OpenSky to BigQuery ML Pipeline

Prompts from Unit 3.3 Notebook

Real-time Flight Data Processing with Pub/Sub and BigQuery ML

Prompt 1

Write a Python code block for Google Colab that performs the following:

Installs the google-cloud-pubsub, google-cloud-bigquery, google-cloud-storage, and requests libraries via pip.

Imports auth from google.colab and authenticates the user to the Google Cloud environment.

Prompt 2

Create a configuration cell that defines Python variables for a Google Cloud project. Specifically, define:

PROJECT_ID (e.g., 'mgmt-467-47888')

REGION (e.g., 'us-central1')

TOPIC_NAME (e.g., 'opensky-realtime-topic')

SUBSCRIPTION_NAME (e.g., 'opensky-realtime-sub')

Prompt 3

Write a Python script using the Google Cloud SDK (google.cloud) to set up infrastructure idempotently. The script should:

Initialize Pub/Sub and BigQuery clients.

Create a Pub/Sub Topic if it doesn't exist.

Create a Pub/Sub Subscription to that topic if it doesn't exist.

Create a BigQuery Dataset if it doesn't exist.

Prompt 4

Write a Python class named OpenSkyApi. The constructor should accept a username and password. Include a method named get_states that accepts bounding box coordinates (lmin, lmax, hmin, hmax) or an icao24 code. The method should make a GET request to https://opensky-network.org/api/states/all and return the list of flight states from the JSON response.

Prompt 5

Create a function named process_and_publish that takes an OpenSky API response, a Pub/Sub publisher client, and a topic path.

The function should:

Iterate through the flight states.

Map the raw list data to a dictionary with keys matching a BigQuery schema (icao24, callsign, longitude, latitude, velocity, geo_altitude, etc.).

Add a current UTC timestamp field named sensor_timestamp.

Prompt 6

Write a function named pull_and_insert_and_predict that runs for a specified timeout period.

Inside the function:

Subscribe to the Pub/Sub subscription.

When a message is received, parse the JSON data.

Insert the row immediately into the BigQuery table realtime_flight_data.

Prompt 7

Write a script to execute the pipeline.

Initialize the OpenSkyApi client (using placeholder credentials if necessary).

Call the API to get flight states for a bounding box around Los Angeles (approx lat 34.0 to 34.5, lon -118.5 to -118.0).

Initialize the Pub/Sub publisher client.

Call the process_and_publish function to send the data to Pub/Sub.

Prompt 8

Write a code block to run the pull_and_insert_and_predict function with a timeout of 60 seconds. Capture the returned predictions into a Pandas DataFrame. If predictions exist, display the head of the DataFrame; otherwise, print that no messages were received.

Prompt 9

Using matplotlib and seaborn, write code to visualize the prediction results stored in the predictions_df DataFrame. Create a scatter plot where the x-axis is 'heading' and the y-axis is 'predicted_velocity'. Color the points based on the predicted velocity using a 'viridis' palette.

Prompt 10

Write a code block that:

Creates an input form field for CALLSIGN_TO_TRACK (defaulting to 'DAL339').

Queries the BigQuery table realtime_flight_data to retrieve the longitude, latitude, geo_altitude, and velocity for that callsign, ordered by timestamp.
Stores the result in a DataFrame.

Uses Seaborn to plot a line chart of Longitude vs. Latitude to visualize the flight path.

Prompt 11

Write a script to perform classification using BigQuery ML.

Define a model name flight_on_ground_classifier.

Construct a SQL query that uses ML.PREDICT with this model.

The input data for the prediction should be the most recent 100 records from the realtime_flight_data table (selecting icao24, geo_altitude, and velocity).

Execute the query and display the classification results in a DataFrame.