated development environment (IDE) that covers the entire data science workflow, starting from data exploration to model training and deployment, all within a coding environment.
2. **Environment Selection:**
   * **Pre-built Container:** It's likened to a furnished kitchen, equipped with all the necessary dependencies and libraries (cookware) needed for the ML training code. These pre-built containers come with established platforms like **TensorFlow, PyTorch, Scikit-learn, or XGBoost, along with Python code to facilitate the work.** If your ML training is compatible with these platforms and libraries, a pre-built container could be the most suitable choice.
   * **Custom Container:** In contrast, it's similar to an empty kitchen without predefined tools. Here, you define and install the specific tools required for your machine learning task, allowing for a highly tailored environment that meets your precise needs.

The decision between these two container options typically depends on the specific requirements of your ML training. If your project aligns with the platforms and tools offered in pre-built containers, they offer a quicker and more straightforward setup. However, if you need a more tailored and specific environment, a custom container allows greater flexibility but requires defining and setting up the tools and dependencies from scratch.

Google's journey in developing machine learning and AI technologies has been extensive, from early initiatives like Scikit-learn in 2007 to the advanced Vertex AI platform today. As an AI-first company, Google has integrated AI into numerous products and services such as Gmail, Google Maps, Photos, and Translate.

However, the development of these technologies has faced various challenges, especially concerning machine learning model creation and deployment. Challenges include handling large datasets, choosing the right ML models, and acquiring sufficient computational power. Putting ML models into production further adds complexities like scalability, monitoring, continuous integration, and deployment.

 These difficulties often lead to project failures, with only about half of enterprise ML projects advancing past the initial pilot phase, according to Gartner.

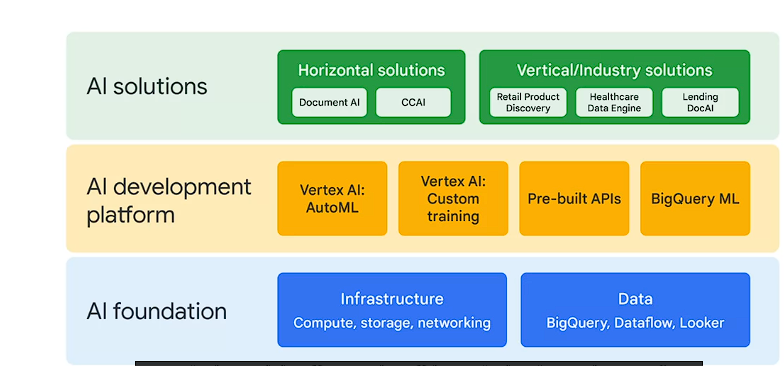


To address these challenges, Google introduced Vertex AI as a unified platform, consolidating various components of the machine learning ecosystem into a cohesive workflow. This unified platform approach entails:

1. **Data Readiness:** Users can upload data from diverse sources like Cloud Storage, BigQuery, or local machines. They can create processed features for the model and share them through the feature store.
2. **Model Training and Hyperparameter Tuning:** Once the data is prepared, users can experiment with different models and fine-tune hyperparameters.
3. **Deployment and Model Monitoring:** Users can set up pipelines to transition models into production, enabling automatic monitoring and continuous improvements.

Vertex AI offers options for building ML models: AutoML for an easy-to-use, codeless solution, and custom training for more control over the development environment and process. The platform's capabilities enable a wide range of tasks within a single environment, delivering several benefits summarized by four 'S's:

1. **Seamless:** Provides a smooth user experience from data upload to model training and deployment.
2. **Scalable:** ML Ops in Vertex AI helps manage ML production, automatically scaling storage and computing power as needed.
3. **Sustainable:** Features and artifacts created using Vertex AI are reusable and shareable.
4. **Speedy:** Vertex AI produces models with significantly fewer lines of code compared to competitors, improving efficiency and speed.



1. **AI Foundation:** This layer constitutes the infrastructure and data resources within Google Cloud.
2. **AI Development Platform:** Positioned in the middle layer, it includes:
   * **AutoML and Custom Training:** These options, available via Vertex AI, enable both codeless and code-based model development.
   * **Pre-built APIs:** These APIs grant access to existing machine learning models, offering a quick solution for various AI functionalities.
   * **BigQuery ML:** This tool allows the creation and execution of machine learning models using SQL within BigQuery.
3. **AI Solutions:** Representing the top layer, these solutions are classified into two categories:
   * **Horizontal Solutions:** Applicable across multiple industries:
     + **Document AI:** Uses computer vision, OCR, and NLP to create pretrained model and extract information from documents efficiently.
     + **Contact Center AI (CCAI):** Enhances customer service in contact centers by automating interactions and aiding human agents.
   * **Vertical or Industry Solutions:** Tailored solutions for specific sectors:
     + **Retail Product Discovery:** Facilitates superior search and recommendation experiences for retailers.
     + **Google Cloud Healthcare Data Engine:** Offers end-to-end healthcare analytics and insights.
     + **Lending DocAI:** Streamlines mortgage document processing for a more efficient home loan experience.

These AI solutions, serving diverse industry needs, encompass both broad, adaptable solutions and industry-specific offerings. They provide innovative tools and technologies to address challenges and leverage opportunities across various sectors. Visit cloud.google.com/solutions/ai to explore further.

In machine learning, the process differs from traditional programming by allowing machines to learn patterns from data rather than relying solely on predefined rules or algorithms. With traditional programming, data and rules lead to answers that computers execute. However, in machine learning, data and examples are provided to a machine learning model, allowing the system to learn and draw conclusions independently.

Machine learning involves three fundamental stages:

1. **Data Preparation:** This stage involves **uploading data and feature engineering**. Feature engineering is an essential step in preparing the data for model training and involves creating relevant features from raw data.
2. **Model Training:** The model undergoes iterative training and evaluation cycles. The model learns from the provided data during this stage, with continual refinement through training and evaluation iterations.
3. **Model Serving:** Once trained, the model is deployed, monitored, and managed to make predictions. This stage ensures that the model is used practically to predict results, making it operational.

This workflow analogy can be likened to running a restaurant:

* **Data Preparation:** Similar to preparing raw ingredients in a restaurant kitchen.
* **Model Training:** Analogous to experimenting with different recipes and continuously improving them.
* **Model Serving:** Finalizing the menu and serving meals to customers.

Vertex AI, Google's AI platform, supports this machine learning workflow through two main options: AutoML (codeless) and Custom Training (code-based). It offers various features accessible through both AutoML and Vertex AI Workbench:

* **Feature Store:** Centralizes and serves features for training models.
* **Vizier:** Aids in tuning hyperparameters in complex ML models.
* **Explainable AI:** Assists in interpreting training performance and model behaviors.
* **Pipelines:** Helps automate and monitor the ML production line.

These features streamline the machine learning process and offer tools for better model development, optimization, and deployment.

The AutoML workflow involves several critical stages, starting with data preparation:

1. **Data Upload:** When uploading data in the Vertex AI interface, naming the dataset and selecting the data type and objective are essential steps. Vertex AI supports four data types: **image, tabular, text, and video.** To ensure the correct selection, checking the data requirements is crucial. These requirements are available in the resources section of the course.
2. **Labeling:** Labels are crucial for training targets. If, for instance, you aim to distinguish between cats and dogs, providing labeled images is necessary. Labels can be manually added or obtained through Google's paid label service via **the Vertex console, where** human labelers generate accurate labels.
3. **Uploading Data:** Data can be sourced locally, from BigQuery, or Cloud Storage. The lab exercises will provide hands-on practice for these steps.

Once the data is in AutoML, the next step involves feature engineering, analogous to preparing ingredients before cooking:

* **Feature Engineering:** Just as ingredients need processing before cooking, data requires feature engineering before model training. Features are the contributing factors to predictions. These can be independent variables in statistics or columns in tables. This process can be challenging and time-consuming.

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.

To assist with feature engineering, Vertex AI offers the Feature Store:

* **Feature Store:** This is a centralized repository managing, organizing, and serving machine learning features. It consolidates diverse features from various sources into a central repository, simplifying their availability. The Feature Store automates feature aggregation, enhancing scalability.

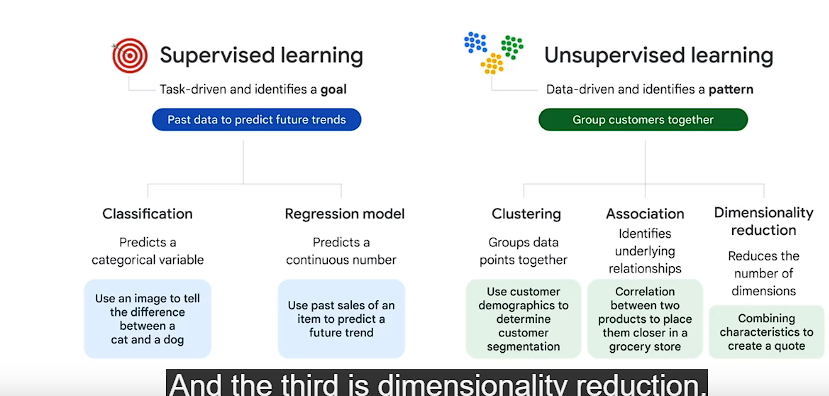
The benefits of Vertex AI Feature Store include:

1. **Shareable Features:** Features are shareable for various tasks, ensuring consistency across the organization.
2. **Reusable Features:** Reduces duplication efforts and saves time, particularly for high-value features.
3. **Scalable Features:** Offers low-latency serving, automatically scaling features for deployment.
4. **User-Friendly Interface:** Built on an intuitive user interface, making it easy to navigate and utilize.

By streamlining feature management and access, Vertex AI's Feature Store simplifies and enhances the efficiency of the feature engineering process.

Understanding the machine learning process is crucial for effective model training. Here's a breakdown of some essential concepts:

1. **Artificial Intelligence (AI) vs. Machine Learning (ML):**
   * **AI** encompasses all computer-based activities imitating human intelligence.
   * **ML** is a subset of AI, mainly dealing with supervised and unsupervised learning.
   * **Deep Learning** involves deep neural networks, adding layers between input and output for more in-depth learning.
2. **Supervised vs. Unsupervised Learning:**
   * **Supervised Learning:** Task-driven, providing labeled data points for a specific goal.
   * **Unsupervised Learning:** Data-driven, identifying patterns without labeled data points.
   * **Examples:**
     + *Supervised Learning:* Predicting sales trends with labeled sales data.
     + *Unsupervised Learning:* Grouping customers based on demographics without specific labels.
3. **Types of Supervised Learning:**
   * **Classification:** Predicts categories like distinguishing between cats and dogs in images.
   * **Regression:** Predicts continuous values, like future sales trends based on past sales data.
4. **Types of Unsupervised Learning:**
   * **Clustering:** Groups similar data points, like segmenting customers based on demographics.
   * **Association:** Identifies relationships, like product correlations for store placements.
   * **Dimensionality Reduction:** Reduces dataset dimensions, improving model efficiency.



1. **Role of Hyperparameters:**
   * **Hyperparameters** are user-defined settings guiding the machine learning process, acting as adjustable knobs.
   * Examples include learning rates, determining learning speed, and influencing model behavior.
2. **AutoML and Pre-built APIs vs. BigQuery ML and Custom Training:**
   * **AutoML and Pre-built APIs:** No need to specify machine learning models. Define objectives like text translation or image detection; Google's backend selects suitable models automatically.
   * **BigQuery ML and Custom Training:** Require specifying the model and setting hyperparameters manually.
   * **AutoML's Advantage:** Handles hyperparameter tuning automatically in the backend, leveraging neural architecture search to select optimal models.

Understanding these concepts is crucial in defining objectives, selecting suitable models, and setting hyperparameters for effective machine learning, especially when using Google Cloud's AutoML and pre-built APIs.

Top of Form

Understanding model evaluation metrics is vital in determining how well a machine learning model performs. Here's an elaboration on confusion matrices, recall, precision, and feature importance:

1. **Confusion Matrix:**
   * **Definition:** A table showcasing predicted vs. actual values in classification problems.
   * **Components:**
     + **True Positive:** Model predicts positive, and it's true.
     + **True Negative:** Model predicts negative, and it's true.
     + **False Positive (Type 1 Error):** Model predicts positive, but it's false.
     + **False Negative (Type 2 Error):** Model predicts negative, but it's false.
2. **Recall and Precision:**
   * **Recall:** Indicates how many positive cases were correctly predicted. Calculated as True Positives / (True Positives + False Negatives).
   * **Precision:** Reflects how many predicted positive cases are actually positive. Calculated as True Positives / (True Positives + False Positives).
   * **Trade-off:** Precision and recall often have a trade-off relationship.
3. **Example for Recall and Precision:**
   * **Recall:** Catching 80 fish out of 100 total fish in the lake with a wide net results in a recall of 80%.
   * **Precision:** If the net catches 80 fish out of 160 total fish and rocks, precision becomes 50%.
4. **Adjusting Precision and Recall:**
   * **Trade-off Scenario:** Optimizing for one may affect the other based on specific use cases.
   * **Example:** Gmail's spam filter might prioritize recall to catch more potential spam or emphasize precision to avoid blocking non-spam emails.
5. **Visualization in Vertex AI:**
   * **Precision-Recall Curve:** Visualizes precision and recall trade-offs for problem-solving adjustments.
6. **Feature Importance in Vertex AI:**
   * **Definition:** Illustrates the contribution of each feature to predictions via a bar chart.
   * **Larger Contribution:** Longer bars or larger numerical values indicate greater importance.
   * **Role:** Guides decisions on including features in machine learning models.
7. **Explainable AI:**
   * **Definition:** A Vertex AI feature providing tools and frameworks to interpret and understand machine learning model predictions.

Understanding these metrics is pivotal for assessing a model's effectiveness, making necessary adjustments, and interpreting predictions, all of which are supported by Vertex AI's comprehensive functionalities like Explainable A

Model serving, the final stage of the machine learning workflow, parallels the process of serving a meal in a restaurant. Here's a breakdown of model serving, including model deployment, model monitoring, and the relevance of MLOps:

1. **Model Deployment:**
   * **Analogy:** Similar to serving the meal to a customer in a restaurant.
   * **Purpose:** Implementation of the trained model.
   * **MLOps Role:** Offers best practices to automate deployment.
   * **Options:**
     + **Endpoint Deployment:** Provides immediate results with low latency, suitable for real-time recommendations based on user behavior.
     + **Batch Prediction Deployment:** Useful for processing accumulated data periodically, like sending ads based on recent purchasing behavior.
     + **Offline Prediction Deployment:** Deploys the model in a specific non-cloud environment.
2. **Model Monitoring:**
   * **Analogy:** Similar to checking restaurant operations for efficiency.
   * **Purpose:** Continuous monitoring of the deployed model's performance.
   * **MLOps Role:** Utilizes Vertex AI Pipelines to automate, monitor, and govern machine learning systems.



* + **Functionality:** Vertex AI Pipelines orchestrates workflows in a serverless manner, providing real-time alerts based on predefined thresholds.
  + **Using Vertex AI Workbench:** Enables the definition of custom pipelines using prebuilt components, simplifying the pipeline creation process.

1. **MLOps Significance:**
   * **Definition:** MLOps combines machine learning development with operational principles akin to DevOps.
   * **Purpose:** Addresses production challenges in machine learning by automating and monitoring ML system construction.
   * **Focus Areas:** Emphasizes continuous integration, continuous training, and continuous delivery, accommodating the evolving nature of data and code in ML.

The restaurant analogy effectively illustrates the significance of MLOps and the process of model serving. MLOps streamlines the deployment of models, ensuring they are effectively implemented and monitored, akin to serving a meal seamlessly to a hungry customer and maintaining the restaurant's operational efficiency.

Top of Form

Top of Form

Top of Form

Top of Form

Top of Form

//Lab Notes

Vertex AI offers two options on one platform to build a ML model: a codeless solution with **AutoML** and a code-based solution with **Custom Training** using Vertex **Workbench**. You use **AutoML** in this lab.

**ntroduction to Vertex AI**

* **Platform**: Vertex AI, Google Cloud's unified AI platform.
* **Purpose**: Train and deploy ML models, **offering codeless (AutoML)** and **code-based (Custom Training) solutions.**
* **Task**: Building an ML model to predict loan repayment by customers.

**Task 1: Prepare Training Data**

* **Initial Dashboard**: Illustrates major stages - prepare training data, train model, and get predictions.
* **Dataset Creation**:
  + Name: LoanRisk
  + Data Type/Objective: Tabular, Regression/Classification
* **Data Upload**:
  + Source: Cloud Storage
  + Import Path: **spls/cbl455/loan\_risk.csv**
* **Optional**:
  + Generate statistics for descriptive insights.

**Task 2: Train Your Model**

* **Training**: Using AutoML for classification.
* **Model Details**:
  + Name: LoanRisk
  + Target Column: Default
* **Training Options**:
  + Excluding irrelevant columns like ClientID.
  + Budget: 1 compute hour, with early stopping enabled.

**Task 3: Evaluate Model Performance**

* Metrics: Precision/Recall curve, Confusion Matrix, Feature Importance.
* **Precision/Recall Curve**: Adjustment of confidence threshold for trade-off between precision and recall.
* **Confusion Matrix**: Evaluates model accuracy in predicting repay and default cases.
* **Feature Importance**: Illustrates each feature's contribution to predictions.

**Task 4: Deploy the Model**

* Demonstration steps for deploying the trained model to an endpoint in Vertex.
* Endpoint creation, machine type selection, explainability options, and model monitoring.

**Task 5: SML Bearer Token**

* Retrieving Bearer Token for pipeline authentication.
* Instructions for obtaining the token within a specified timeframe.

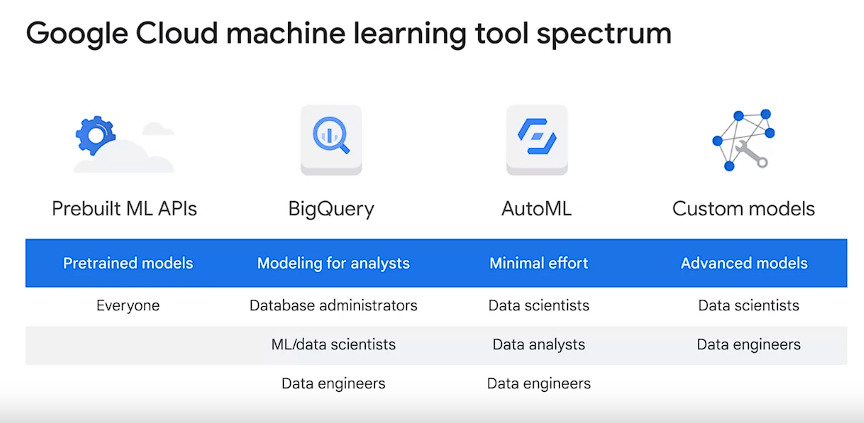
**Task 6: Get Predictions**

* Set environment variables for SML service.
* Steps to download lab assets, create environment variables, and perform requests to the SML service for predictions.
* Example scenarios for testing model predictions and interpreting responses.

**Conclusion**

* Overview of using Vertex AI functionalities:
  + Dataset upload, AutoML training, performance evaluation, model deployment, and predictions.

**Notes on MLOps Fundamentals: Vertex AI & Google's Unified AI Platform**

****

**Notes on MLOps Fundamentals: Vertex AI & Google's Unified AI Platform**

Introduction:

* **Vertex AI Overview**: Introduction to Vertex AI, Google's unified AI platform.
* **Role in MLOps**: Exploring how Vertex AI contributes to MLOps workflows.
* **Expertise Requirement**: Recognizing the need for expertise in both workflow and products for ML model production.

Key Machine Learning Constructs:

* **Dataset Creation**:
  + Data ingestion, analysis, and cleaning for dataset formation.
  + Methods like ETL (Extract, Transform, Load) or ELT (Extract, Load, Transform) used.
* **Model Training**:
  + Experimentation with feature processing, model architecture, and hyperparameter tuning.
  + Iterative model adjustments due to new data, code alterations, or scheduled updates.
* **Model Evaluation and Deployment**:
  + Comparative analysis with existing model versions.
  + Deployment for both online and batch predictions.
* **MLOps Framework**: Entire ML model creation process termed as MLOps, as discussed earlier.

Data Impact on Pipeline:

* **Data Influence**:
  + Data type (JPEG, TensorFlow records) and storage location (Cloud Storage, BigQuery) impact the pipeline.
* **Model Deployment Variability**:
  + Deployment differences between TensorFlow and PyTorch models.
  + Deployment processes vary based on AutoML or custom TensorFlow models.

Vertex AI Features:

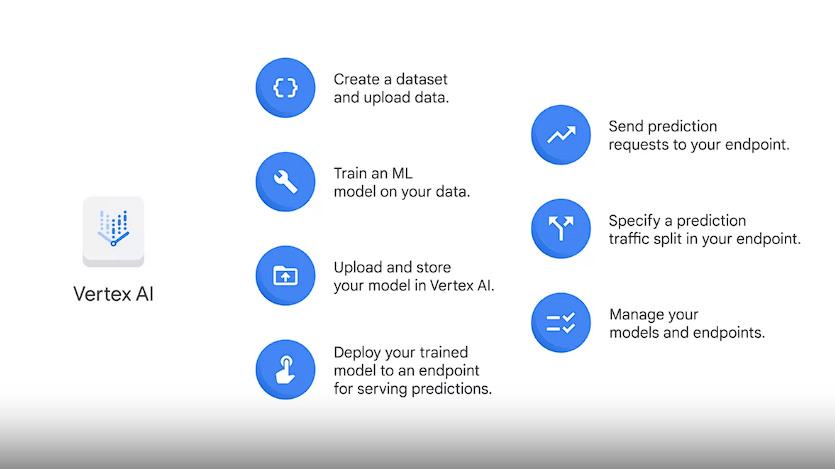
* **Unified Platform Benefits**:
  + Vertex AI consolidates Google Cloud services for ML and AI in a unified platform.
  + Aids enterprises in leveraging data for value and accelerates time-to-value.
* **Workflow Unification**:
  + Unifies certain parts of the ML workflow.
  + Manages structured/unstructured datasets with metadata, stored in Cloud Storage or BigQuery.
* **Training Pipeline**:
  + Steps to train ML models using datasets.
  + Containerization for generalization, reproducibility, and auditability.
* **Model Building and Endpoint Usage**:
  + Building an ML model within the training pipeline or loading from compatible external sources.
  + Endpoint invocation for online predictions and explanations, managing multiple models and versions.

Vertex AI Operations:

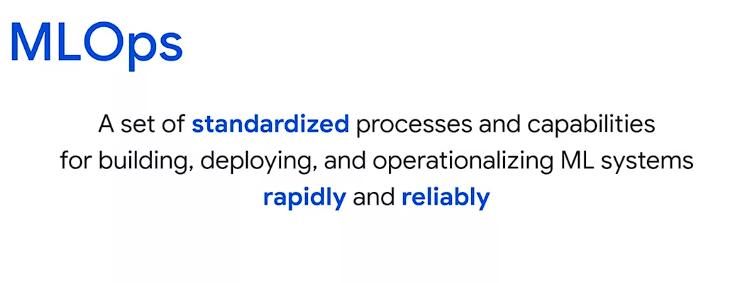
* **Interface Functionality**:
  + Direct management via the UI for various stages of ML workflow.
  + Includes dataset creation, model training, evaluation, hyperparameter tuning, model storage, deployment to endpoints, prediction requests, traffic split specifications, and model/endpoint management.
* **Flexibility in Training**:
  + Options for AutoML (minimal technical effort) or custom training tailored for specific outcomes.
* **Summary of Vertex AI Benefits**:
  + Facilitates fast experimentation, accelerated deployment, and simplified model management to achieve ML goals.

Conclusion:

* **Platform Utility**:
  + Vertex AI enables ML model development experimentation and quick deployment for achieving ML objectives.
* **Additional Information**:
  + Encouragement to explore more Vertex AI courses in the ML training catalog at cloud.google.com/training.



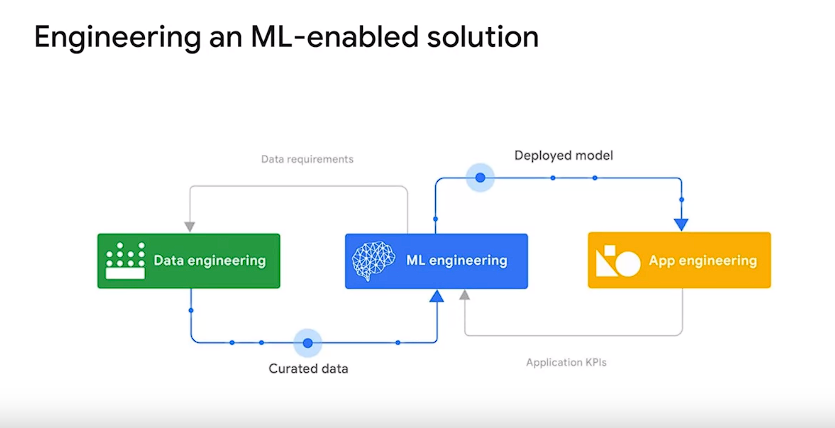
**Introduction to Vertex AI and its Role in MLOps:**

****

* **Vertex AI Overview**: Google's unified AI platform introduced, addressing time-consuming aspects of ML model training and deployment.
* **Challenges in ML Model Iteration**: Necessity for repetitive data addition, feature trials, model variations, and parameter tuning.
* **Building ML Engineering Culture**: Emphasis on establishing an ML engineering culture and capability within organizations.
* **MLOps Concept and Significance**: Defined as standardized processes and technology capabilities for rapid and reliable ML system construction and operation.
* **MLOps Goals**: Automation, monitoring, and maintenance of aligned versions of data, models, code, and components.

**Understanding MLOps' Alignment with Engineering Disciplines:**

* **ML Engineering**: A subset of software engineering tailored for practical ML application.
* **MLOps Methodology**: Focused on operationalization and automation of model development, training, deployment, and governance processes.
* **Data Engineering**: Concentration on data ingestion, management, and processing for dataset and feature preparation.
* **App Engineering**: Concerns with designing, developing, and integrating ML models into applications.



* Deploying ML models and integrating them with your applications and systems is the main subject of app engineering when you build an ML application.
* ML engineering also produces the models deployed by the app engineering team, which in turn shares the applications key performance indicators or KPIs with the ML engineering team

**Integration of ML Engineering with Data and App Engineering:**

* **Interdisciplinary Collaboration**: Iterative process between ML, data, and app engineering disciplines.
* **Feedback Loop**: ML engineering utilizing data outputs from data engineering and providing data requirements while deploying models developed by app engineering.

**MLOps Lifecycle Overview:**

* **Integration and Lifecycle Phases**: Utilization of existing investments in DataOps and DevOps.
* **Six Iterative Processes**:
  1. **ML Development**: Experimentation and prototyping of ML models.
  2. **Training Operationalization**: Checking model viability(works or not) for production.

This includes testing internal and external data connections and configurations and putting the model routine and a stable operational phase after the first experimentation phase is performed.

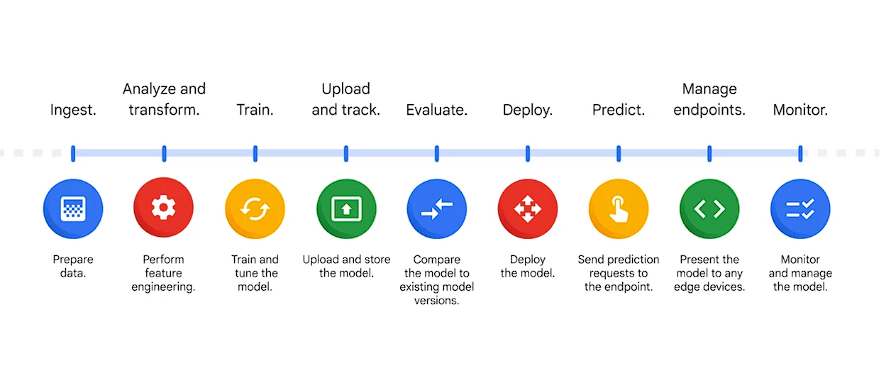
* 1. **Continuous Training**: Retraining production models with new data.
  2. **Model Deployment**: Continuous integration and delivery of models occur.
  3. **Prediction or Inference Serving**: Hosting models as services to serve online predictions or as part of a batch prediction system to serve offline predictions..
  4. **Continuous Monitoring**: Identifying model performance issues and data anomalies.

**Central Elements: Data and Model Management:**

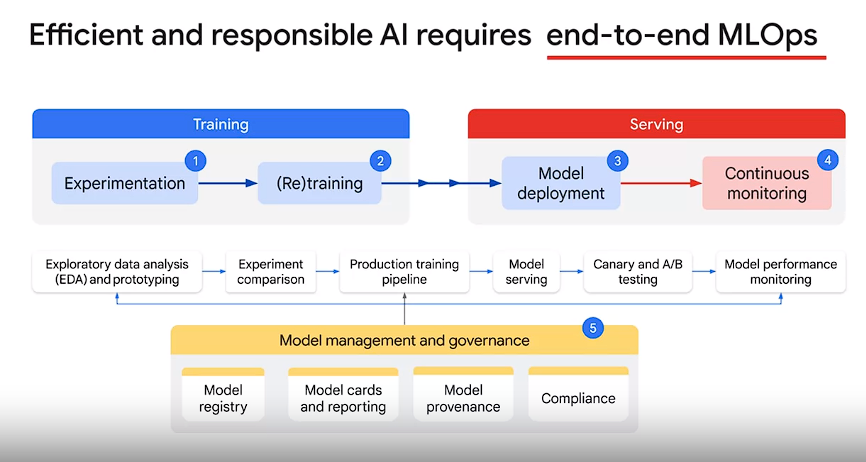
* **Governance Functions**: Support for auditability, traceability, compliance, shareability, reusability, and discoverability of ML assets.
* **Artifact Definition**:
* An artifact is a discrete entity or piece of data produced and consumed by an ML workflow such as datasets, models, input files and training logs. Note that data and model management processes are transitional phases.

Data and model management transitional phases within the MLOps lifecycle.

**Vertex AI's Automation and Roles in MLOps:**

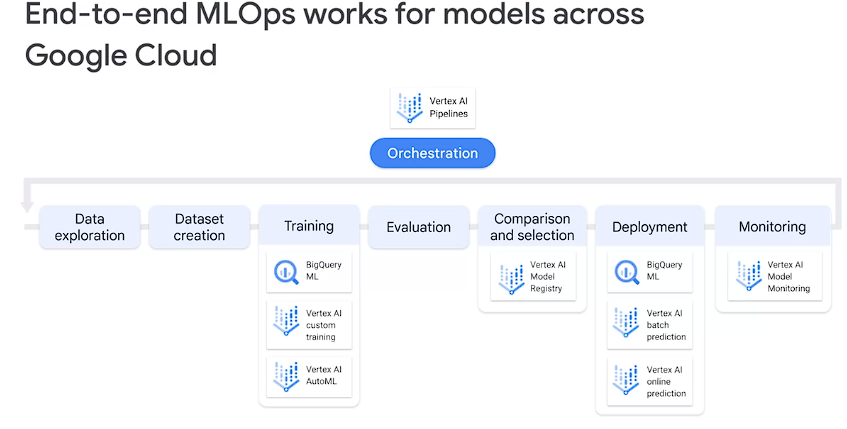
****

* **Automated Steps**: Vertex AI automates several MLOps steps, **including data preparation, feature engineering, model serving, and deployment to edge devices.**
* **User Diversity**: Vertex AI caters to various user needs within an organization, serving product managers, data analysts, and data engineers.
* **Four Main Reasons for Vertex AI Suitability**:
  1. **Unified Data and ML Platform**: Enhanced connections between data and ML, leveraging Google Cloud services.
  2. **End-to-End MLOps**: Efficient management, monitoring, governance, and explanation throughout the development lifecycle.



* 1. **Flexibility**: Open and scalable ML infrastructure for framework and hardware flexibility, accelerating model velocity.
  2. **Integration with Google Resources**: Utilization of Google's extensive research to streamline AI use cases with optimal infrastructure.

**Vertex AI's Features and Integration:**

* **Access to Advanced AI Algorithms**: Access to state-of-the-art AI from Google research and DeepMind.
* **Unified AI Platform**: Brings together Google Cloud services, covering every ML tool needed, including MLOps capabilities and scalable infrastructure.
* **Open Source Framework Integration**: Integrates with TensorFlow, PyTorch, scikit-learn, and supports all ML frameworks and AI branches through custom containers for training and prediction.
* Top of Form

**Vertex AI Capabilities:**

1. **Automated ML Processes**:

* **Objective**: Designed to streamline ML workflows, preventing practitioners from starting from scratch.
* **Advantages**:
  + **Faster Scaling**: Efficient scaling of ML models leveraging Vertex AI's managed infrastructure.
  + **Quick ML Environment Setup**: Rapid establishment of ML environments.
  + **Orchestration Automation**: Automated orchestration facilitating management of large clusters and low-latency applications.

2. **Diverse AI and ML Toolset**:

* **Catering to User Skill Sets**: Offers a wide range of tools suitable for users with varying levels of expertise.
* **Built-in MLOps Capabilities**:
  + **Impact on Enterprises**: Facilitates improved insights, predictive analysis, and automation of core business processes.

3. **Management and Governance of ML Models**:

* **Leveraging Google Cloud Services**:
  + **Simplification of MLOps Processes**: Uses Google Cloud Managed Services to streamline and simplify MLOps.
* **Key Capabilities**:
  + **Explanation and Insights**: Reveals explanations behind models and predictions.
  + **Performance Monitoring**: Allows monitoring of data and model performance.
  + **Experiment Tracking and Analysis**: Tracks and compares multiple experiment runs, analyzes primary model metrics.

**Vertex AI's Key Components:**

1. **Managed Infrastructure**:

* **Enables Efficient Scaling**: Provides managed infrastructure allowing rapid and efficient scaling of ML models.

2. **ML Environment Setup Automation**:

* **Quick Environment Configuration**: Automates the setup of ML environments for rapid initiation.

3. **Orchestration and Cluster Management**:

* **Efficient Cluster Management**: Allows easy management of large clusters and facilitates the setup of low-latency applications.

4. **Diverse AI and ML Toolset**:

* **Adaptability to Skill Levels**: Offers tools catering to users with different skill sets, enabling seamless utilization.

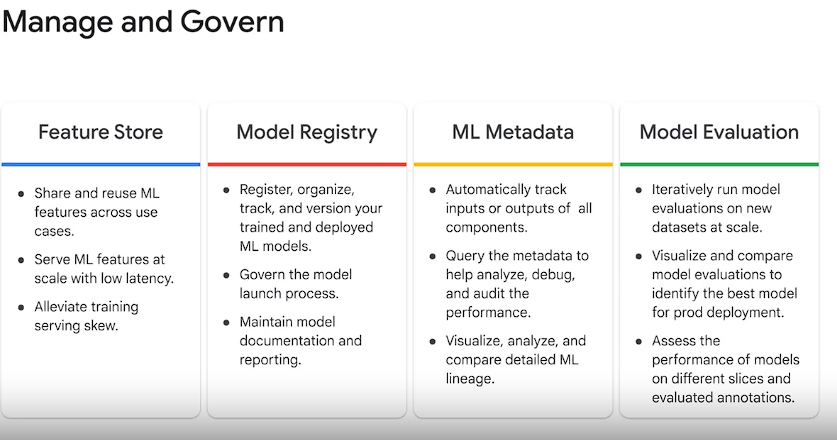
5. **MLOps Capabilities Integration**:

* **Enhanced MLOps Processes**: Integrates with Google Cloud Managed Services to streamline MLOps activities, such as explanation tracking, performance monitoring, and experiment analysis.

**Conclusion:**

Vertex AI's comprehensive suite of tools, managed infrastructure, and built-in MLOps capabilities empower users to efficiently manage, govern, and scale ML models while providing insights and facilitating predictive analysis for enterprises across their organizations.

**Managing and Governing Capabilities:**

****

1. **Feature Management with Vertex AI Feature Store**:

* **Purpose**: Create and manage ML features, share/reuse across use cases, and serve at scale with low latency.
* **Benefits**: Alleviate training-serving skew, improve scalability, and facilitate feature management.

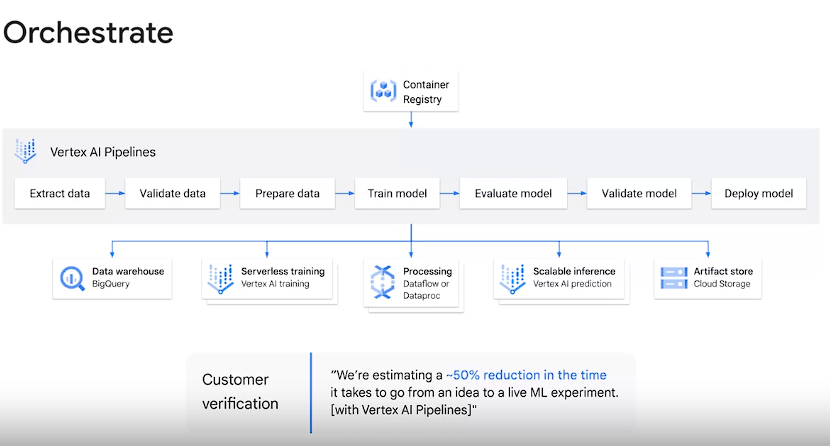
2. **Model Management through Vertex AI Model Registry and ML Metadata**:

* **Vertex AI Model Registry**:
  + **Function**: Central repository for managing ML model lifecycles.
  + **Capabilities**: Register, organize, track, version trained/deployed models, govern launch process, and maintain documentation/reporting.
* **Vertex ML Metadata**:
  + **Function**: Record metadata and artifacts produced by ML systems.
  + **Capabilities**: Automatic tracking of inputs/outputs, analysis, debugging, auditing, and visualization of detailed ML lineage.

3. **Model Evaluation**:

* **Evaluation via Vertex AI**:
  + **Approaches**: Creation through Model Registry in Google Cloud Console or utilizing Model Evaluation feature as a pipeline component within Vertex AI pipelines.

**Orchestrating ML Workflows:**

****

1. **Vertex AI Pipelines for Simplified ML Operations**:

You can simplify ML operations by using vertex AI pipelines to automate, monitor, and govern your ML systems.

* **Purpose**: Automate, monitor, and govern ML systems via serverless orchestration.
* **Key Features**:
  + **Leveraging Google Cloud Managed Services**: Utilize services like BigQuery, Vertex Training, or Dataflow for workflow orchestration.
  + **Portability and Scalability**: Container-based portable and scalable ML workflows.
* **Benefits**: Faster iteration, enhanced productivity, easier transition to production, and increased independence for ML practitioners.
* **Components**:
  + **ML Pipelines Structure**: Composed of input parameters and steps represented by pipeline components.
  + **Facilitated Iteration and Production Transition**: Empowers ML practitioners to focus on building solutions instead of infrastructure.

2. **Applying MLOps Strategies via ML Pipelines**:

* **MLOps Automation and Monitoring**:
  + **Strategies**: Automation, monitoring, and application of MLOps principles to repetitive processes.
* **Use Cases**:
  + **Experimentation**: Run ML workflows with different hyperparameters.
  + **Workflow Reusability**: Reuse workflows for training new models.
* **Pipeline Flexibility**:
  + **Using Vertex AI Pipelines**: Run pipelines built with Kubeflow Pipelines SDK or TensorFlow Extended (TFX).

**Conclusion:**

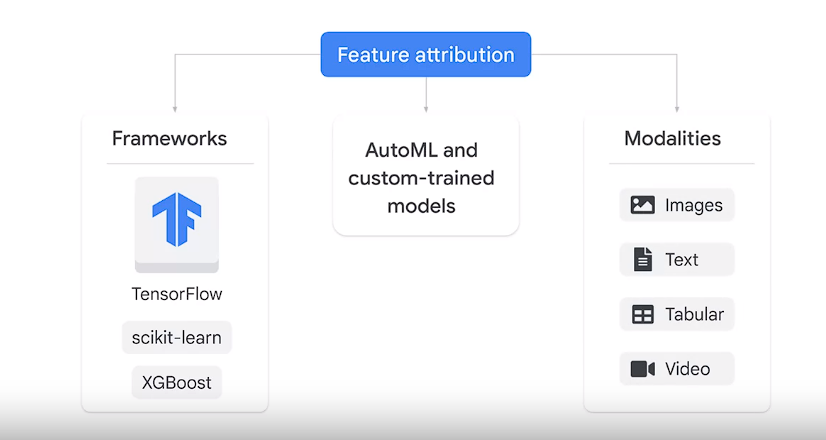
* **Understanding Model Behavior**:
  + Next video will delve into understanding model behavior for a deeper understanding of the model's performance and characteristics.

**Reading Material:**

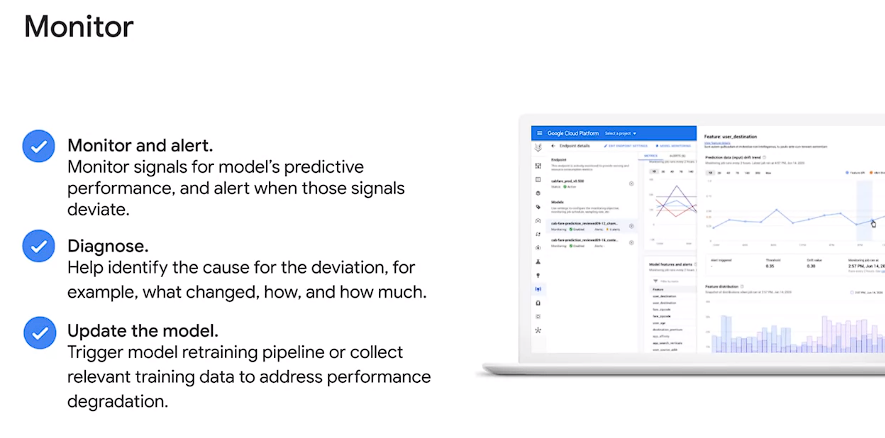
* **Choosing Between Kubeflow Pipelines SDK and TFX**:
  + Provides insights into the selection between Kubeflow Pipelines SDK and TFX for specific use cases within the ML workflow orchestration.

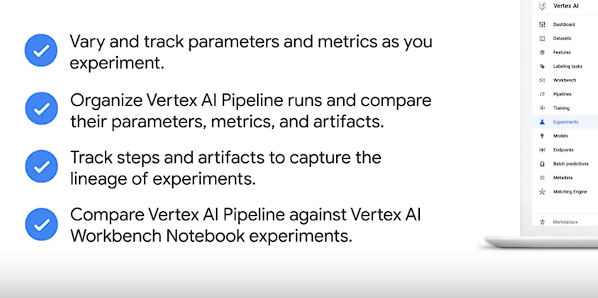
Top of Form

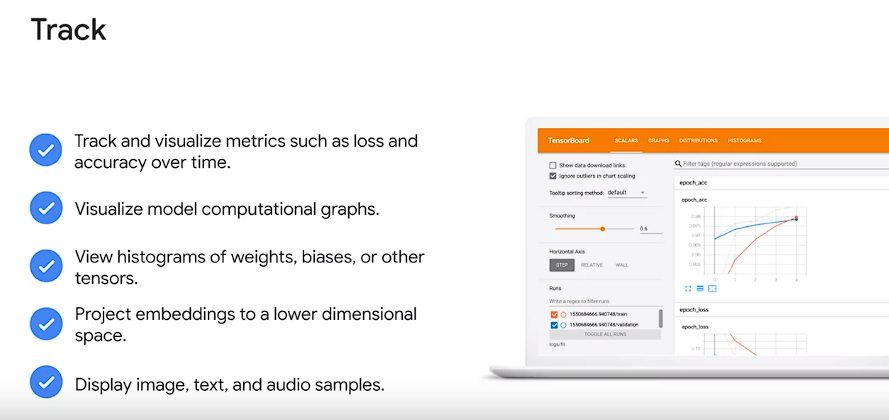
* Vertex Explainable AI: Understand model outputs for classification and regression, offering feature-based explanations integrating feature attributions. Reveals feature contributions and model biases using sampled shapeley, integrated gradients, or XRAI methods.
* Attribution Methods: Based on cooperative game theory (Shapley values), assigning proportional credit to each feature, treating them as 'players' influencing prediction outcomes.



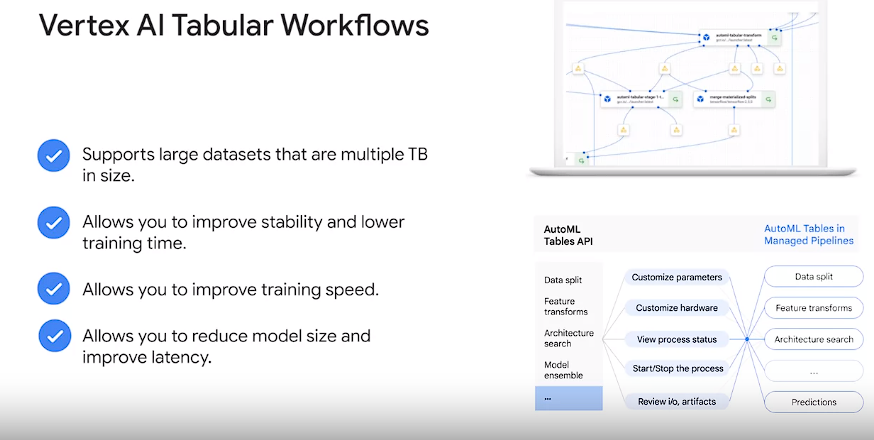
* Model Performance Monitoring: Vertex AI's model monitoring detects feature skew and drift in input data, ensuring optimal model performance by comparing feature distributions in production against training data.



* Skew vs. Drift Detection: Skew occurs when feature data in production differs from the training data, while drift signifies significant changes in feature data distribution over time. Detection is enabled based on availability of original training data.
* Experiment Tracking and Comparison: Vertex AI experiments aid in tracking, analyzing, and comparing various model architecturess tensorflow, pytorch, or psychic learn, different hyper-parameters and different training environments. s. Utilizes contexts to group artifacts and executions, offering detailed visualizations via Vertex AI TensorBoard.
* 
* Recall then artifact is a discrete entity or piece of data produced and consumed by an ML workflow. A context is used to group artifacts and executions together under a single queriable and taped category. Context can be used to represent sets of metadata. An example of a context can be a run of ML pipeline. Vertex AI experiments is a context in Vertex ML metadata where an experiment can contain an experiment runs in addition to an pipeline runs. An experiment run consists of parameters, summary metrics, time series metrics, artifacts, executions and Vertex AI resources such as, pipeline job, which is created when users want to run an ML pipeline on Vertex AI pipelines. Another tool that lets you track, visualize, and compare ML experiments and share them with your team is Vertex AI TensorBoard. Open-source TensorBoard, TB is a Google open-source project for ML experiment visualization. Vertex AI TensorBoard is an enterprise ready managed version of tensorboard. Executions and artifacts of a pipeline run are viewable in the Google Cloud Console
* Vertex AI TensorBoard: Enterprise-ready managed version of TensorBoard, showcasing detailed visualizations like metrics tracking, computational graphs, histograms, embeddings projection, and media sample displays.



* Vertex AI Tabular Workflows: Managed pipelines for end-to-end ML with tabular data using Google's AutoML. Supports large datasets, offers stability improvement, training speed enhancement, model size reduction, and transparency in customization options and inspection of AutoML components.



The following are some of the benefits of tabular workflow for end-to-end AutoML. Supports large datasets that are multiple terabytes in size. Lets you improve stability and lower training time by limiting the search space of architecture types or skipping architecture search. Lets you improve training speed by manually selecting the hardware used for training and architecture search. Lets you reduce model size and improve latency with distillation or by changing the ensemble size. Additionally, each AutoML component can be inspected in a powerful pipelines graph interface that lets you see the transformed data tables, evaluated model architectures, and many more details.

* Modular MLOps Tools: Tools for collaboration, model improvement, monitoring, alerting, diagnosis, and explanations, designed to be integrated into existing systems as needed.

Top of Form