Real-Time Airline Price Prediction using GCP

# Introduction

This report details the implementation of a real-time airline price prediction model using Google Cloud Platform (GCP). The project leverages GCP components such as Cloud SQL, Cloud Storage, and BigQuery to extract, load, transform (ELT), and utilize real-time data for machine learning predictions.

# ELT Method

The ELT (Extract, Load, Transform) method involves three main steps:

1. Extract: Pull data from various sources (APIs, databases, etc.).

2. Load: Load the raw data into a centralized storage (Cloud Storage or Cloud SQL).

3. Transform: Transform the data within the storage (using BigQuery).

## Steps to Implement ELT

### Step 1: Extract Data

Use tools like Airflow, Dataflow, or custom scripts to extract real-time data from airline APIs or other sources.

Example code to extract data from an API and load it into Cloud SQL:

import requests  
import sqlalchemy  
from google.cloud.sql.connector import Connector  
  
API\_URL = "https://api.example.com/flights"  
DB\_USER = "your-username"  
DB\_PASS = "your-password"  
DB\_NAME = "airline\_data"  
INSTANCE\_CONNECTION\_NAME = "your-project:your-region:your-instance"  
  
connector = Connector()  
  
def getconn():  
 conn = connector.connect(  
 INSTANCE\_CONNECTION\_NAME,  
 "pymysql",  
 user=DB\_USER,  
 password=DB\_PASS,  
 db=DB\_NAME  
 )  
 return conn  
  
engine = sqlalchemy.create\_engine(  
 "mysql+pymysql://",  
 creator=getconn,  
)  
  
response = requests.get(API\_URL)  
flight\_data = response.json()  
  
with engine.connect() as connection:  
 for flight in flight\_data:  
 connection.execute(  
 "INSERT INTO flights (departure\_time, arrival\_time, price) VALUES (%s, %s, %s)",  
 (flight["departure\_time"], flight["arrival\_time"], flight["price"])  
 )

### Step 2: Load Data into Cloud SQL

Create a MySQL instance in Cloud SQL and set up a database and table for storing airline data.

CREATE DATABASE airline\_data;  
  
CREATE TABLE airline\_data.flights (  
 flight\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 departure\_time DATETIME,  
 arrival\_time DATETIME,  
 price FLOAT,  
 timestamp DATETIME DEFAULT CURRENT\_TIMESTAMP  
);

### Step 3: Storing Raw Data in Cloud Storage

from google.cloud import storage  
import json  
  
client = storage.Client()  
bucket\_name = "your-bucket-name"  
bucket = client.bucket(bucket\_name)  
  
raw\_data = json.dumps(flight\_data)  
blob = bucket.blob("raw\_flights.json")  
blob.upload\_from\_string(raw\_data, content\_type="application/json")

### Step 4: Transforming Data in BigQuery

Create a BigQuery dataset and load the raw data from Cloud Storage.

CREATE DATASET IF NOT EXISTS airline\_data;  
  
LOAD DATA INTO TABLE airline\_data.raw\_flights  
FROM FILES (  
 format = 'json',  
 uris = ['gs://your-bucket-name/raw\_flights.json']  
);

Use SQL queries to transform the data in BigQuery.

CREATE OR REPLACE TABLE airline\_data.transformed\_flights AS  
SELECT  
 flight\_id,  
 departure\_time,  
 arrival\_time,  
 price,  
 LEAD(price) OVER (PARTITION BY flight\_id ORDER BY timestamp) AS next\_price  
FROM  
 airline\_data.raw\_flights;

## Step 5: Training and Deploying the ML Model

### Extracting Data from BigQuery for Training

from google.cloud import bigquery  
  
client = bigquery.Client()  
  
query = """  
SELECT  
 flight\_id,  
 departure\_time,  
 arrival\_time,  
 price,  
 next\_price  
FROM  
 `your-project.airline\_data.transformed\_flights`  
"""  
  
df = client.query(query).to\_dataframe()  
df.to\_csv("transformed\_flights.csv", index=False)

### Training the ML Model

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestRegressor  
  
data = pd.read\_csv("transformed\_flights.csv")  
  
X = data.drop(columns=["next\_price"])  
y = data["next\_price"]  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
model = RandomForestRegressor(n\_estimators=100, random\_state=42)  
model.fit(X\_train, y\_train)  
  
import joblib  
joblib.dump(model, "flight\_price\_model.joblib")

### Deploying the Model

from google.cloud import aiplatform  
  
client = aiplatform.gapic.PipelineServiceClient()  
  
model = aiplatform.Model.upload(  
 display\_name="flight\_price\_model",  
 artifact\_uri="gs://your-bucket-name/flight\_price\_model.joblib",  
)  
  
endpoint = aiplatform.Endpoint.create(  
 display\_name="flight\_price\_prediction\_endpoint",  
)  
model.deploy(endpoint=endpoint)

# Automating the Process

To automate the ETL process, use Cloud Functions to trigger data extraction and loading, Cloud Composer (Airflow) for orchestrating ETL workflows, and Pub/Sub for real-time data ingestion.