**MBUSA Sprint Project Report  
Data Pipeline Team**

**Introduction:**

A data pipeline is a set of processes that extract, transform, and load data from various sources to a target system, such as a database or a data warehouse. Data pipelines are used to automate the flow of data between different systems and ensure that data is accurate, consistent, and available in real-time.

Data pipelines are crucial for modern businesses as they help organizations make data-driven decisions, improve operational efficiency, and provide insights that lead to better customer experiences. Data pipelines can be used for a variety of purposes, including marketing analytics, fraud detection, inventory management, and many more.

**Real Use Cases of Data Pipelines**

One real-world example of a data pipeline is a retail company that wants to improve its inventory management system. The company may use a data pipeline to extract data from various sources, such as sales data, inventory data, and customer data, and then use this data to optimize its inventory levels, reduce stock-outs, and increase profitability.

Another example is a healthcare organization that wants to improve patient outcomes. The organization may use a data pipeline to collect data from electronic health records, medical devices, and other sources, and then use this data to develop predictive models that can identify patients who are at risk of developing certain conditions. This information can be used to develop personalized treatment plans and improve patient outcomes.

**Introduction to Anomaly Detection Pipeline**

Our anomaly detection pipeline is a comprehensive solution that automates the process of identifying anomalies in large datasets. The pipeline uses GCP cloud services to extract data from SAP paragon systems, perform machine learning modelling to generate datasets with outliers and inliers, and deploy the data to a database and dashboard. The pipeline is designed to be scalable, reliable, and easy to use.

**Diagram

Description automatically generated with low confidence**

**Data Pipeline Process:**

**Source Files (MARA, MVKE):**

The first stage of the data pipeline is the source files. In our case, we extract data from SAP paragon systems. These source files are the initial data inputs for the pipeline and contain the raw data that needs to be processed.

**Extraction:**

In the extraction stage, we use GCP cloud services to extract data from the source files and load it into a GCP cloud storage bucket. The extraction process involves connecting to the source system, reading the data from the source files, and transferring it to the cloud storage bucket.

We can use various GCP tools, such as Cloud Storage Transfer Service, to extract data from various sources like databases, APIs, and flat files. The extracted data can be in various formats like CSV, JSON, or Parquet.

**Transformation (ML Modeling):**

In this stage, we use machine learning modelling to transform the data into datasets with outliers and inliers. This stage involves several sub-stages:

a) Data Preparation: In this sub-stage, we prepare the data for modelling by performing tasks like data cleaning, data normalization, data integration, and data feature engineering.

b) Modelling: In this sub-stage, we apply machine learning algorithms to the prepared data to develop predictive models that can identify anomalies in the data.

**Load to Database:**

After transformation, we load the data into a database for storage and analysis. The database can be an SQL database, NoSQL database, or a data warehouse.

We can use various GCP tools, such as Cloud SQL, Bigtable, or BigQuery, to load the transformed data into the database. The transformed data can be in various formats like CSV, JSON, or Parquet.

**Load to Dashboard:**

Finally, we load the data into a dashboard for data specialists to analyze the outliers. The dashboard can be a web application or a desktop application that provides interactive visualizations and reports to the data specialists.

We can use various GCP tools, such as Data Studio or Looker, to build and deploy the dashboard. The dashboard can be customized to meet the specific needs of the data specialists.

**Cloud Platform:**

Google Cloud Platform (GCP) is utilized as our Cloud Infrastructure.  
  
**Google Cloud Platform (GCP)** is a suite of cloud computing services provided by Google that runs on the same infrastructure that Google uses internally for its own products like Google Search and YouTube. GCP provides a wide range of services for building and deploying applications, websites, and data pipelines on the cloud.

We chose GCP for our anomaly detection pipeline because it offers a highly scalable and reliable infrastructure for data processing, machine learning, and dashboarding. GCP provides a wide range of cloud services that we can use to build a robust and scalable pipeline.

**Services Used in GCP**

**Google Cloud Storage Buckets:**

Extraction:

Google Cloud Storage Buckets is a highly scalable and durable object storage service that allows us to store and retrieve data on the cloud. We use Google Cloud Storage Buckets to extract data from source files and store it in the cloud for further processing.

Data Ingestion Stage:

The data ingestion stage involves extracting data from source systems and loading it into the cloud for further processing. In our pipeline, we used Google Cloud Storage Buckets for data extraction. Google Cloud Storage Buckets is a highly scalable and durable object storage service that allows organizations to store and retrieve data on the cloud. We used this service to store the input files from SAP Paragon systems, which were then processed in the subsequent stages.

**Data Proc:**

Transform by creating a Data Proc cluster and running ML code in cluster and saves output dataset to buckets.

Data Preparation Stage:

The data preparation stage involves transforming the data into a format suitable for analysis. In our pipeline, we used Google Cloud Dataproc for data transformation. Google Cloud Dataproc is a fully-managed cloud service that allows organizations to create Apache Hadoop and Apache Spark clusters on the cloud. We created a Dataproc cluster and ran machine learning code on the cluster to prepare the data for analysis. The cluster provides a highly scalable and distributed computing environment for running ML algorithms on large datasets. After transformation, the output dataset was saved to Google Cloud Storage Buckets.

Data Segregation Stage:

The data segregation stage involves separating the dataset into inliers and outliers. In our pipeline, we used machine learning algorithms to segregate the dataset into two parts: normal data points (inliers) and anomalous data points (outliers). This stage is critical in anomaly detection as it allows organizations to identify and flag potential anomalies in the data.

**BigQuery:**

Load to database using BigQuery

Google BigQuery is a highly scalable and fully managed cloud data warehouse service that allows us to store and analyze massive amounts of data. We use Google BigQuery to load the transformed data from Google Cloud Storage Buckets and store it in the data warehouse for further analysis. BigQuery provides a highly scalable and cost-effective solution for storing and analyzing large datasets on the cloud.

The model deployment and scoring stages involve deploying the machine learning model to a production environment and scoring new data points for anomalies. In our pipeline, we deployed the model to Google BigQuery, which is a highly scalable and fully managed cloud data warehouse service. We loaded the transformed data from Google Cloud Storage Buckets to Google BigQuery and used the model to score new data points for anomalies.

**Dashboard:**

The load to dashboard stage involves creating a dashboard to visualize the data and provide insights into anomalies. In our pipeline, we used Python's inbuilt Plotly library to create interactive data visualizations from the output dataset available in Google Cloud Storage Buckets. We used Dash API to host the dashboard on the cloud. Dash is a Python framework that allows organizations to build web-based dashboards using Python programming language.

We designed the dashboard to provide data specialists with insights into the anomalies detected by the machine learning model. The dashboard included various interactive charts and graphs, such as scatter plots, box plots, and histograms, that allowed data specialists to explore the data in real-time. We also included various filters and search options that enabled data specialists to drill down into specific subsets of the data.

Overall, the dashboard provided a user-friendly interface that enabled data specialists to quickly identify and investigate anomalies in the data. By combining machine learning with interactive data visualization, our anomaly detection pipeline provided a powerful tool for detecting and analyzing anomalies in large datasets.

**Input and Build Instructions:**

To run the anomaly detection pipeline on GCP, we used Cloud Composer, which is a managed service that runs the open-source Apache Airflow workflow orchestration platform. Airflow is a powerful tool that allows you to define, schedule, and monitor complex workflows on GCP. By using Cloud Composer, we were able to automate our pipeline and schedule it to run on a regular basis.

To get started with our pipeline, you'll need to follow this input and build instructions:

1. Create a GCP project and enable billing for the project.
2. Create a Google Cloud Storage bucket to store the input data files. This bucket will be used as the source for the ETL process. Here upload source data files in data lake and all the required scripts in scripts folder to run the pipeline.

Graphical user interface, application

Description automatically generated

1. Create a BigQuery dataset to store the output data. This dataset will be used to store the output of the machine learning model.

Text, table

Description automatically generated with medium confidence

1. Set up a Cloud Composer environment in GCP. This environment will be used to schedule and run the pipeline. In the Composer environment, you'll need to install the following packages: google-cloud-storage, google-cloud-bigquery, pandas, numpy, scikit-learn, joblib, matplotlib, and plotly.

**Environment Configuration**:

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

1. Upload the Python scripts and DAG file to the Cloud Composer environment. The Python scripts include the ETL process, machine learning model, and dashboard code. The DAG file defines the workflow for the pipeline.

Graphical user interface, application

Description automatically generated

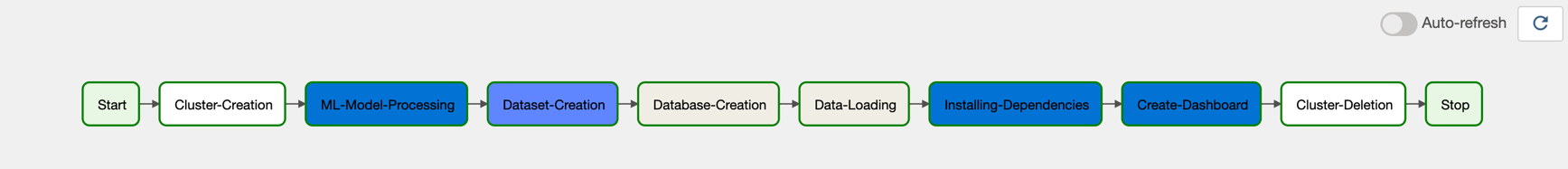
1. Update the configuration variables in the DAG file to specify the input and output paths for the ETL process and machine learning model, and the BigQuery dataset and table to store the output data.
2. Test the pipeline by running it manually in the Cloud Composer environment. You can monitor the progress of the pipeline using the Airflow web interface.

Once the pipeline is running successfully, schedule it to run on a regular basis using the Cloud Composer scheduler.

By following this input and build instructions, you can set up and run our anomaly detection pipeline on GCP using Cloud Composer and Airflow.

**Steps to run the pipeline:**

To ensure that the anomaly detection pipeline is functioning correctly, we need to validate it at each stage. There are several steps that we need to follow to validate the pipeline.

****

**Input Configuration:**

To run the pipeline for a specific material group, you need to provide the material group as input in the DAG configuration file. The DAG configuration file is a JSON file that contains the configuration settings for the DAG. This includes the input parameters for the DAG, such as the material group, input file location, and output file location. By specifying the material group, we can ensure that the pipeline processes only the data for that material group.

**Triggering the Pipeline:**

Once the DAG configuration file is configured, we can upload the DAG file to the Cloud Composer environment and use the Airflow web interface to start the DAG. The Airflow web interface provides a user-friendly interface for managing DAGs and monitoring the progress of the pipeline. We can monitor the status of the pipeline, view logs, and troubleshoot any issues that arise during the pipeline execution.

**Output Validation:**

After the pipeline has completed running, we should validate the outputs of outliers in the BigQuery database and dashboard. In the BigQuery database, we can query the output dataset to ensure that it contains the correct columns and that the data values are correct. We can use SQL queries to perform data validation checks and ensure that the output data is accurate.

**Dashboard:**

Table

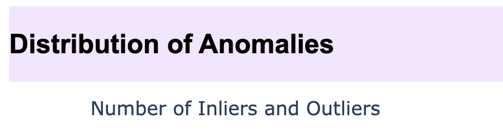
Description automatically generated

In the dashboard, we should ensure that the interactive charts and graphs are displaying the data correctly. We can use the filters and search options to drill down into the data and identify any outliers that may be present. By comparing the data in the dashboard with the data in the BigQuery database, we can ensure that the data is consistent and accurate across both platforms.

**Visualizations:**

Scatter chart

Description automatically generated



Chart, pie chart

Description automatically generated

By validating the pipeline at each stage, we can ensure that the pipeline is working correctly and that it is providing accurate and reliable results. This will help us to detect and analyze anomalies in our data more effectively and enable us to make more informed decisions based on the insights gained from the pipeline.

In summary, the validation process is critical to ensuring the accuracy and reliability of the anomaly detection pipeline. By following the steps outlined above, we can ensure that the pipeline is functioning correctly and that it is providing accurate and reliable results. This will enable us to make more informed decisions based on the insights gained from the pipeline and will help us to optimize our business processes and operations.

**Conclusion:**

The pipeline has been validated at each stage, and we have ensured that the pipeline is functioning correctly and that it is providing accurate and reliable results. By using this pipeline, we can detect and analyze anomalies in our data more effectively and make more informed decisions based on the insights gained from the pipeline.

**Enhancements:**

In the future, we can further enhance the pipeline by adding an input box in the dashboard where we can specify the material group. This will enable us to analyze data for specific material groups quickly and easily without having to modify the DAG configuration file manually.

The input box in the dashboard can be created using the Dash API. We can create a new input component that allows users to enter the material group. This input component can be linked to the existing dashboard charts and graphs, and the data displayed in the dashboard will be automatically updated based on the material group specified in the input box.

By adding this enhancement, we can make the pipeline more user-friendly and more accessible to data specialists who may not have technical expertise in modifying the DAG configuration files manually. It will also enable us to analyze data for specific material groups quickly and easily, making it easier to identify and analyze anomalies in our data.

In summary, by adding an input box to the dashboard to specify the material group, we can enhance the pipeline and make it more user-friendly and accessible. This will enable us to analyze data for specific material groups quickly and easily and make more informed decisions based on the insights gained from the pipeline.