The Impact of Covid-19 on Air Traffic: Spatiotemporal Analysis and Benchmarking

Adelle Driker, Bo Yan, Yuan Hu

Advisor: Professor Rose Yu

Raw Data Sources

Dataset Summary

OpenSky Flight Data[1]

Our foundational dataset, which is derived and cleaned from OpenSky data source, contains spatiotemporal flight information data since the beginning January 01, 2019 and continues through the COVID-19 pandemic. The OpenSky dataset is composed of monthly CSV files with fields such as CallSign (ID), Origin, Destination, Day/Time of first/last message received by OpenSky Network, Latitude/Longitude/Altitude for both Origins and Destinations, among others. This dataset is to be updated for the remainder of the pandemic.

Johns Hopkins Global COVID19 Data

As one of the largest providers of public COVID19 data since the beginning of the pandemic, it made sense for us to link the OpenSky Flight data to that from Johns Hopkins University. This dataset contains spatiotemporal information about global confirmed cases, recoveries and deaths in separate files. All files hold the same attributes: Province/State, Country/Region, Lat, Long, and dates ranging from 01/22/2022 - present. Johns Hopkins provides daily updates to their dataset and claims to update the historical data if any inaccuracies are reported.

Bansard Airline Code and Country Mapping

In order to map an airline to the country it belongs to, it is necessary to have a dataset which contains these links. The Bansard Airline Code and Country Mapping file, which is sourced from the International Air Travel Association (IATA) and International Civil Aviation Organization (ICAO) and will act as a mapping table between the OpenSky and Johns Hopkins datasets so that further investigation at the country level may be conducted. Fields included in this dataset are: Airline Name, IATA Designator, 3-Digit Code, ICAO Designator, and Country.

Table of Dataset Logistics

Dataset Name	Source Location	Destination in Pipeline	Data Size
OpenSky_ Flight_Data	OpenSky Network	InfluxDB bucket	5.7GB and growing (zipped)
Notebook/Script: nov_21_top_airline.ipynb			
JH_Global_ Confirmed	CSV from JH GitHub	InfluxDB bucket	1.03MB and growing
JH_Global_ Recovered	CSV from JH GitHub	InfluxDB bucket	0.8MB and growing
JH_Global_ Deaths	CSV from JH GitHub	InfluxDB bucket	0.7MB and growing
Notebook/Script: confirmed_covid_us.ipynb, recovered_us.ipynb, death_us.ipynb			
Airline_Code_to_ Country	Bansard International	Data Aggregation	74kB

Data Exploration, Cleaning, Wrangling, and Engineering

Data Exploration Summary

To conduct our data exploration, we will be relying on tools such as Python and Tableau which have easy to use libraries and visualizations, respectively. Upon initial exploration of the OpenSky dataset, we immediately noticed quite a lot of empty entries for attributes, namely CallSign, Number (Commercial), and ICAO24 (transponder ID), Origin, and Destination. Several of these, such as CallSign, Origin, and Destination, we are considering as fields of importance, and would like to preserve as much information from as possible. We plan to further investigate if any aggregations of records (e.g. by airport origin/destination, aircraft type, etc) are affected by empty data in a statistically significant manner. In a similar fashion, we plan to identify any possible correlations and outliers, which may be linked to data collection errors.

The Johns Hopkins COVID dataset appears much cleaner at first glance, which we believe is because it is corrected for historical inaccuracies and as updated much more frequently. Nevertheless, we plan to conduct the same checks as listed above for the OpenSky dataset to determine if we would need to remove any extraneous records. Our steps to address these data inconsistencies will be described in the following section.

The Bansard Airline Code to Country file is mostly clean, with a very small number of empty entries. The file contains 253 distinct airlines and the countries they belong to, which should be accurate, given that the data is sourced from IATA and ICAO.

Data Preprocessing Approach

Some important factors to keep in mind are that the OpenSky dataset is provided as-is and both that and the Johns Hopkins data are updated on a monthly and daily basis, respectively, so incoming data must be preprocessed in the same fashion. Although both datasets contain some empty/dirty data as expected, we have devised a plan to mitigate the amount of unusable data and extract the data that would be of highest importance to us. Before resorting to removing records with empty entries, we would like to impute the values of the missing data. For example, given the latitude, longitude, and altitude of origin and destination, it may be possible to narrow down and fill in the missing airport codes. String values such as Country Names in the COVID and Airline Code to Country datasets will need to be checked for spelling/abbreviations so that the mapping is smoother. Some columns may end up being removed if they do not provide useful information, which will also help save space in the database and add another layer of efficiency to finish preparing the data for storage in the database.

Approach for Storing Processed and/or Integrated Data

For the processed and integrated data, we will store it in the InfluxDB, which is purpose-built for time series data. InfluxDB can store large volumes of time series data and quickly perform real-time analysis on that data.

For the raw data, we will use Amazon Simple Storage Service (Amazon S3), which is a scalable, high-speed, web-based cloud storage service to store processed and/or integrated data. As this service is designed for online backup and archiving of data and applications on Amazon Web Services (AWS), it can store and retrieve any amount of data from anywhere, which offers industry leading durability, availability, performance, security, and virtually unlimited scalability at very low costs.

Original Dataset	Processed Dataset Description
OpenSky_ Flight_Data	Description: Extraneous columns removed, missing values filled with imputed data, parsed timestamps Why: Decrease dimensionality and storage while preserving records that contain useful information, ensure required protocol in data flow
JH_Global_ Confirmed	Description: Transposed from original format, parsing time variable Why: Easier to query when dates are in row form
JH_Global_ Recovered	Description: Transposed from original format, parsing time variable Why: Easier to query when dates are in row form
JH_Global_ Deaths	Description:Transposed from original format, parsing time variable Why: Easier to query when dates are in row form
Airline_Code _to_ Country	Description: Mapping out missing attributes, such as Countries Why: Metrics taken need additional attributes

Processed Dataset Name	Input Datasets	Link to Processing Scripts and Notebooks	Provisional Data Size
OpenSky_ Flight_ Data_Final	OpenSky_Flight_ Data	Notebook/Script: nov_21_top_airline.ipynb	10GB
JH_Global_ Confirmed_ Final	JH_Global_ Confirmed	Notebook/Script: confirmed_covid_us.ipynb	2MB
JH_Global_ Recovered_ Final	JH_Global_ Recovered	Notebook/Script: recovered_us.ipynb	2MB
JH_Global_ Deaths_ Final	JH_Global_ Deaths	Notebook/Script: death_us.ipynb	2MB
Airline_Code_to_ Country_Final	Airline_Code_to_ Country	N/A	1MB

Approach for Feature Engineering and Data Modeling

We will perform our feature engineering in three steps. First, data preparation, which involves the manipulation and consolidation of raw data from different sources into a standardized format so that it can be used in a model. Data preparation may entail data augmentation, cleaning, delivery, fusion, ingestion, and/or loading. Second, exploratory analysis, which is used to identify and summarize the main characteristics in a data set through data analysis and investigation. Third, benchmark, which is setting a baseline standard for accuracy to which all variables are compared. This is done to reduce the rate of error and improve a model's predictability. Experimentation, testing and optimizing metrics for benchmarking will be performed.

For feature engineering, it consists of feature creation, feature transformation, feature extraction, and selection of features, also known as variables.

- Feature Creation: We will add some features which will be most helpful for our models. And we will remove some features which will be irrelevant to our models. We may need to combine features if needed.
- Feature Transformation: We will transform our features from one representation to another. This process may entail aggregation, attribute construction, discretisation, generalisation, integration, manipulation, normalisation, and smoothing.
- Feature Extraction: We will extract features from our data sets to identify useful information and compress the amount of data into manageable quantities for algorithms to process.
- Feature Selection: We will only consume these features we need and try to reduce the features that are redundant or irrelevant, which can actually negatively impact our model performance. Having a smaller number of features will make our model more interpretable and easy to understand.

Summary of feature sets: For all our features, they are measurable properties of the object we are trying to analyze. Each feature represents a measurable piece of data that can be used for analysis. Since features are the basic building blocks of datasets, the quality of the features in the dataset has a major impact on the quality of the insights we will gain when we use that dataset for deep learning. Through processes like feature selection and feature engineering, we are able to improve the quality of the dataset's features.

Link to Input Datasets	Link to Feature Engineering Scripts and Notebooks	Provisional Data Size
OpenSky_Flight_Data	Notebook/Script: nov_21_top_airline.ipynb	10GB
JH_Global_Confirmed	Notebook/Script:	2MB

	confirmed_covid_us.ipynb	
JH_Global_Recovered	Notebook/Script: recovered_us.ipynb	2MB
JH_Global_Deaths	Notebook/Script: death_us.ipynb	2MB
Airline_Code_to_Country	N/A	1MB

For data modeling, we will discover 4 deep learning methods that can be used to develop time series forecasting methods.

- MLPs: the classical neural network architecture including how to grid search model hyperparameters.
- CNNs: simple CNN models as well as multi-channel models and advanced multi-headed and multi-output models.
- LSTMs: simple LSTM models, Stacked LSTMs, Bidirectional LSTMs and Encoder-Decoder models for sequence-to-sequence learning.
- Hybrids: hybrids of MLP, CNN and LSTM models such as CNN-LSTMs, ConvLSTMs and more.

We will also discover 3 baseline methods that can be used to develop robust baselines for our time series forecasting problems.

- Simple forecast methods, such as naive or persistence forecasting and averaging methods, as well as how to optimize their performance
- Autoregressive forecasting methods, such as ARIMA and Seasonal ARIMA (SARIMA) and how to grid search their hyperparameters.
- Exponential smoothing forecasting methods, such as single, double and triple exponential smoothing also called ETS and how to grid search their hyperparameters.

Approach for Data Access

Initial design of data querying interface: data will be processed and loaded into influxDB for access and querying. To query the data, python API(influxdb-client-python) will be used to connect influxDB and query data via influxQL or Flux language. See supporting influxdb-client-python documentation[(2)].

We will programmatically access the data. InfluxDB time series data can be queried and graphed in dashboards. InfluxDB allows us to quickly see the data that we have stored via the Data Explorer UI. We can also use templates or Flux (InfluxData's functional data

scripting language designed for querying and analyzing), which can rapidly build dashboards with real-time visualizations and alerting capabilities across measurements.

Data Pipeline and Automation

Description of the Needs, Approach, Data Access, and Refresh Frequency

Current design of the data pipeline covers data flows from data source to a reliable data access/querying point, which includes:

- Sourcing Data: gzip files and CSVs will be downloaded from the website and saved to AWS S3 bucket <air-traffic-raw>, this process will be implemented with a DAG running cURL script in Airflow. The S3 lambda function will be triggered to unzip and save it back to S3 bucket <air-traffic-csv> whenever there is a new zipped file uploaded.
- Data Preprocessing: data will be extracted from S3 through boto3 SDK for preprocessing, which includes but is not limited to data cleaning, feature engineering, timestamp parsing.
- Data Aggregation: based on the metrics taken, preprocessed data next will be aggregated according to the need of analysis and modeling.
- Formatting tables: in order to import to influxdb, data will be further transformed, based on metrics taken, tables(aka measurements in influxdb) will be generated, attributes will be transformed to fields, tags will be added as needed.

Once the processed data was loaded into influxdb, that data will be accessed via python API (influxdb-client) with influxQL or Flux language.

The entire pipeline will be implemented and automated by DAGs using Apache Airflow. This will enable us to set a time scheduler to run specific tasks or steps. The air traffic data is currently updating monthly, we could refresh the data by month, or we may leave a holdout dataset for data streaming under the task management of Airflow.

Diagram of Major Data Pipeline Components

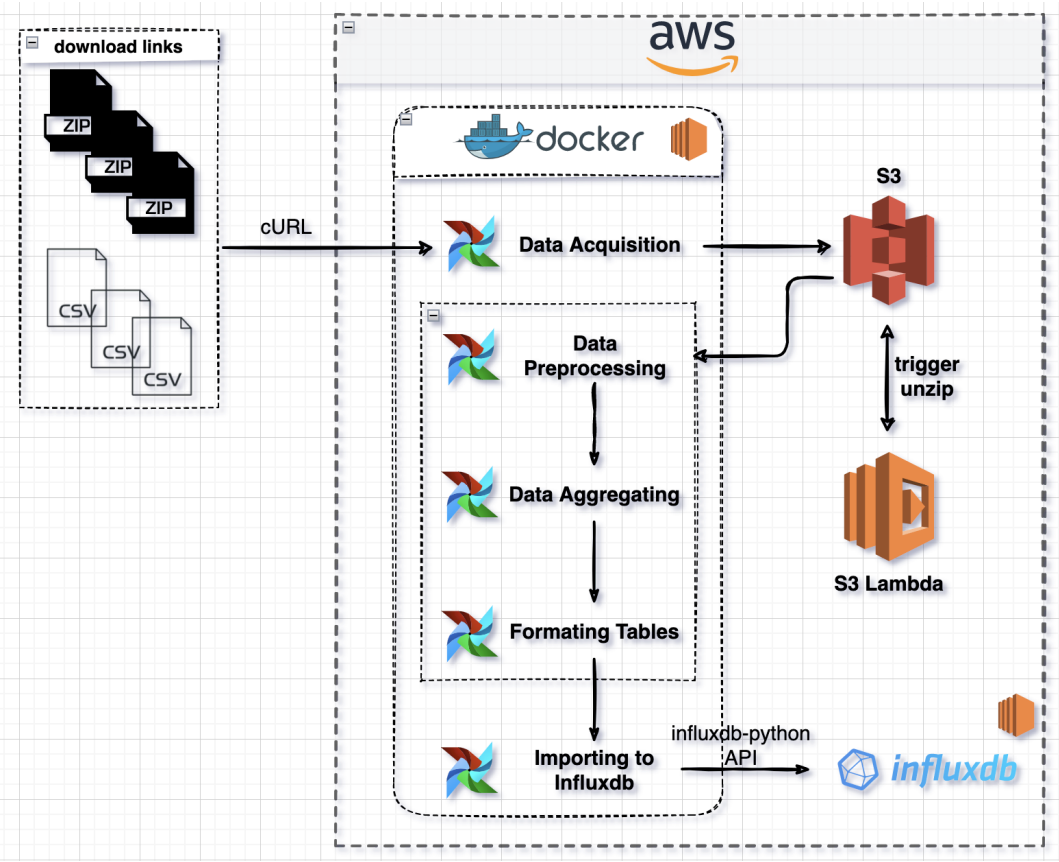

Figure 1. Diagram of Major Data Pipeline Components

Data Environment Set Up

- AWS S3 will store downloaded raw datasets, the lambda function will be set up on the targeted S3 bucket.
- An EC2 instance will be created for launching docker airflow server.
- An EC2 instance will be created for launching the influxdb server, grafana will also be launched in this instance if it's needed.

Team Member Contributions

Bo

- Maintained group "Capstone Project Planning"

- Contributed to “Approaches for Storing Processed and/or Integrated Data”, “Approach for Feature Engineering and Data Modeling”, and “Approach for Data Access”. Collaborated with Yuan on the pipeline and potential risks

Yuan

- Proposed air traffic dataset, explored and processed data
- Designed data pipeline diagram, set up AWS, airflow and influxdb server
- Contributed to “Data Pipeline”&”Data Environment Set up”

Adelle

- Coordinated meeting schedules between both professors as well as the group
- Set up outlines for report and presentation slides
- Contributed to “Raw Data Sources” and “Data Exploration, Cleaning, Wrangling, and Engineering” sections

Updates to Step 1

In our work since the previous report, we have narrowed down our project to focus on a single dataset to be able to solve a real-world problem while still incorporating benchmarking and performance metrics. The challenge we aim to address centers around the idea that many global industries were affected by the COVID19 pandemic, the air travel industry being one of the hardest hit. We plan to investigate these trends on the Country and Airline levels, and create a model that represents the spatiotemporal data in a comprehensive way. With this in hand, we will use a Deep Learning algorithm with a concentration in Spatiotemporal/Time Series data to predict future trends in the air travel industry. Once this is established, as a further exercise we plan to perform some benchmarking using other such Deep Learning models and create a Kaggle-like dashboard to track performance metrics of each model.

References

(1) Martin Strohmeier, Xavier Olive, Jannis Lübbe, Matthias Schäfer, and Vincent Lenders
"Crowdsourced air traffic data from the OpenSky Network 2019–2020"
Earth System Science Data 13(2), 2021
https://doi.org/10.5194/essd-13-357-2021

(2) Influxdb-client-python:
https://docs.influxdata.com/influxdb/v2.1/api-guide/client-libraries/python