Cloud-based Web Application for Real-Time Flight Information with Anomaly Detection

JUNGWON ZANG, LAUREN MEYER, MICHAEL FORNITO, KRYSH RAJENDRAN, THOMAS FIELLO
Embry Riddle Aeronautical University, Daytona Beach, FL 32114 USA
Corresponding authors: Jungwon Zang (e-mail: zangj@my.erau.edu)
Lauren Meyer (e-mail: meyerl7@my.erau.edu)
Michael Fornito (e-mail: fornitom@my.erau.edu)
Krysh Rajendran (e-mail: rajendrk@my.erau.edu)
Thomas Fiello (fiellot@my.erau.edu)

Abstract— According to most estimates, there are roughly 150,000 worldwide flights every day. Additionally, there are an estimated 5,000 flights happening at any given time in the United States. Generally, for one flight, there are many related stakeholders within that process. These items are provided through real-time data such as flight, aeronautics, weather, and airspace. Therefore, it has become increasingly important for each of these elements to be organized and combined, and then ultimately shared among industry partners and even non- industry individuals. This project aims to further heighten the importance of delivering accurate information to the right people at the right time using cloud computing in the aviation industry. The aircraft industry is mostly affected by unexpected factors such as weather, airport status, mechanical delays, impacts from the COVID-19 pandemic, and so on. Those unpredictable elements can cause flight delays and even flight cancellations. This project aims to query and record real-time flight status of all airline flights at the Daytona Beach International Airport into a database and then share this information through a web application using cloud computing.

Keywords—cloud computing, API, Flask Python, Google Cloud Platform, JSON, real-time data

I. Introduction

US Federal Aviation Administration (FAA) reports that the aviation industry is implementing NextGen Air Traffic Management (ATM) to increase predictability and efficiency of the National Airspace System (NAS). In the NextGen program, there are several components, including Trajectory Based Operation (TBO), Data Communications (Data Comm), Performance Based Navigation (PBN), Automatic Dependent Surveillance-Broadcast (ADS-B) and System Wide Information Management (SWIM). SWIM is the promising digital data-sharing backbone of NextGen, according to the FAA. A single access point for aviation data is offered by this system to stakeholders for data publishing and users for data access whenever and wherever they need it [1].

The SWIM system implements cloud computing service to share aviation data such as flight data; Flight Information eXchange Model (FIXM), aeronautical data; Aeronautical Information eXchange Model (AIXM) and weather data; Weather Information eXchange Model (iWXXM) [2] to air traffic stakeholders. The transmission for those data sets uses ADS-B open data system to communicate and share aircraft's real-time tracking information. Cloud computing that uses the internet for storing and managing data on remote databases and then accessing data via the internet augments the efficiency of dealing with and analyzing big data of aircraft real-time quickly. In addition, analyzed data can be shared to the users in real-time [2].

The project we are proposing combines database management and cloud computing technologies. With the use of cloud computing, this project will extract real-time flight information, including flight number, airline name, flight departure/arrival time, departure/arrival airport, and flight status, and will share the information to the web. An Application Programming Interface (API) is provided by OpenSky for real-time flight information and global aviation data. OpenSky data is sent as JSON data to the Python architecture system for cleaning and storing it on a cloud-based server for retrieval and display on a web server. An open-sharing system such as ADS-B is vulnerable to cyber-security attacks. Therefore, our project proposed ADS-B anomaly detection model to analyze errors and prediction errors. This work shows that combination the error-free real-time flight data and cloud computing can help travelers be more aware of changes in their travel arrangements such as flight delays or cancellations.

II. Literature Review

For real-time flight data collection, many methods have been proposed and implemented. The Automatic Dependent Surveillance-Broadcast (ADS-B) is installed on every single aircraft for determining its position and velocity using GPS and periodically broadcasting the real time location of aircraft to ADS-B receivers, which is installed around the globe. OpenSky operates and controls those ADS-B receivers and harvests data via the Internet with cloud computing to share data to web application [3]. Also, because of extensive application of data-driven technologies, ADS-B is vulnerable to attackers for cyber security.

Satria Bagus Panuntun and Setia Pramana proposed a website scraping method from a site that provides monitoring and tracking services for all flights in the world. This method uses Python programming language and API provided by the website. They extract information about flight code, aircraft code, airline name, departure airport, departure city, arrival airport, arrival city, date/time of departure and arrival and flight status. In this paper, they have presented a streaming real-time flight data and sharing into web application with the proposed method [4]. Peng Luo et al and Jing Wang et al insisted that ADS-B has become an important part of the NextGen air transport system because ADS-B technology has the advantages of high accuracy and data sharing. However, ADS-B has an open sharing system, it is vulnerable to spoofing attack [5]. They proposed an ADS-B spoofing attack detection method based on LSTM or VAE-SVDD. Both papers used machine learning methods to detect different types of anomalous data by setting different thresholds for different features.

III. Problem Statement and Solution

Anomaly ADS-B data detection is the main problem nowadays. Due to many unexpected situations and conditions including meteorological complexity of the model and poor data integrity, there are many errors between the actual flight tracking and ADS-B data received. Those errors lead to information leakage, lack of authentication mechanism and cyber security problems. Based on the current system, there is no such promising system which provides anomaly ADS-B data detection that is important to attain accurate data tracking from ADS-B equipment in general aviation.

To solve the problem that we stated above, generally, it is difficult to change the existing ADS-B protocol structure, hardware installations and requirements. After researching most of the literature review, our project decided to use an Isolation Forest model for anomaly detection. We aim to detect anomaly and abnormal flights by using machine learning and deep learning techniques.

IV. Project Methodology

This section briefly explains all project methodologies that our project uses. Sub-sections consist of API, REST API, ADS-B, Google BigQuery, some web visualization tools and machine learning model for Anomaly Detection.

A. *API*

API stands for an Application Programming Interface. It provides a service for companies to open their applications' data and functionality to external and internal developers, and business partners. With API, developers do not need to know how the API is implemented, but they just simply use the interface to communicate with other services. API enables developers to communicate with each other and leverage each other's data.

OpenSky API collects real-time flight information through ADS-B. The API includes flights and tracking information of each aircraft in a two-dimensional array. The collected ADS-B properties are:

- ICAO24
- callsign
- origin_country
- time_position
- last_contact
- longitude
- latitude
- baro_altitude
- on_ground
- velocity
- true_track
- vertical_rate
- sensors
- geo_altitude
- squawk
- spi
- position_source

B. *REST API*

OpenSky has provided their API for our proposed project to be able to use Representational State Transfer (REST) APIs that have more useful return types. Using a REST API, a client can request a resource, and the server will send back the current state of the resource. Requests for a resource are received and all relevant information about the requested resource is returned, in a form that clients can comprehend easily. [6]

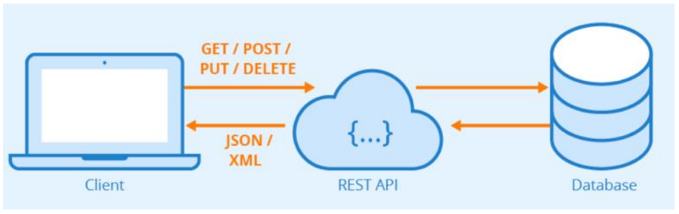


Fig 1. REST API architecture

This project uses REST API to fetch real-time data from OpenSky with Python programming language to collect and clean the data.

C. *ADS-B*

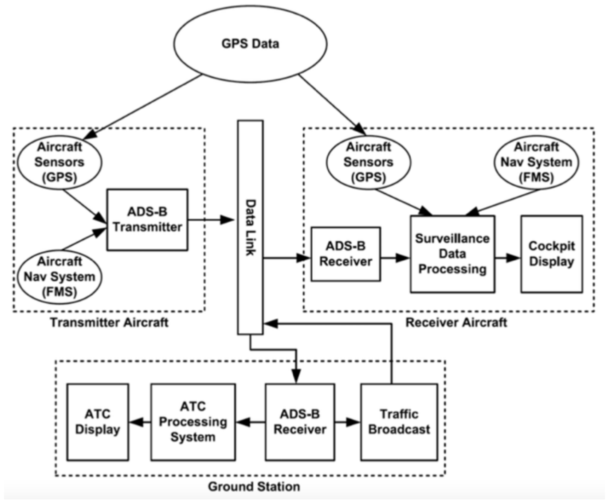


Fig 2. ADS-B system architecture

OpenSky provides open-source data for flight tracking. They use ADB-S for getting real-time flight data. The ADS-B is a surveillance system that determines flight's position and velocity using GPS (Global Positioning System). Airplanes periodically broadcast their real-time information and other important information on the 1090 MHz radio frequency. Open sky operates a network of ADB-S receivers around the world and collects the data via the internet. Then, OpenSky multiple servers collect and clean the incoming raw flight data and create state vector, which represents the position and velocity of an aircraft in any location around the world to upload and fetch data from each aircraft every second. After receiving flight data from receivers, OpenSky provides API of those data with JSON file format.

D. *Google BigQuery*

Google BigQuery is a data processing architecture and scalable data warehouse with a SQL query engine. This is a Relational Database Management System (RDMS) that supports Online Transaction Processing (OLTP) database and Structured Query Language (SQL). BigQuery provides the function to write a query for extracting partial data that our project needs. BigQuery is a serverless and can run queries without the need to manage infrastructure. This enables our project to carry out mountain of data in seconds to minutes. BigQuery becomes more useful when the project needs to join datasets from various sources or when it queries against data that is stored outside BigQuery.

ETL is the traditional way with data warehouse starting with Extract, Transform, and Load (ETL) process, which is the process the raw data is extracted from its resource location, transformed and then loaded into the data warehouse. This is used when the raw data needs to be controlled and transformed before being loaded into BigQuery and when the data loading needs to happen contimuously.

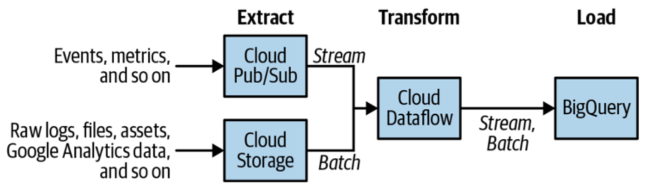


Fig 3. ETL Architecture

E. *Web Visualization tools*

Flask Python is a web framework that represents a collection of libraries and modules enabling web application developers to easily write applications and this is a Python module, which is suitable for developing web applications easily. This is designed with microframework to keep the application simple and scalable and supports extensions to the application.

To save time and resources, we decided to use a free template from AppSeed.us. Appseed's templates are very easy and nice to work with largely because of their incorporation of Bootstrap, a front-end open-source development toolkit. Bootstrap saves us from having to develop a cohesive CSS environment completely from scratch, while the Appseed template provides additional features, such as: local authentication, default images, and an overall cohesive theme.

Finally, we wished to use the collected information to build an interactive map which displays the current position of any aircraft over Daytona Beach. To that end, we went with the Python library *Folium* which allowed us to not only display a detail rich interactive map, but also allowed us to add custom airplane markers at any given latitude and longitude.

F. *Machine Learning model for Anomaly Detection*

Anomaly detection is a form of unsupervised machine learning that specializes in identifying anomalous data. In general, it works by examining all given data to establish an acceptable tolerance for what any future data may look like. Any data which falls outside of the realm of acceptable possibility is thus labeled as an 'anomaly'. Possible anomalies which may be uncovered by our model may include such things as: engine failures, spoofed flight data, inexperienced pilots, etc.

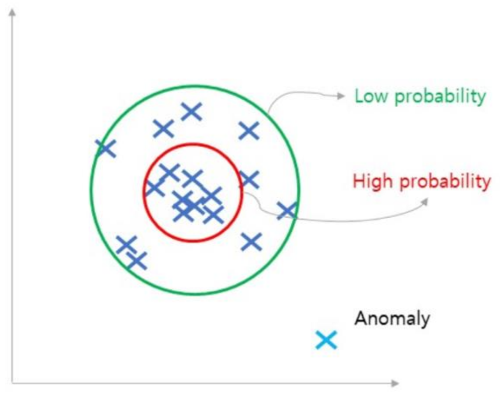


Fig 4. Anomaly detection example

For our purposes, we decided to use the *Isolation Forest* algorithm from the SKLearn library. *Isolation Forest*, like the popular classifier *Random Forest*, is an ensemble model made up of multiple high variance/binary decision trees. It works by isolating observations through random selection of features and testing different split values between their max and min values. The fewer the number of splits it takes to isolate a value, the more likely it is an anomaly, as shown in Fig 5. [7] [8]

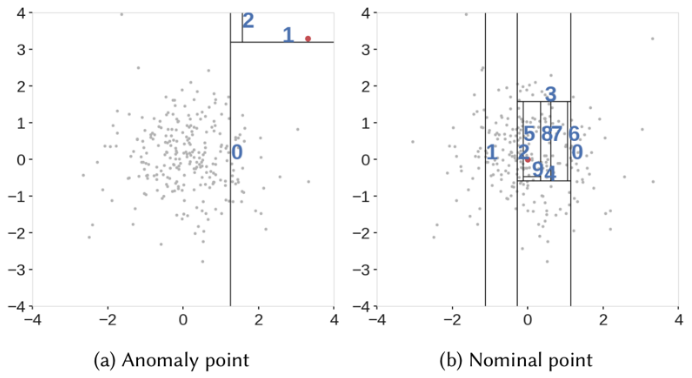
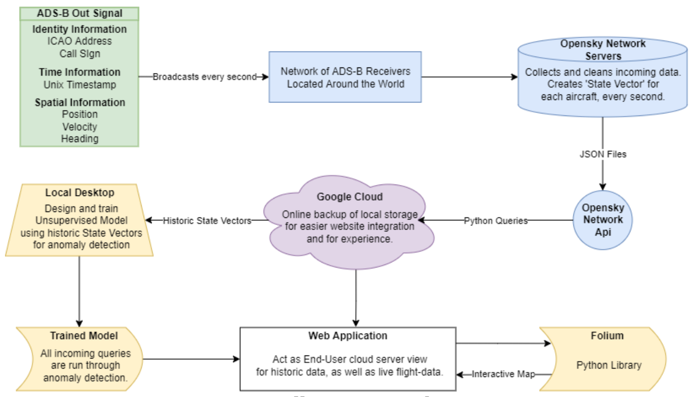


Fig 5. (a) shows how quickly the anomalous point was separable from the rest of the data,
while (b) shows how nominal data takes many more such splits.

V. Project Overview

A. *System Architecture*



ADS-B Out Signal
Identity Information
ICAO Address
Call SIgn
Time Information
Unix Timestamp
Spatial Information
Position
Velocity
Heading
Broadcasts every second
Network of ADS-B Receivers Located Around the World
Opensky Network Servers
Collects and cleans incoming data. Creates 'State Vector' for each aircraft, every second.
JSON Files
Opensky Network Api
Python Queries
Google Cloud
Online backup of local storage for easier website integration and for experience.
Historic State Vectors
Local Desktop
Design and train Unsupervised Model using historic State Vectors for anomaly detection
Trained Model
All incoming queries are run through anomaly detection.
Web Application
Act as End-User cloud server view for historic data, as well as live flight-data.
Interactive Map
Folium
Python Library

Fig 6. Overall System Architecture

Fig 6 is our project's system architecture. ADS-B Out signal sends identity such as ICAO address, call sign, time like Unix timestamp, and spatial information such as velocity, position, and heading. It broadcasts real-time information every second to the networks of ADS-B receivers located around the world. The OpenSky network server collects and cleans incoming real-time data and creates 'state vectors' for each aircraft for every second.

This project uses OpenSky REST API data to fetch real-time data as JSON files to populate data to Google BigQuery with Python programming language for storing and exporting cleaned real-time data through Google Cloud system to web application. Also, this project deals with anomaly detection of flight data using historical state vectors of flight data by designing and training Isolation model. Through the trained model, all incoming queries are running anomaly detection and flagged anomalies to web application.
.

B. *System Phase*

This project has four representative system phases: importing from the API, cleansing and storage, exporting to website, and visualization.

1. Data Fetching and Cleaning

This project decided to use Python programming language to fetch and clean data from OpenSky API. Python has various libraries to support connection to BigQuery, Pandas, FlightRadat24 API, JSON, math, datetime, geopandas, geoplot and folium. After fetching real-time data from OpenSky, we cleaned the imported data with defining Python functions.

- Fetching airport data from FlightRadar24

We used FlightRadar24 API for finding airport location by returning airport's latitude and longitude. We also set the distance around the airport for OpenSky data as 20km range.

- Querying OpenSky data

When we get the location information from API, we query OpenSky network with the given latitude and longitude information and returns data as Pandas data frame. In our data frame, the columns of imported data were 'icao 24', 'callsign', 'origin_country', 'time_position', 'last_contact', 'longitude', 'latitude', 'baro_altitude', 'on_ground', 'velocity', 'true_track', 'vertical_rate', 'sensors', 'geo_altitude', 'squawk', 'spi', 'x', 'y'.

- Cleaning data frame

For cleaning, we dropped the 'x', 'y', 'sensors', and 'squawk' columns that we thought useless for our project and checked whether the data has null values or not that is critical in data cleaning. Three columns that contain time elements such as 'time_position', 'last_contact' and 'obs_time' are applied to_datetime() function with seconds unit.

- Pushing cleaned data frame to the cloud

After fetching and cleaning process, we connected to Google Cloud BigQuery and pushed cleand data frame to the cloud. Fig 7 is our table schema after pushing cleaned data to Google BigQuery. Fig 7 presents the data frame with name and data type that we cleaned with Pandas and Python after pushing to the Google BigQuery.

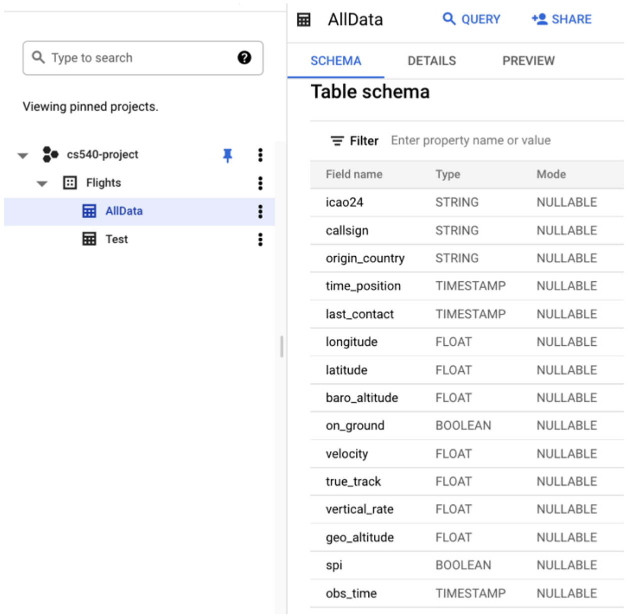


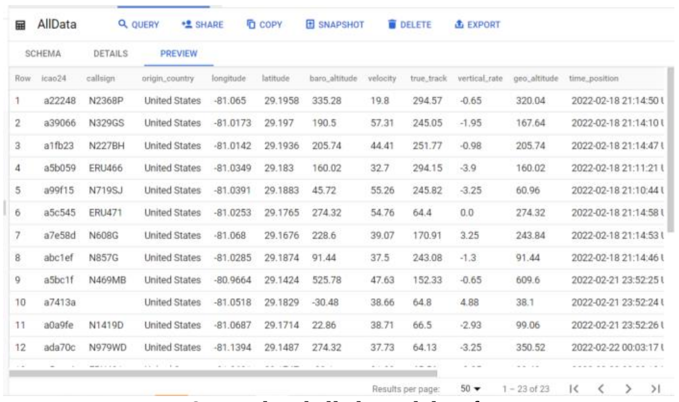
Table schema

Field name	Type	Mode
icao24	STRING	NULLABLE
callsign	STRING	NULLABLE
origin_country	STRING	NULLABLE
time_position	TIMESTAMP	NULLABLE
last_contact	TIMESTAMP	NULLABLE
longitude	FLOAT	NULLABLE
latitude	FLOAT	NULLABLE
baro_altitude	FLOAT	NULLABLE
on_ground	BOOLEAN	NULLABLE
velocity	FLOAT	NULLABLE
true_track	FLOAT	NULLABLE
vertical_rate	FLOAT	NULLABLE
geo_altitude	FLOAT	NULLABLE
spi	BOOLEAN	NULLABLE
obs_time	TIMESTAMP	NULLABLE

Fig 7. Pushed cleaned data to BigQuery

2. Data Populating and Storing

Fig 8 is the real-time data imported from OpenSky and cleaned by Python code that we wrote. We used Flask Python for writing a BigQuery connection code. This project needs tabulate, signal and flask library and maps. The URL is associated with the root URL. To check the status that we relate to BigQuery, we tested the data exporting to the web limited only Embry Riddle Aircrafts whose callsign starts from 'ERU'. With SQL, we selected the columns, database to extract data and restricted data in WHERE statement.

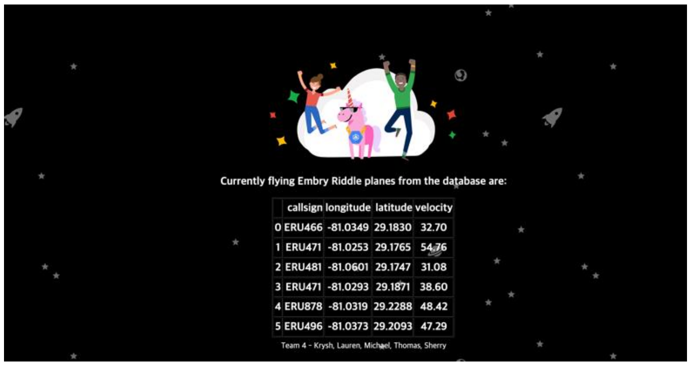


Row	icao24	callsign	origin_country	longitude	latitude	baro_altitude	velocity	true_track	vertical_rate	geo_altitude	time_position
1	a22248	N2368P	United States	-81.065	29.1958	335.28	19.8	294.57	-0.65	320.04	2022-02-18 21:14:50 U
2	a39066	N329GS	United States	-81.0173	29.197	190.5	57.31	245.05	-1.95	167.64	2022-02-18 21:14:10 U
3	a1fb23	N227BH	United States	-81.0142	29.1936	205.74	44.41	251.77	-0.98	205.74	2022-02-18 21:14:47 U
4	a5b059	ERU466	United States	-81.0349	29.183	160.02	32.7	294.15	-3.9	160.02	2022-02-18 21:11:21 U
5	a99f15	N719SJ	United States	-81.0391	29.1883	45.72	55.26	245.82	-3.25	60.96	2022-02-18 21:10:44 U
6	a5c545	ERU471	United States	-81.0253	29.1765	274.32	54.76	64.4	0.0	274.32	2022-02-18 21:14:58 U
7	a7e58d	N608G	United States	-81.068	29.1676	228.6	39.07	170.91	3.25	243.84	2022-02-18 21:14:53 U
8	abc1ef	N857G	United States	-81.0285	29.1874	91.44	37.5	243.08	-1.3	91.44	2022-02-18 21:14:46 U
9	a5bc1f	N469MB	United States	-80.9664	29.1424	525.78	47.63	152.33	-0.65	609.6	2022-02-21 23:52:25 U
10	a7413a		United States	-81.0518	29.1829	-30.48	38.66	64.8	4.88	38.1	2022-02-21 23:52:24 U
11	a0a9fe	N1419D	United States	-81.0687	29.1714	22.86	38.71	66.5	-2.93	99.06	2022-02-21 23:52:26 U
12	ada70c	N979WD	United States	-81.1394	29.1487	274.32	37.73	64.13	-3.25	350.52	2022-02-22 00:03:17 U

Results per page: 50 1 – 23 of 23

Fig 8. Populated all cleaned data frame

Fig 9 is the page that indicates that we succeeded fetching data that we were intended to test. The table in Fig 8 presents all Embry Riddle University aircraft that are hovering over the Daytona Beach Airport.


Currently flying Embry Riddle planes from the database are:

	callsign	longitude	latitude	velocity
0	ERU466	-81.0349	29.1830	32.70
1	ERU471	-81.0253	29.1765	54.76
2	ERU481	-81.0601	29.1747	31.08
3	ERU471	-81.0293	29.1871	38.60
4	ERU878	-81.0319	29.2288	48.42
5	ERU496	-81.0373	29.2093	47.29

Team 4 - Krysh, Lauren, Michael, Thomas, Sherry

Fig 9. BigQuery Connection

We were planning to use MySQL for this project but decided to use Google BigQuery instead of MySQL. For this project, Google BigQuery was over MySQL because BigQuery is a cloud-based architecture and provides auto-scale up and down based on the data loading and it is efficient for data analysis.

3. Web Application with exporting real-time data

We used Flask with an Appseed template for our website to improve the web application process by connecting with Google BigQuery. The first page of our website is the login page with requiring account and password, also creating the new account as Fig 10.

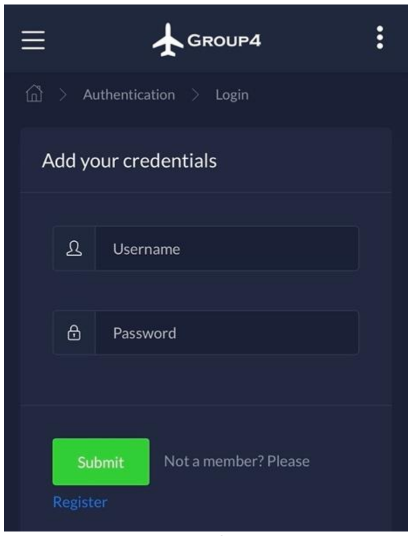
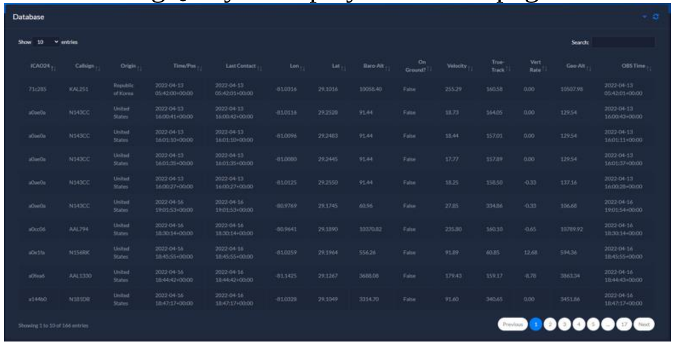


Fig 10. Login page of website

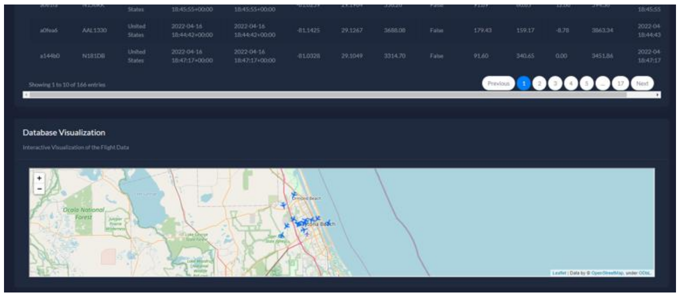
After you logged in, users can check the real-time flight data as a table like Fig 11. Users can search their flight by checking flight number as callsign in the table and can check origin and real-time tracking information. The dashboard grabs the latest data from OpenSky, cleans and uploads that data to BigQuery and selects all of the data from BigQuery to display on the webpage.



ICAO24	Callsign	Origin	Time/Pos	Last Contact	Lon	Lat	Baro-Alt	On Ground?	Velocity	True-Track	Vert Rate	Geo-Alt	OBS Time
71c285	KAL251	Republic of Korea	2022-04-13 05:42:00+00:00	2022-04-13 05:42:01+00:00	-81.0316	29.1016	10058.40	False	255.29	160.58	0.00	10507.98	2022-04-13 05:42:01+00:00
a0ae0a	N143CC	United States	2022-04-13 16:00:41+00:00	2022-04-13 16:00:42+00:00	-81.0116	29.2520	91.44	False	18.73	164.05	0.00	129.54	2022-04-13 16:00:43+00:00
a0ae0a	N143CC	United States	2022-04-13 16:01:10+00:00	2022-04-13 16:01:10+00:00	-81.0096	29.2483	91.44	False	18.44	157.01	0.00	129.54	2022-04-13 16:01:11+00:00
a0ae0a	N143CC	United States	2022-04-13 16:01:35+00:00	2022-04-13 16:01:35+00:00	-81.0080	29.2445	91.44	False	17.77	157.89	0.00	129.54	2022-04-13 16:01:37+00:00
a0ae0a	N143CC	United States	2022-04-13 16:00:27+00:00	2022-04-13 16:00:27+00:00	-81.0125	29.2550	91.44	False	18.25	158.50	-0.33	137.16	2022-04-13 16:00:28+00:00
a0ae0a	N143CC	United States	2022-04-16 19:01:53+00:00	2022-04-16 19:01:53+00:00	-80.9769	29.1745	60.96	False	27.85	334.86	-0.33	106.68	2022-04-16 19:01:54+00:00
a0cc06	AAL794	United States	2022-04-16 18:30:14+00:00	2022-04-16 18:30:14+00:00	-80.9641	29.1890	10370.82	False	235.80	160.10	-0.65	10789.92	2022-04-16 18:30:14+00:00
a0e1fa	N1568K	United States	2022-04-16 18:45:55+00:00	2022-04-16 18:45:55+00:00	-81.0259	29.1964	556.26	False	91.89	60.85	12.68	594.36	2022-04-16 18:45:55+00:00
a0fea6	AAL1330	United States	2022-04-16 18:44:42+00:00	2022-04-16 18:44:42+00:00	-81.1425	29.1267	3688.08	False	179.43	159.17	-8.78	3863.34	2022-04-16 18:44:43+00:00
a144b0	N181DB	United States	2022-04-16 18:47:17+00:00	2022-04-16 18:47:17+00:00	-81.0328	29.1049	3314.70	False	91.60	340.65	0.00	3451.86	2022-04-16 18:47:17+00:00

Showing 1 to 10 of 166 entries

Fig 11. Real-time flight data frame in web



a0fea6	AAL1330	United States	2022-04-16 18:44:42+00:00	2022-04-16 18:44:42+00:00	-81.1425	29.1267	3688.08	False	179.43	159.17	-8.78	3863.34	2022-04 18:44:43
a144b0	N181DB	United States	2022-04-16 18:47:17+00:00	2022-04-16 18:47:17+00:00	-81.0328	29.1049	3314.70	False	91.60	340.65	0.00	3451.86	2022-04 18:47:17

Showing 1 to 10 of 166 entries

Previous
Next
Database Visualization
Interactive Visualization of the Flight Data
Ocala National Forest
Leaflet | Data by © OpenStreetMap, under ODbL

Fig 12. Database visualization

The website presents all current flights on the map. We used Folium interactive map centered at Daytona Beach Airport with specified 20km range around the airport. It grabs the database from BigQuery and uses the current location of each plane and then presents data in interactive map. The real-time flight Daytona Beach Airport map looks like Fig 13.

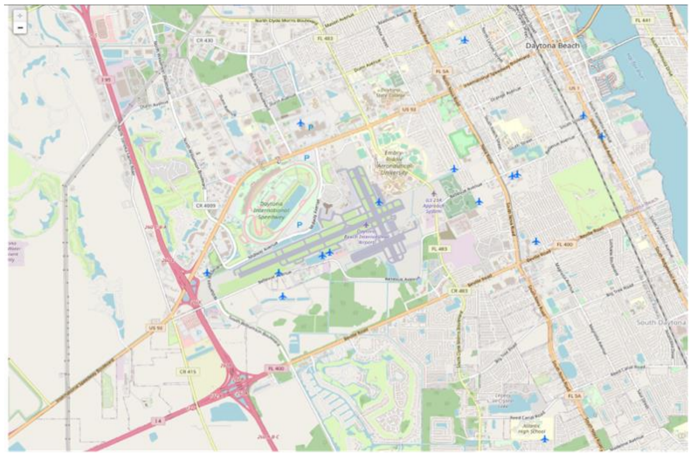


Fig 13. Real-time map on Daytona Beach Airport

4. Anomaly Detection

For anomaly detection, we fetched historical data from OpenSky transformed as a CSV file format. First, we fetched and imported data and checked the data type of each column. For data cleaning, we reorganized headers, converted time to datetime format and, we made attributes int 64.

Checking null values in a dataset is important. We implemented null checking using np.random.RandomState() and isnull() functions and then used dropna() and isna() functions to remove detected null values and missing values.

Before starting to build an isolation model, we checked outliers with a boxplot in Fig 14. Outlier describes how numerically distant from the rest of the data. Box plot diagram depicts quartiles and inter quartiles to define the upper limit and lower limit that we considered as outliers below this lower limit. We set the attributes including 'velocity', 'vertrate', 'latitude' and 'longitude' for checking outliers as these are key attributes to look at when considering an anomaly in an aircraft's flight pattern.

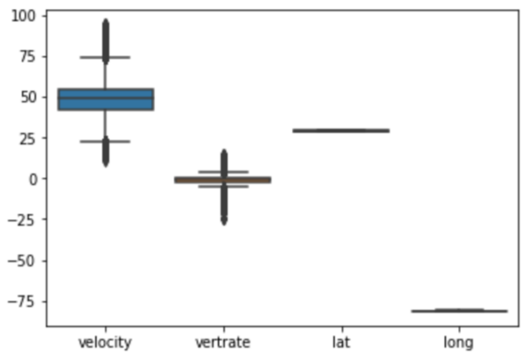


Fig 14. Boxplot of outliers

For our isolation forest model, we set the contamination number as 0.003, as we estimated that 1 to 3 flights of 1000 flights experience go-around flights; a circle of the airport to reapproach and land. The contamination parameter in the model represents the percentage of the data that would be considered as an anomaly. An anomaly in our data would be any abnormal flight data that appears in the vicinity of DAB, including go arounds or abnormal flight patterns. For isolation forest, a good way to select contamination would be to test different variables or visualize the original data, like the boxplot in Fig 14, to better understand where the anomalies would be.

After implementing and running the isolation forest model, it is easier to view anomalies among the normal data within a 3D visualization. In the figure, the green circles indicate all normal data points that received a score of 1. The red x indicates anomaly points that received an anomaly score of -1. Therefore, the anomaly score and its results were simply visualized for further understanding based on what the model generated.

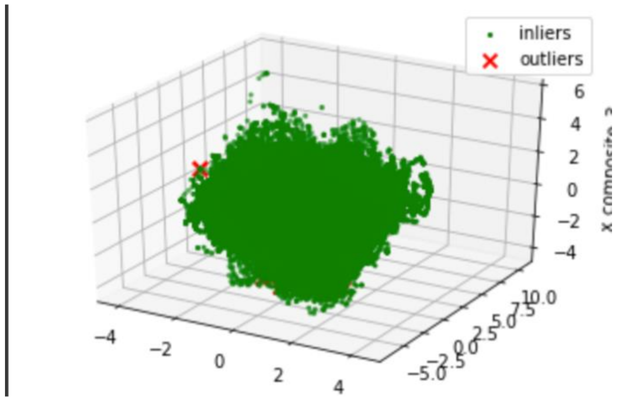


Fig 15. Visualization of outliers

Fig 16 represents a normalized view of the data set in a 2D format. Isolation forest looks at each data point and considers its anomalous behavior. Therefore, the 2D graph shows that outliers are typically in the midrange for this dataset. When you consider we are evaluating flight data, most planes are either flying or on ground. The anomalies occur mostly during the landing, take off, or even go around.

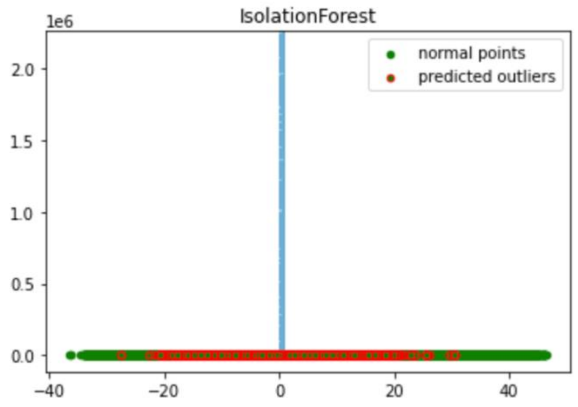


Fig 16. Isolation forest model of outliers

VI. Conclusion

Our whole database is less than 40 instances and the only way we must populate it is by refreshing the web page or running the code ourselves. Both are inconvenient and would take forever to populate the database with the amount of data. This makes this website take about 10 seconds to load a page.

The first future work would be getting the Google Cloud Functions with Google Scheduler to work basically to set up some code to grab the latest data automatically like every ten minutes. Next, the second future work would be for better understanding the anomalies, we would suggest that drills down deeper into the anomalous points to understand the importance of these data points during a flight plan. In addition, as we are currently using historical data to test this model and its predictions, there is potential for this project to run these models with the real-time flight data from the OpenSky API in Big Query. We feel that viewing the anomalies on the same map as the live flights would be effective for the viewer and be a unique way to show the anomalous flight data.

References

[1] FAA NextGen website: https://www.faa.gov/nextgen/this_is_nextgen/

[2] SWIM Distribution Service PPT slide, Melissa Matthews, 09.2019, Available: https://www.faa.gov/air_traffic/flight_info/aeronav/atiec/media/Presentations/Day%201%20PM%20009%20Melissa%20Matthews%20SCDS.pdf

[3] OpenSky website: https://opensky-network.org/

[4] Satria Bagus Panuntum and Setia Pramana, "Development of Automated Flight Data Collection System for Air Transportation Statistics", IOP publishing, DOI: 10.1088/1742-6596/1863/1/012020, Available: https://iopscience.iop.org/article/10.1088/1742-6596/1863/1/012020/pdf

[5] Jing Wang, Yunkai Zou and Jianli Ding, "ADBS-B spoofing attack detection method based on LSTM", EURASIP Journal on wireless communications and networking 2020, DOI: 10.1186/s13638-020-01756-8, Available: https://www.researchgate.net/publication/343614169_ADS-B_spoofing_attack_detection_method_based_on_LSTM

[6] REST API: https://en.wikipedia.org/wiki/Representational_state_transfer

[7] Anomaly detection with isolation forest guide, Available: https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/

[8] Adithya Krishnan, "Anomaly Detection with Isolation Forest & Visualization", Available: https://towardsdatascience.com/anomaly-detection-with-isolation-forest-visualization-23cd75c281e2

[9]