The Impact of Covid-19 on Air Traffic: Spatiotemporal Forecasting with Deep Learning and Benchmarks

Adelle Driker, Bo Yan, Yuan Hu
Jacobs School of Engineering
University of California San Diego
{adriker, b3yan, yuh053} @ucsd.edu

Advisor: Professor Rose Yu
Jacobs School of Engineering
University of California San Diego
roseyu@eng.ucsd.edu

June 03, 2022

ABSTRACT

Many global industries have been affected by the COVID 19 pandemic, the airline industry being one of the most heavily hit. As a result, flight trends around the world have drastically deviated from their normal patterns. This presents an interesting scenario to explore, especially with the transience and abnormality of effects of the Covid data. The phenomenon also created uncertainty for both passengers and airline companies, especially due to the multiple waves of virus mutations prompting the following questions: How should airlines plan future flights? When should passengers schedule their travels? In other words, given a country's COVID situation, how should an airline/passengers plan ahead? To solve this problem, we present a solution in the form of an air traffic forecast using baseline models: AR and ARIMA, and deep learning models: LSTM (Long Short Term Memory), LSTM-mis-regression, LSTM-quantile-regression, and other models that capture the temporal complexity of such data. These algorithms are developed for time series analysis that are available to the open-source community, and can better serve airlines and passengers.

1 Introduction

Air travel comprises a large sector in the global economy, a critical component in travel, trade, and economic growth. To offer a sense of scale, the airline industry was valued at over $818 Billion in 2019,

and the US alone generated over $248 Billion of that amount, according to Zippia. When the novel Covid-19 virus began making its appearance in early 2020, many countries around the world began taking preventative measures to slow the spread of this disease. Such measures included closing borders, implementing masking measures, and prioritizing hospital bed and medical equipment availability. These abrupt changes in day-to-day life prompted airlines to drastically reduce the number of flights they offered, leaving many airlines struggling to operate. This situation, coupled with the multiple waves of Covid-19 gave rise to uncertainty amongst airline companies and future passengers - how should airlines plan future flights? When would be the best and safest days for passengers to travel?

Our team took a deep dive into exploring how a forecasting model, which utilizes flight and Covid-19 datasets and well as deep learning techniques, would be able to assist the airline industry in predicting future flights based on the Covid-19 situation in a certain country. Through data exploration and EDA, the team was able to better understand the intricacies of the datasets and clean and preprocess them in such a way that would allow for both the traditional and deep learning models to consume the data and provide unbiased results. Since the datasets consisted of time series data, special attention was devoted to sampling and creating a train/test split, especially since the Covid-19 is considered a transient event compared to the decades-long history of the airline industry.

1.1 Related Work

The models utilized within this project are part of a series of models within the TorchTS library, which was created by Professor Rose Yu's lab. More specifically, TorchTS is a deep learning based library built up from PyTorch which is specialized towards time series data and analysis. This library particularly emphasizes scalability and takes advantage of auto-differentiation and different inductive biases in the data. The TorchTS library is instrumental in a larger ongoing project which aims to standardize data pre-processing protocols and evaluation metrics for spatiotemporal data. The new framework will eventually incorporate various time series/spatiotemporal datasets for the open source community to test various models with, the results of which will be displayed on a leaderboard and will encourage further improvements within the time series/spatiotemporal deep learning domain.

2 Team Roles and Responsibilities

Our team is composed of three key members - Bo Yan, Yuan Hu, and Adelle Driker. Based on each team member's strengths and experience, we have assigned our roles in the following way: Bo, as the Record Keeper as well as the Software/ML/DL Engineer, will manage the project GitHub repo and help provide additional understanding in the Deep Learning domain with her previous experience in the field. Yuan will be the Budget Manager and Data Engineer, and will be in charge of tracking resources used by the team, mapping out the ETL and code automation process, and providing insight for the visualization portion of the project. Finally, as the Project Coordinator/Manager and Data/Business Analyst, Adelle will be responsible for maintaining contact between the Group and the Advisor, tracking project progress, and aiding with coding/analytical tasks.

Throughout the duration of the Capstone, our roles and responsibilities have slightly evolved based on the state of the project. As a team, this Capstone project also helps us build our team's values, core, and spirit. It's all about people. We work together. We are one team, we respect and trust each other. This project

helps us to learn the true reality of an end-to-end project. We will need to collaborate with team members who have various levels of experience on different parts of the project.

3 Data Acquisition

3.1 Data Sources

OpenSky Flight Data

Our foundational dataset, which is derived and cleaned from the OpenSky data source, contains spatiotemporal flight information data 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 expected to be updated for the remainder of the pandemic.

Johns Hopkins Global COVID19 Data

As one of the largest providers of public COVID-19 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.

Being able to access data from a variety of sources gives us greater freedom and enables us to see the bigger picture, while improving data quality and removing bias. While the problem may become more complex due to this, it also becomes more exciting to connect the dots to solve it.

3.2 Data Collection

The OpenSky Flight Network files amount to slightly over 10GB of data when unzipped but not preprocessed, and the Johns Hopkins Covid-19 dataset comprises about 2MB of data.

Our strategy to extract this data into our pipeline is as follows: Gzip files and CSVs will be downloaded from the OpenSky Network website and saved to an 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. The process to retrieve the Johns Hopkins Covid-19 data will be implemented in a similar manner.

3.3 Data Pipelines

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

a. Sourcing Data: see Section 3.2
b. 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.
c. Data Aggregation: based on the metrics taken, preprocessed data next will be aggregated according to the need of analysis and modeling.
d. 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.

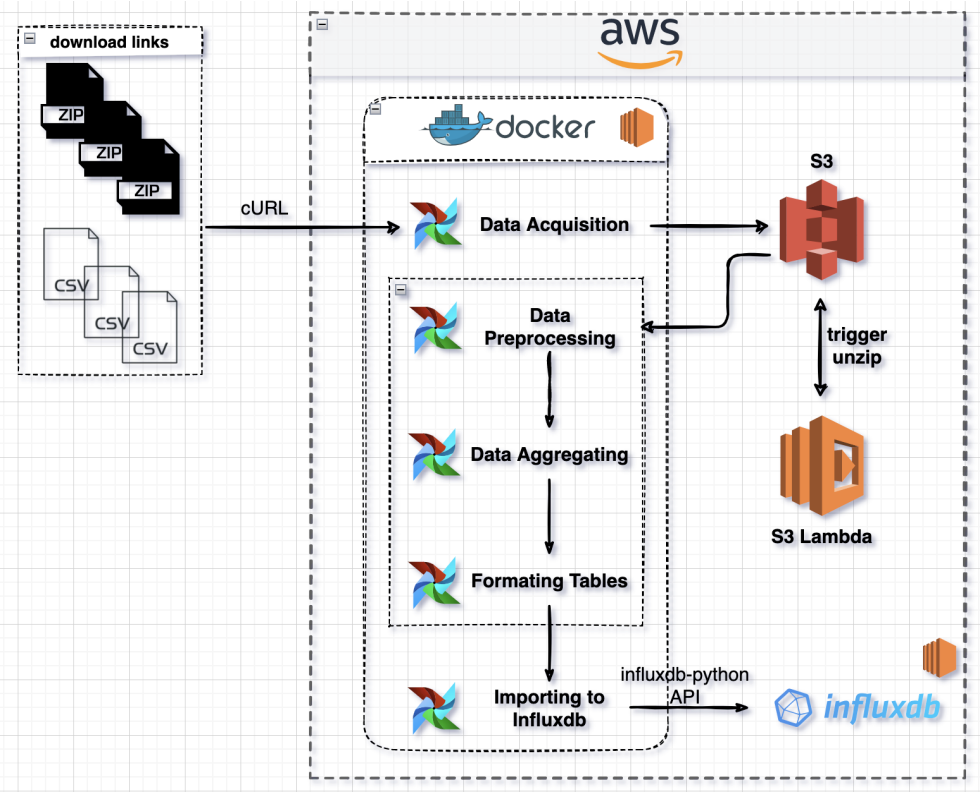
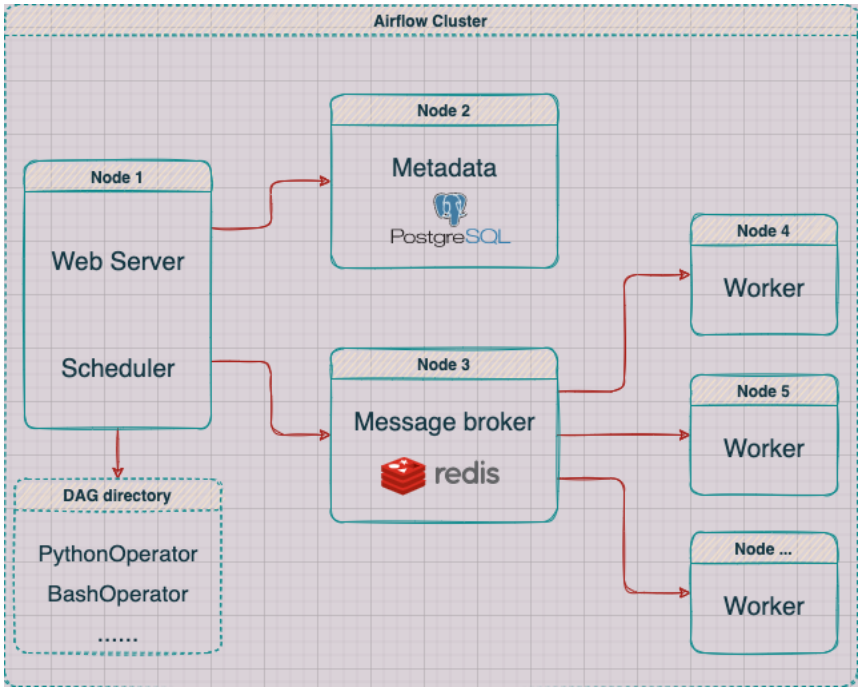


Diagram of Major Data Pipeline Components



Airflow Celery executor Architecture

3.4 Set up for Our Data Environment

We currently store the processed and integrated data in 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 are using 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.

Two buckets in S3 were created to store data, 'air-traffic-raw' stores raw zipped files, 'air-traffic-csv' stores automated unzipped csv files. Next, an S3 lambda function was configured to unzip files and the Influxdb database has been set up on an AWS EC2 instance. To ensure data transfer speed, 'gp3' was chosen and 'IOPS' was set to '3000'. This instance serves as the main data entry point when the modeling stage starts.

Initial Design of Data Querying Interface: data was processed and loaded into InfluxDB for access and querying. To query the data, the python API (InfluxDB-client-python) was used to connect InfluxDB and query data via influxQL or Flux language.

We programmatically access the data. For accessing the data, we use InfluxDB to query and graph 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.

4 Data Preparation

4.1 Data Preprocessing

Some important factors we had to keep in mind were 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 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 planned to impute the values of the missing data. For example, given the latitude, longitude, and altitude of origin and destination, it was 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 were checked for spelling/abbreviations so that the mapping is smoother. Some columns ended up being removed, as they did not provide useful information, which also helped to save space in the database and add another layer of efficiency to finish preparing the data for storage in the database. Below are more in-depth descriptions of how each dataset was preprocessed.

Open Sky Network Flight Data

The Data Preprocessing method for the flight data was broken into two main steps: merging the Flight and Covid data, and imputing as many missing values as possible. We have used a single file of flight data to create and test the data preprocessing method and expect this to apply to the other files with little to no additional adjustments. Using the Bansard Airline to Country Mapping data, we added two more columns - Airline Name and Country - to the Flight dataset with the help of the ICAO Designator.

Once completed, we created a list of distinct airport names and their GPS coordinates, which initially had very slightly different readings due to the nature of Open Sky Network's data collection process. To resolve this, we looked at one of two methods - order the entries in alphabetical order by airport name and take the first entry for each airport, or take the average of each coordinate reading for a single airport. The first idea, while easier to perform, may end up including dirty data and the second, while more computationally expensive, would allow for better accuracy when creating visualizations and models.

With the cleaned list of airports and their coordinates, we filled in more than half of the missing origin and destination airports. Since the records in the flight data are not dependent on each other when filling in the data, we plan to employ parallel processing to make the preprocessing more efficient.

Johns Hopkins Covid-19 Data

The Johns Hopkins Covid dataset contains a transposed format of data, with countries/regions in the first column and dates beginning on 01/22/2020 as columns holding cumulative counts of confirmed cases, recoveries, and deaths. In order to be stored in the InfluxDB database for easier querying, the time series data will need to be transposed once more so that the date columns will become rows with NULLs replaced with zeros. For EDA purposes however, the file was loaded into a pandas dataframe with NaNs

replaced with zeros. We expect any preprocessing of the Covid data to be smooth due to the consistency and regular maintenance by Johns Hopkins.

4.2 Feature Engineering

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.

We have performed our feature engineering in three steps. First, data preparation, which involved 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 entailed data augmentation, cleaning, delivery, fusion, ingestion, and loading. Second, exploratory analysis, which was used to identify and summarize the main characteristics in a data set through data analysis and investigation.

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

Feature Creation: We have added some features such as Country and Airline Name to our flight dataset, and we removed some features which will be irrelevant to our models, such as altitude and IATA Designator.

Feature Transformation: We transformed our features from one representation to another. This process entailed aggregation, attribute construction, discretisation, generalization, integration, manipulation, normalization, and smoothing.

Feature Extraction: We extracted 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 have only consumed the features we need and aimed to reduce the features that were redundant or irrelevant, which can negatively impact our model performance. Having a smaller number of features will make our model more interpretable and easy to understand.

5 Analysis Methods

5.1 Preliminary Analysis

For identifying methods to perform preliminary analysis of our data, we found that files from different data sources mostly share the same data qualities. For example, after loading in several more files of flight data, we found that callsigns, latitudes and longitudes of origin airports consistently hold NaN values, and latitude and longitudes of destination airports have fewer than 100 empty values. This means that the origin and destination airport names will be the main focus of data imputing. The Covid dataset is much cleaner since it is consistently maintained.

5.2 Significance of Methods

Given that we are working with time series data, we will need to incorporate the timestamp (yyyy-mm-dd) as one of the main input features for each data point. Another key identifying feature is the Country of the airline of each flight in addition to the number of Covid cases. The target variable will be the number of flights originating from a Country on a given day. We were experimenting with a few models to get a sense of which one would perform best on this kind of dataset. Using these methods can significantly help us identify the major factors that can be used to build our models and resolve the problem we defined.

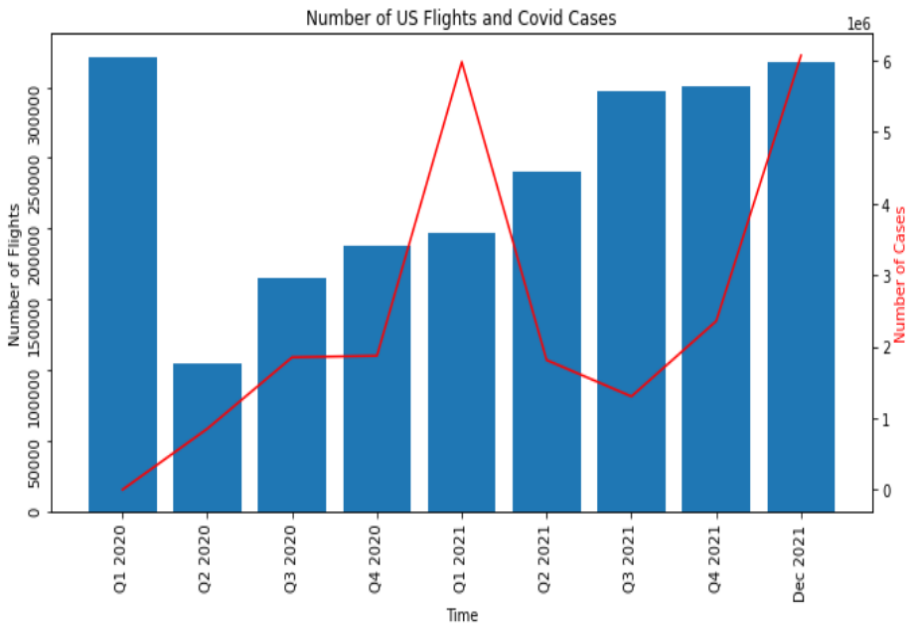
5.3 Influence on Design

By analyzing the datasets and applied preliminary methods to explore the datasets, we got general ideas of how to design our whole system based on this information. Also, the features and data volume also helped us think of scalability and robustness of our system, which significantly benefited ourselves in designing the data pipeline and modeling pipeline. Also, the EDA results also helped us in choosing the right models and defining the right parameters for some models.

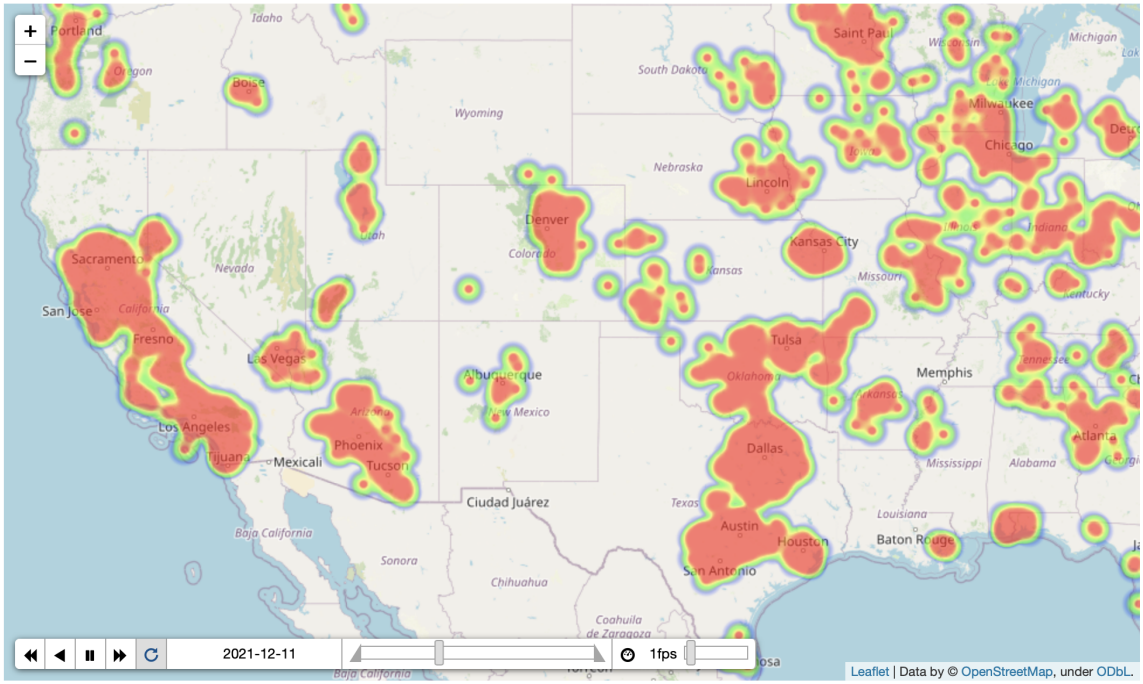
5.4 Applying Analysis Techniques

We began our analysis by conducting EDA on the flight and COVID-19 datasets to get a better sense of key trends and patterns:

An analysis of the time series data from both the flight and Covid perspective in the US shown in the figure to the left shows an expected dip in the beginning of 2020, just as Covid began. A period of uncertainty persisted into late 2020 as cases continued to climb and airline companies tentatively began to allow more flights to resume. Around Q1 2021, a large spike is apparent due to a major Covid surge after the holiday season and a dip ensues as vaccines begin to become available to the public. Another wave begins between Q3 and Q4 2021 and grows into the Delta and eventually the Omicron waves. Although the Covid data depicts significant peaks and dips, the airline data portrays a single dip at the beginning of the pandemic and a steady recovery afterwards. As a result, the data shows that the airline data is mostly independent of the Covid data as additional factors such as vaccines, the holiday season, and increased testing availability in parallel with federal/global guidelines affected the trend.



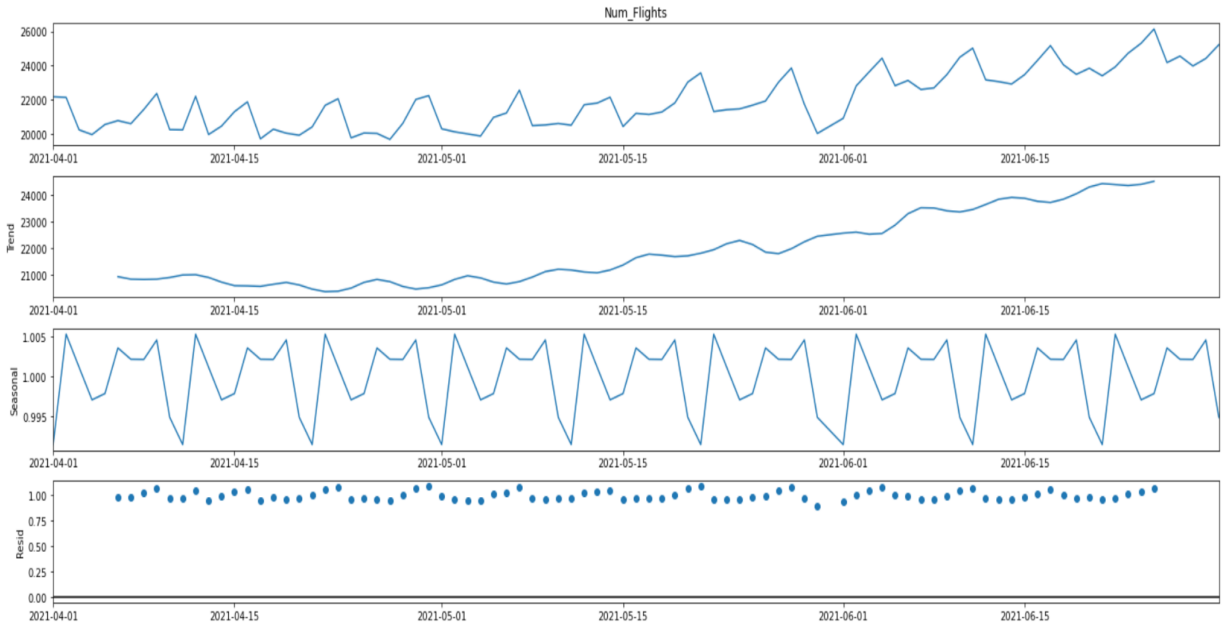
Flight trend heat map representation



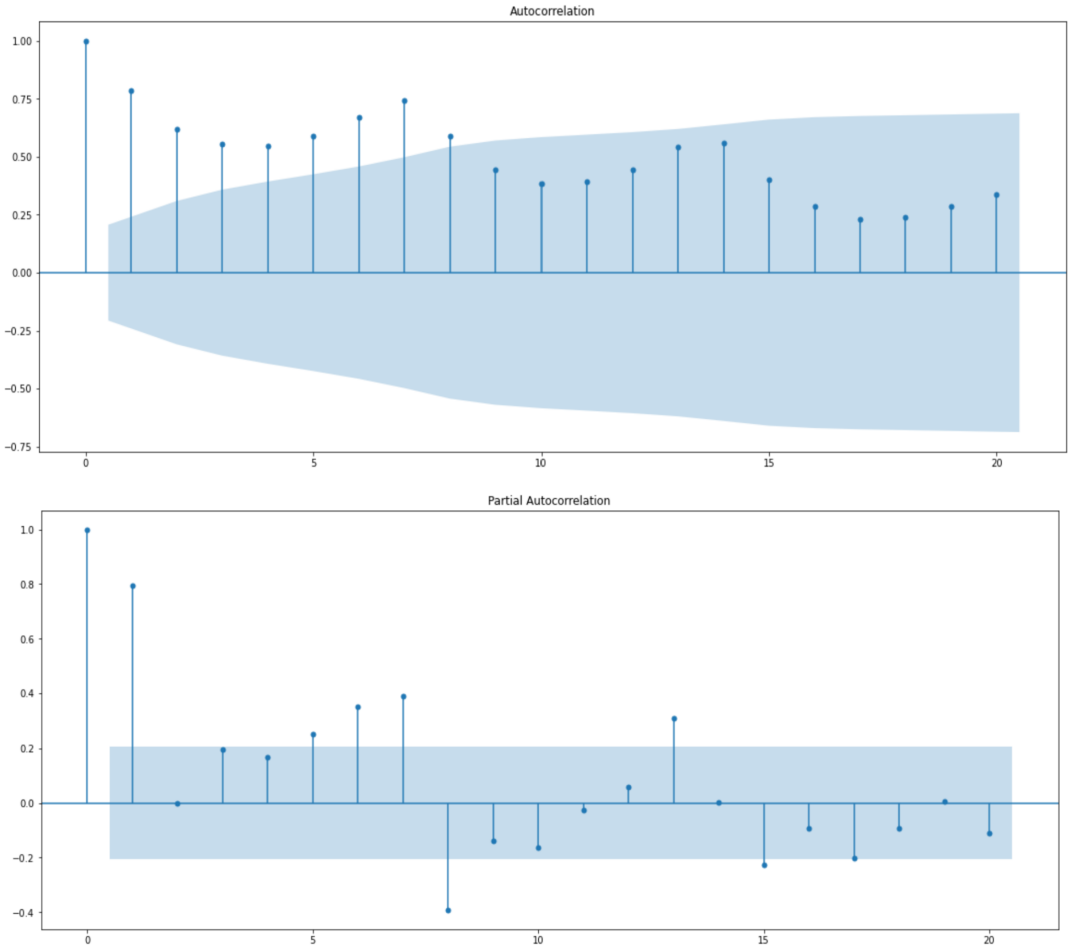
The figure above represents the number of departing flights from the majority of US airports. As of now, the data that is shown only covers Dec 12, 2021, and the time sliding interval has been set to days. Once all the data has been processed, the animation will cover the span of the entire pandemic with a monthly sliding window which will help monitor a whole picture of flight trends after the outbreak of COVID-19.

Flight trajectory network representation using Gephi

This directed node-link network exhibits worldwide flight trajectories on Dec 26, 2021. Nodes represent source and destination airports, edges represent a flight trajectory from source airport to destination. Node size indicates its degree of centrality, which evaluates the number of flights from the same source airport. A gradient color palette was chosen to help distinguish the geo-location of different continental and countries. We can see that most flights are concentrated within North America (specifically the US) and Europe, and in between the two continents.



Decomposition of Flight Data



Autocorrelation Plots

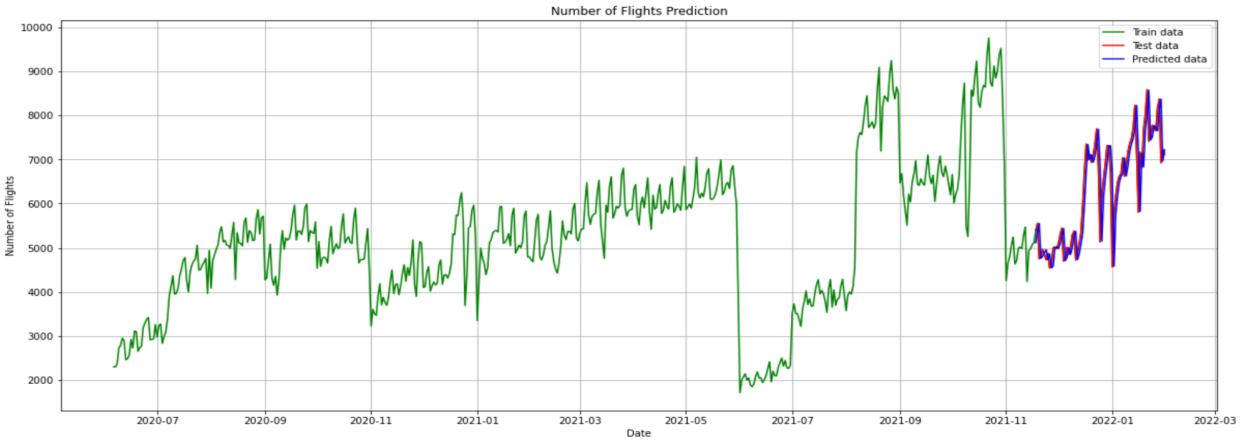
Models

We utilize three core models: two traditional time series models - AR and ARIMA - and a deep learning model - LSTM. For additional information, we have also implemented LSTM with Mean Interval Score Regression and LSTM with Quantile Regression models. The LSTM with Mean Interval Score Regression model uses a mean interval score loss function with a LSTM network. And the LSTM with Quantile Regression modeling uses a quantile loss function with a LSTM network. We continued using different deep learning models to train against our whole dataset and tuned performance. Below is a table of error scoring for the three core models, with additional detailed information about them and the LSTM variants shown below.

Models	ME	MSE	MAE	RMSE	R^2 score
AR	1,530.095	259,351.994	381.051	509.266	0.808
ARIMA	1,782.427	240,363.422	339.022	490.269	0.845
LSTM	1,982.785	313,407.582	427.684	559.828	0.768

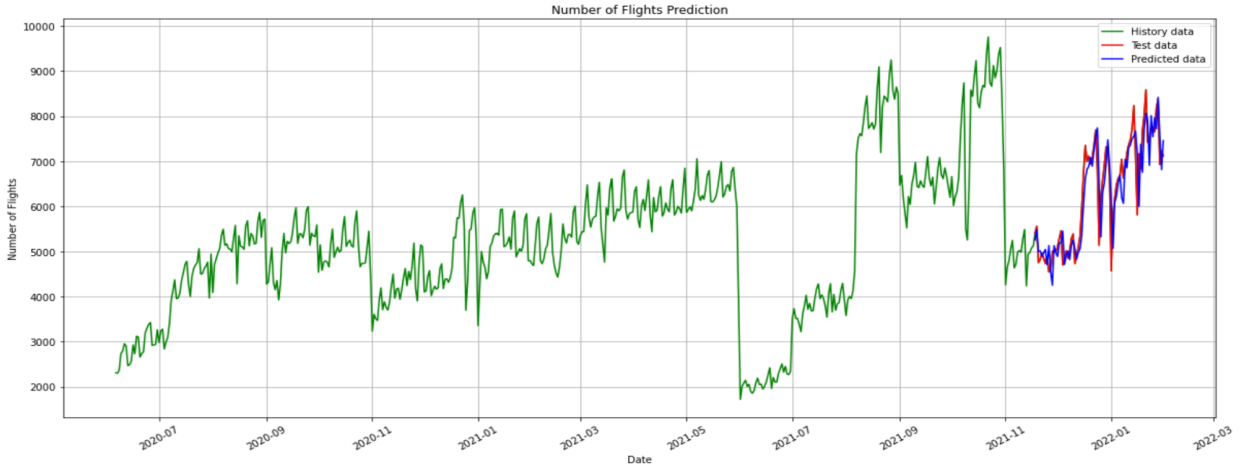
Baseline Models:

- **Autoregressive (AR) Model**
  - Training: AR(11), CMLE



Prediction Results for AR Model

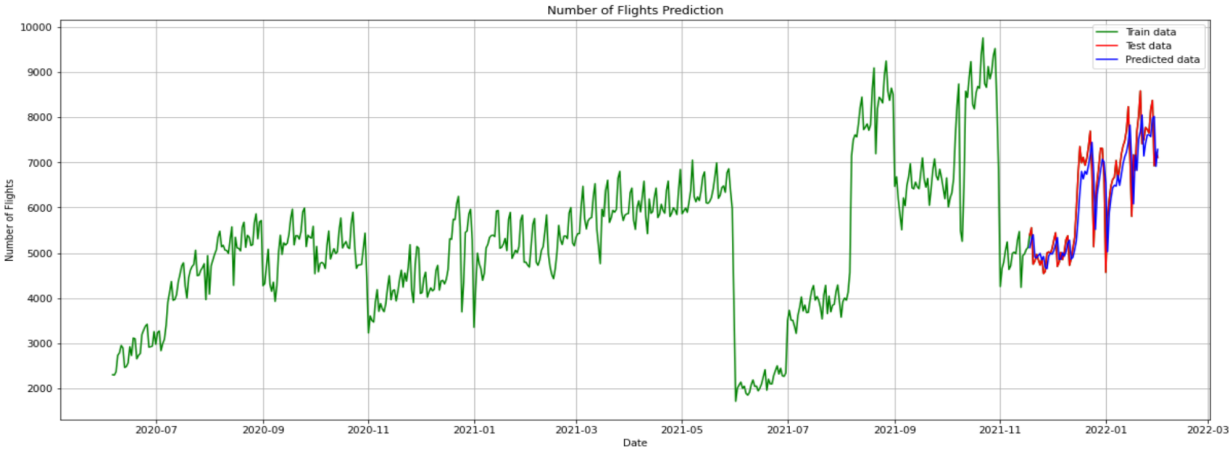
- **ARIMA Model**
  - Training: ARIMA(1, 1, 0), CSS-MLE



Prediction Results for ARIMA Model

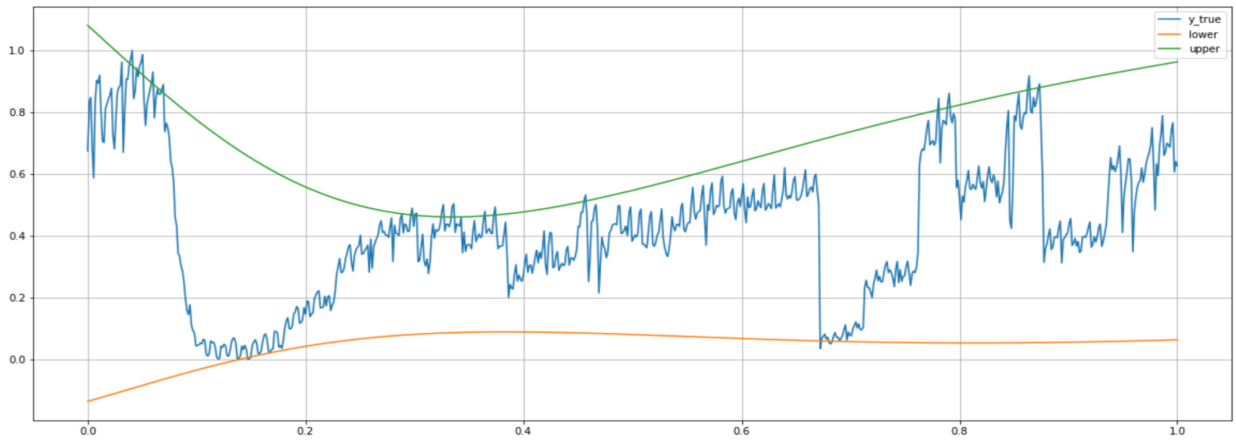
Deep Learning Models:

- **LSTM**
  - Training: sequential, LSTM
  - Scoring: RMSE score
  - Learner Parameterization:
    - Optimization algorithm: Adam optimizer
    - Drop-out rate: 0.2, 0.25
    - Batch size: 25
    - Units: 100

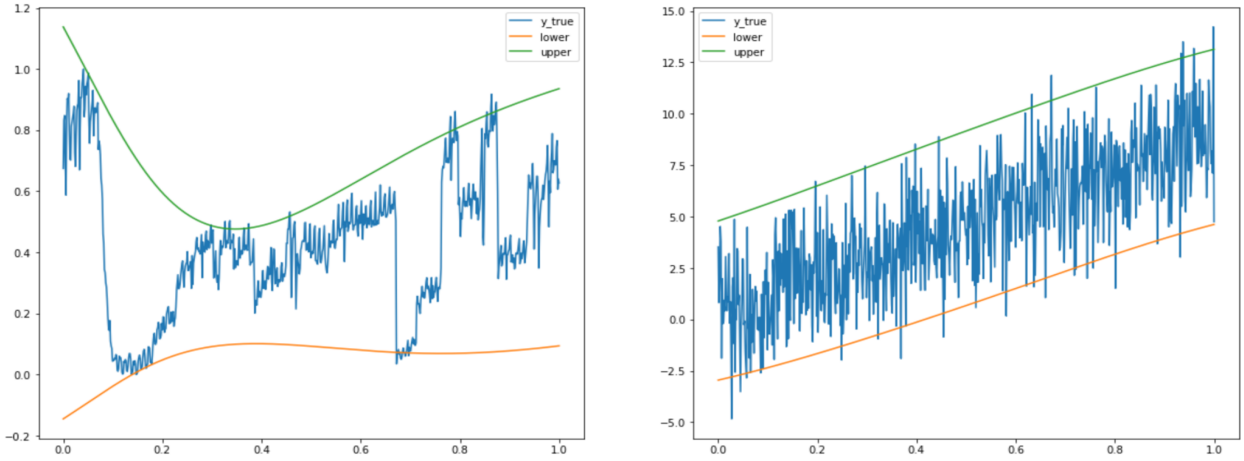


Prediction Results for LSTM

- **LSTM-mis-regression model**
  - Training: LSTM
  - Scoring: Mean interval Score
  - Learner Parameterization:
    - Input size: 1
    - Output size: 1, 2
    - Hidden size: 16
    - Interval: 0.95
    - Optimizer: optim.Adam
    - lr: 0.005
    - Epochs: 100
    - Batch size: 10

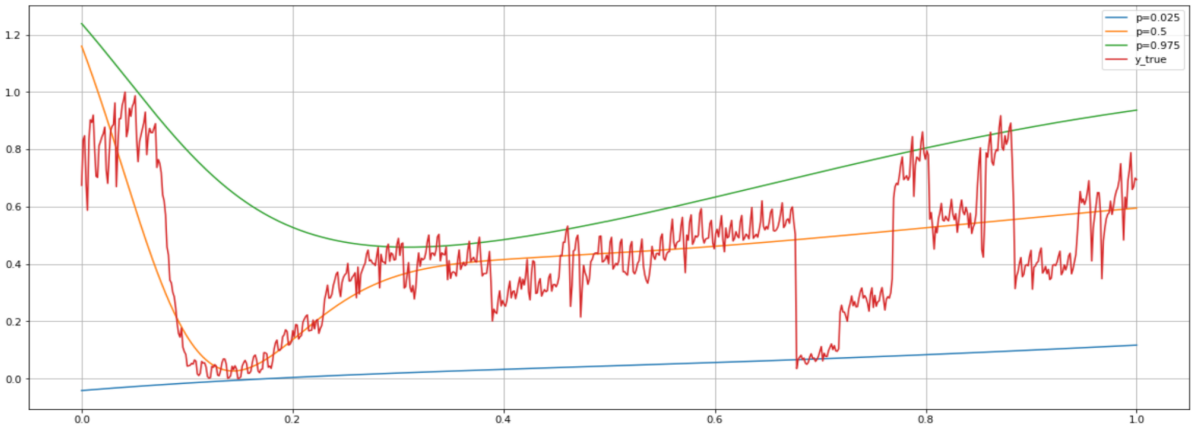


Prediction Results for LSTM-Mis-Regression Model with One Confidence Interval



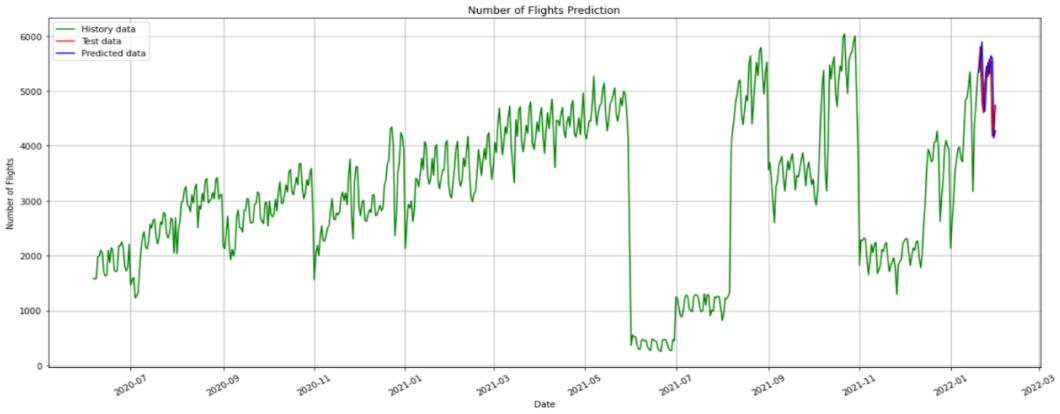
Prediction Results for LSTM-Mis-Regression Model with Two Confidence Intervals

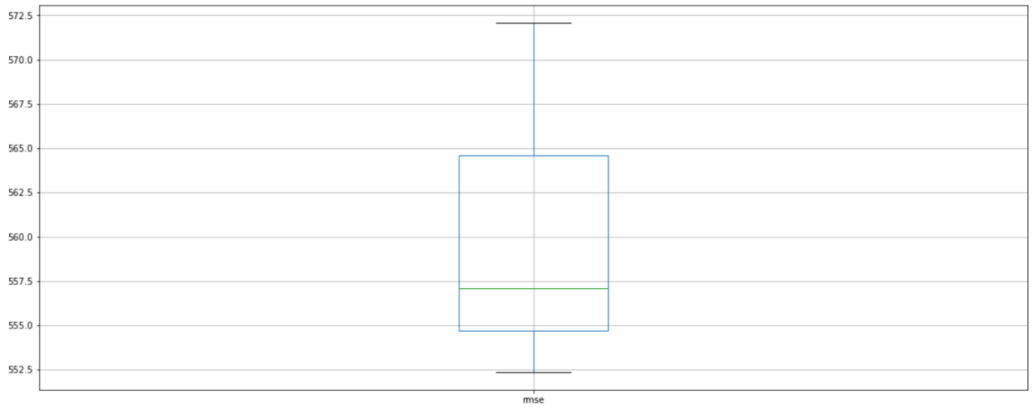
- **LSTM-quantile-regression model**
  - Training: LSTM
  - Scoring: quantile loss function
  - Learner Parameterization:
    - Input Dim: 1
    - Output Dim: 1
    - lr: 0.01
    - Quantiles: 0.025, 0.5, 0.975



Prediction Results for LSTM-Quantile-Regression Model

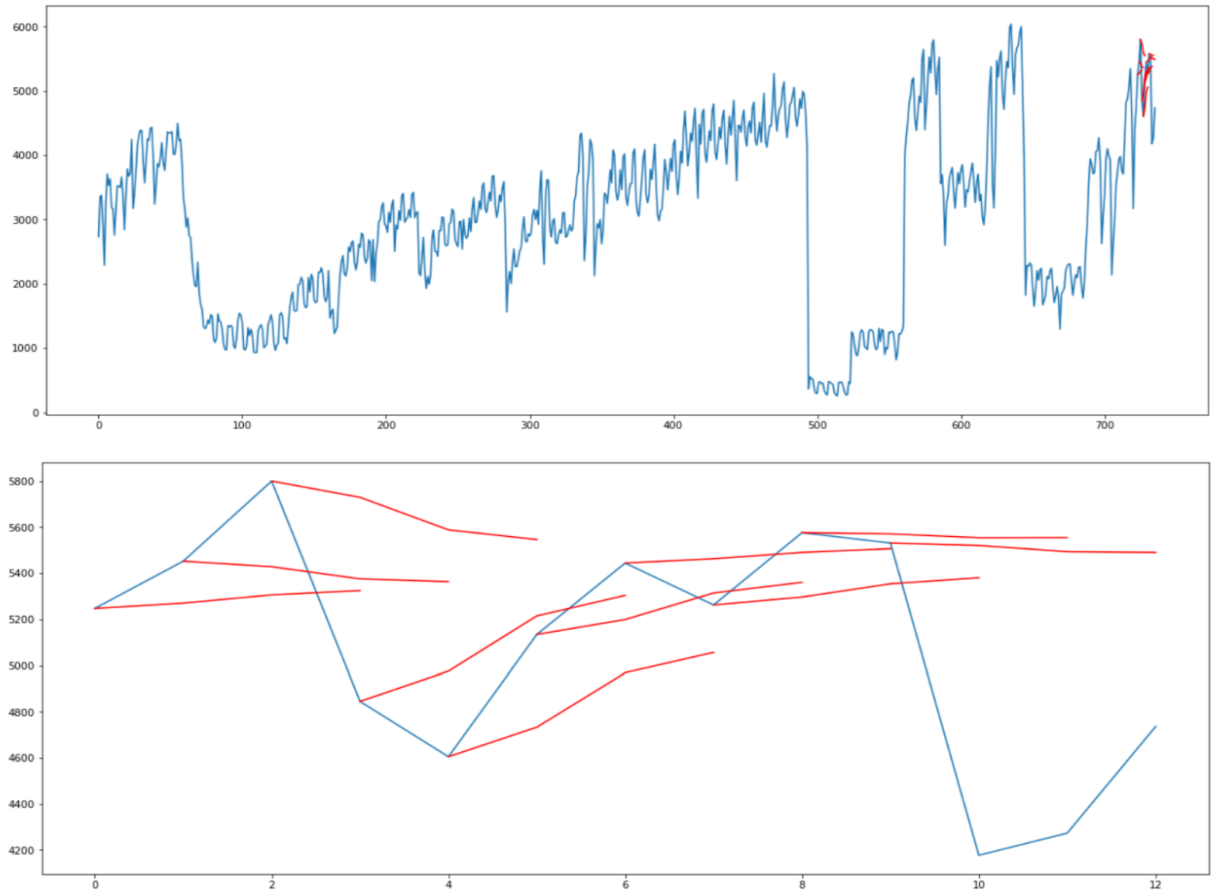
- **LSTM-Univariate model**
  - Training: LSTM
  - Scoring: RMSE
  - Learner Parameterization:
    - n_repeats: 3
    - n_epochs: 3
    - n_batch: 1
    - n_neurons: 4





Prediction Results for LSTM-Univariate Time Series *Model*

- **LSTM-Multi-Step model**
  - Training: LSTM
  - Scoring: RMSE
  - Learner Parameterization:
    - n_lag: 1
    - n_seq: 3
    - n_test: 10
    - n_epochs: 1500
    - n_batch: 1
    - n_neurons: 1



Prediction Results for LSTM-Multi-Step Time Series *Model*

As we can see from the prediction results for the LSTM-Multi-Step Time Series model, a line plot of the test values (blue) with the forecasts (red) is also created. This is marked by the fact that the t+2 appears easier to forecast. We can see that the results at each forecasted time step are better, in some cases much better. The plot shows that although the skill of the model is better, some of the forecasts are not very good and that there is plenty of room for improvement.

Model Interpretation

- The LSTM-MIS-Regression model results consistently tend to stay within the estimation corridor, and wherever the plot travels beyond the confidence intervals shown in blue and green indicates that the confidence interval becomes narrower. A larger sample will reduce the sampling error, give more precise estimates and thus smaller intervals.
- The LSTM-Quantile-Regression modeling results show that the required quantiles, 0.025, 0.5 and 0.975, have 3 output nodes, with each node having a different loss function. This ensures that the structure of the data is shared in the first few layers.

5.5 Basic Analysis Techniques

1. We started with existing data.
    a. We used the existing real data to learn from. In order to train the computer to understand what we want and what we don't want, we prepared, cleaned and labeled our data. We got rid of dirty entries, missing pieces of information, anything that's ambiguous or confusing.

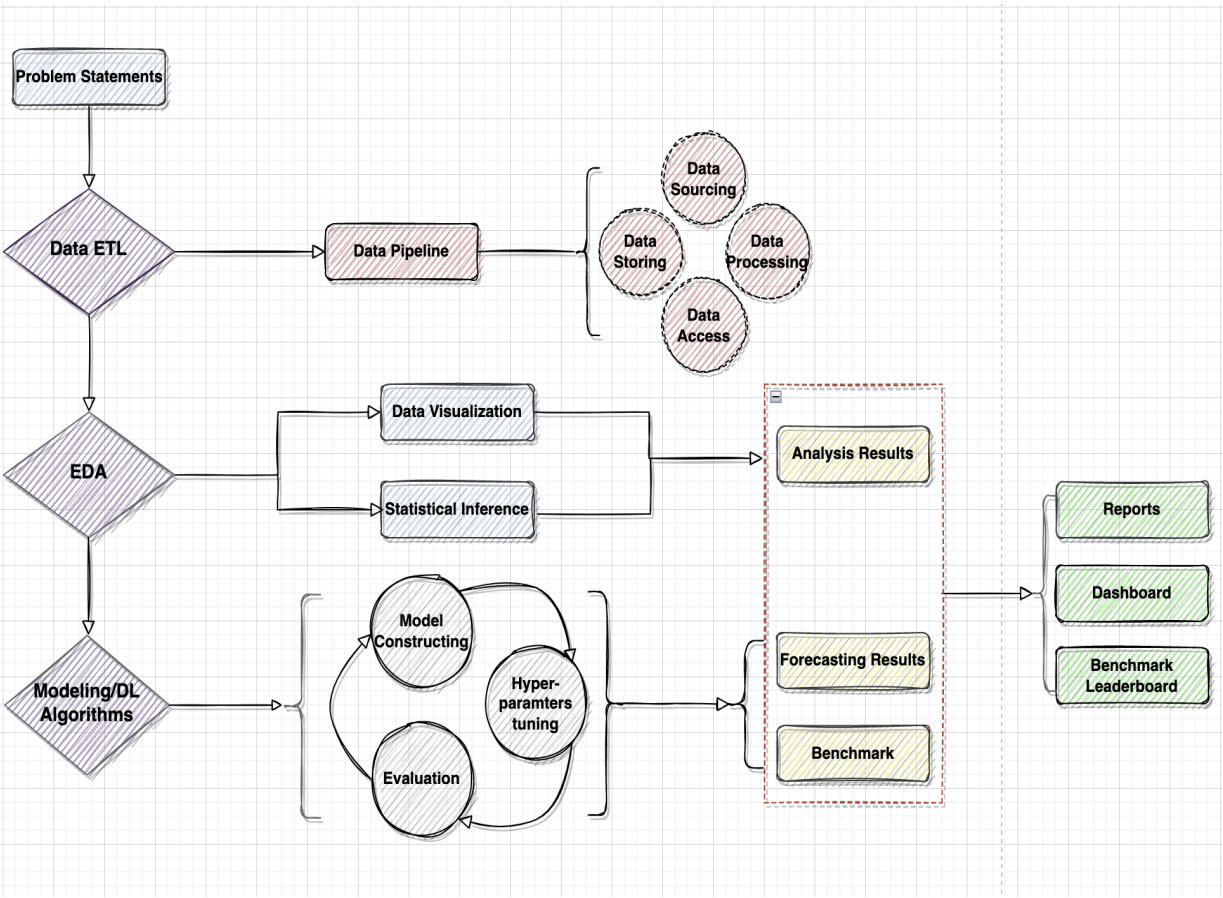
2. We analyzed data to identify patterns.
    a. Based on our EDA results, we have chosen the right algorithms, applied them, configured them and tested them. To make the right choice, we experiment with a few algorithms and test until we find the one that gives us the results most aligned to what we're trying to achieve with our data. After that, we successfully applied a machine learning/deep learning algorithm to analyze our data and learn from it, with a trained model.
    b. We decomposed the cleaned dataset and created autocorrelation and partial autocorrelation plots to help us identify the trends and correlations, which help us find the optimal parameters for different models. Also, we did feature scaling, built RNN, compiled RNN, and fit RNN to the training set and did the prediction for our deep learning algorithm.

3. We made predictions.
    a. The regression is supervised types of algorithms, we need to provide intentional data and direction for the computer to learn. We played around with each algorithm type and use case to better understand probability and practice splitting and training data in different ways.

5.6 Analytical Workflow

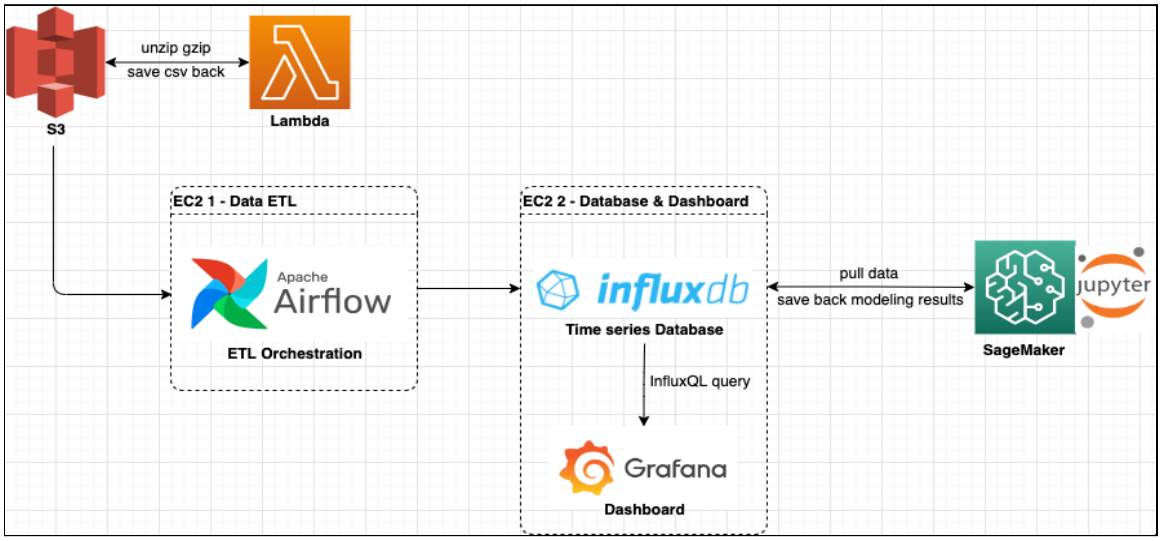
1. We defined our problem first, which was made clear to our stakeholders.
2. Then we started looking for the most suitable datasets that best fit for our problem definition.
3. As we found the right datasets, we started exploring the datasets and preprocessing our datasets. (Data ETL and Data EDA)
4. Based on the findings on our datasets, we started creating models to resolve the problem we defined earlier.

5. Then we created a dashboard to visualize the findings and results to better serve the stakeholders to help them make better decisions.



Project Workflow

5.7 Set Up for Processing Environment



Project Architecture

1. AWS S3 stored downloaded raw datasets, the lambda function was set up on the targeted S3 bucket.
2. An EC2 instance was created for launching docker airflow server.
3. An EC2 instance was created for launching the influxdb server, Grafana was launched in this instance if it's needed.

6 Findings and Reporting

6.1 Findings

1. Using the AR and ARIMA models as the baseline models can reflect the trend information and there is a trend component which grows the flight number month by month. The prediction results for the AR and ARIMA models set the baseline for our project, which we can use these predictions to measure the baseline's performance and then become what we compare all machine learning and deep learning algorithms against.
2. The reported performance for these models also reflects that we need to think about other factors such as seasonality, delay of reported covid cases that have effects on the schedule of flights, and so on. After analyzing the data, we can see that there is a seasonal component which has a cycle less than 2 weeks. The variance in the data keeps on increasing with time.
3. Using the LSTM model to train the data can reflect the trend information. The prediction results for the LSTM model shows that our model performs well.
4. The decomposed data shows that the trend and seasonality information extracted from the series does seem reasonable. The residuals are also interesting, showing periods of high variability in around 7 days of the series.
5. A gradual decrease is shown in the Autocorrelation plot and a sharp cut-off in the Partial Autocorrelation plot. These two plots help us find the optimal parameters.
6. The LSTM-MIS-Regression model results consistently tend to stay within the estimation corridor, and wherever the plot travels beyond the confidence intervals shown in blue and green indicates that the confidence interval becomes narrower. A larger sample will reduce the sampling error, give more precise estimates and thus smaller intervals.
7. The LSTM-Quantile-Regression modeling results show that the required quantiles, 0.025, 0.5 and 0.975, have 3 output nodes, with each node having a different loss function. This ensures that the structure of the data is shared in the first few layers.
8. The prediction result of LSTMs for Univariate Time Series Forecasting performs quite well. The box and whisker plot shows the distribution which captures the middle of the data as well as the extents and outlier results.
9. The prediction result of LSTMs for Multi-Step Time Series Forecasting performs well too. As we see, the lines connect to the appropriate input value for each forecast. It shows that although the skill of the model is better, some of the forecasts are not very good and that there is plenty of room for improvement.
10. After applying deep learning approaches to our forecasting product, we found that the deep learning models performed quite well and provided forecast results with high accuracy. We are able to measure this success by comparing the results against classical baseline models - AR and ARIMA. The success was quantitatively measured by taking the R^2, RMSE, and MAE values to

confirm that the deep learning models performed as well as and better than the baseline models we chose.

6.2 Determining What to Present

In our reporting, we present our findings mainly focusing on the following points.

a. The architecture of our system design.
b. The models we choose to create and performance results.
c. The dashboard we created to show our visualizations.
d. The findings of our capstone project and how these findings benefit our stakeholders.
e. Future works.

Our reports that are featured on the dashboard have been compiled in such a way that our target audience will be able to confidently make important business decisions. Although some models that we worked with, such as Seq2Seq, did not bring anticipated results, we plan to continue to improve those results and eventually incorporate them into the dashboard as well.

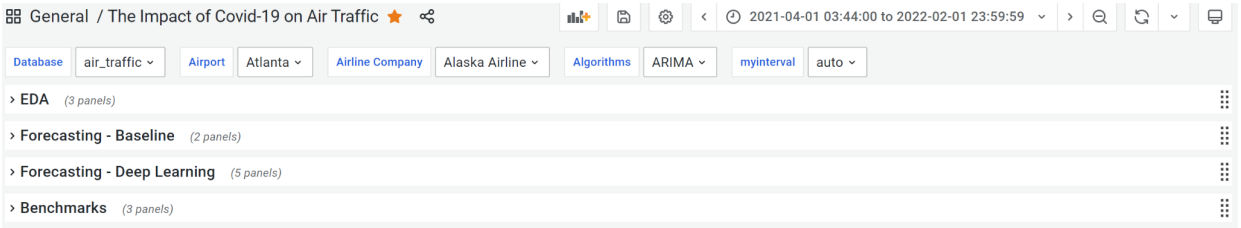
6.3 Techniques and Tools

In order to create the dashboard, we used a tool called Grafana, which is an open-source web application suited for analytics and interactive visualizations. Grafana allows us to both link to our InfluxDB database which stores our raw data and create EDA-like visualizations as well as upload csv files with our forecast modeling results to display our final product. To access and view the dashboard, a user will need to have an account to log in, thus adding a layer of security as well.

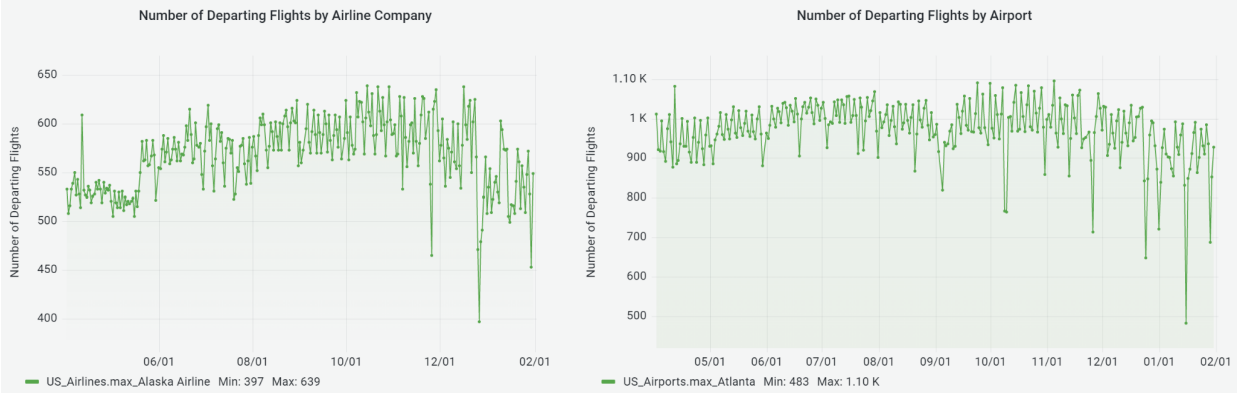
6.4 Visualizations and Reportable Products

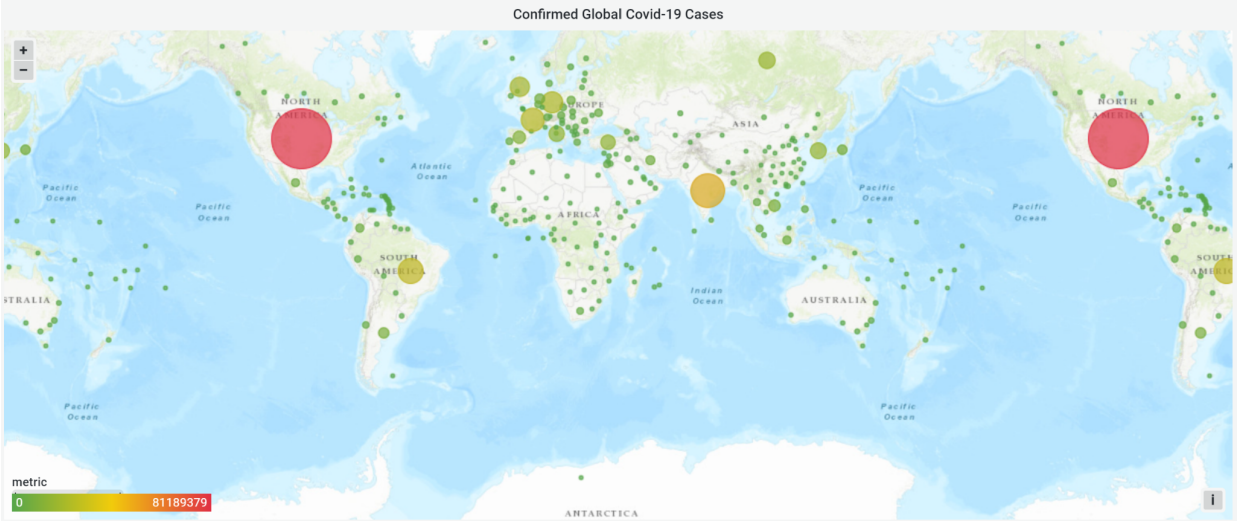
Visualization from Product Dashboard:

Our dashboard consists of several visualizations, each of which can be controlled by a set of filters at the top. A user may select to view a set of visualizations of their choice - EDA or Forecasting.



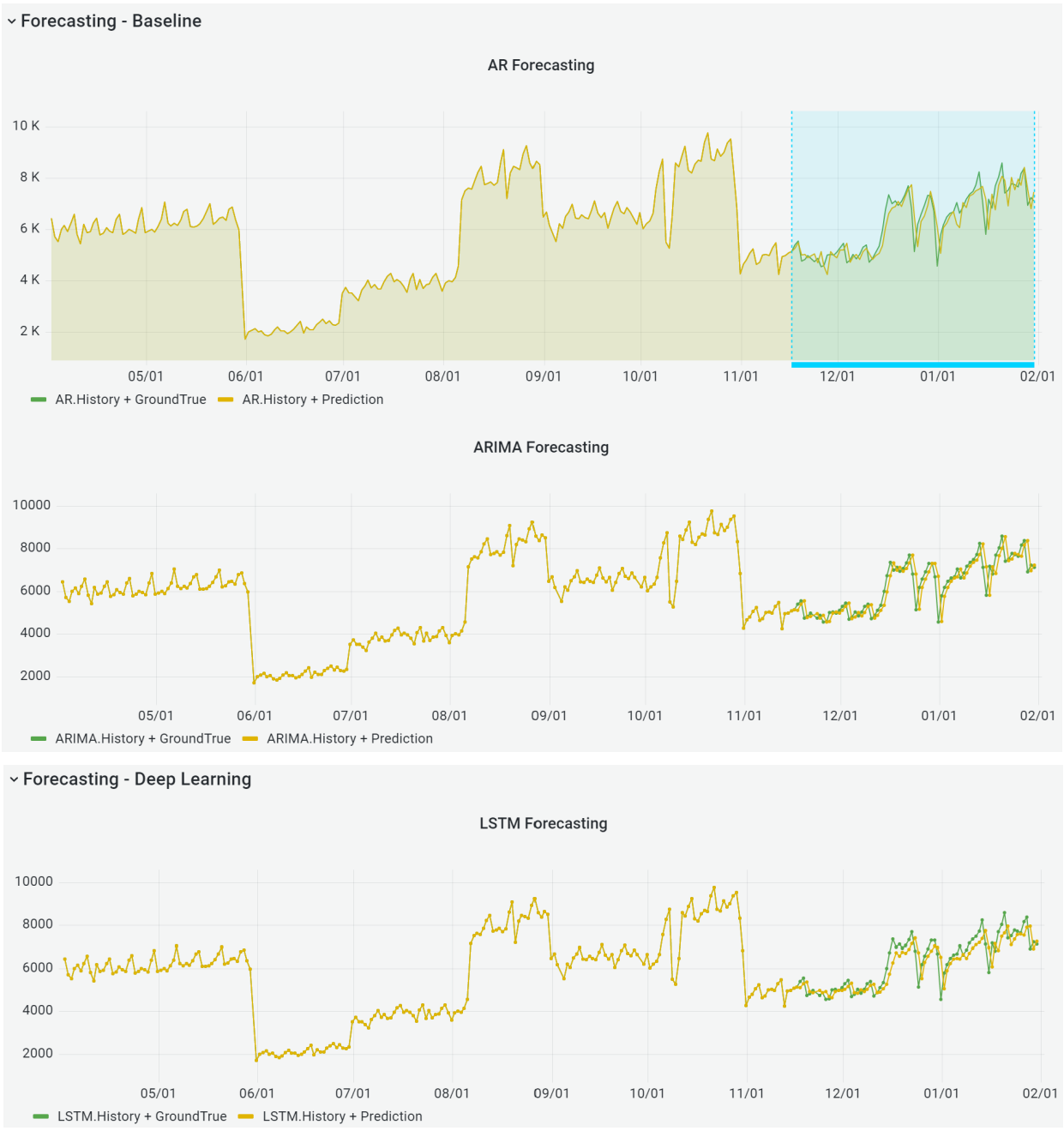
Each plot can be filtered by date both by using the filters on top and by dragging the blue highlighted window as shown below. Additionally, a user may filter by airline type or hover over a point on the plot to reveal the quantitative information behind the data point such as the date, time and number of flights around the world.

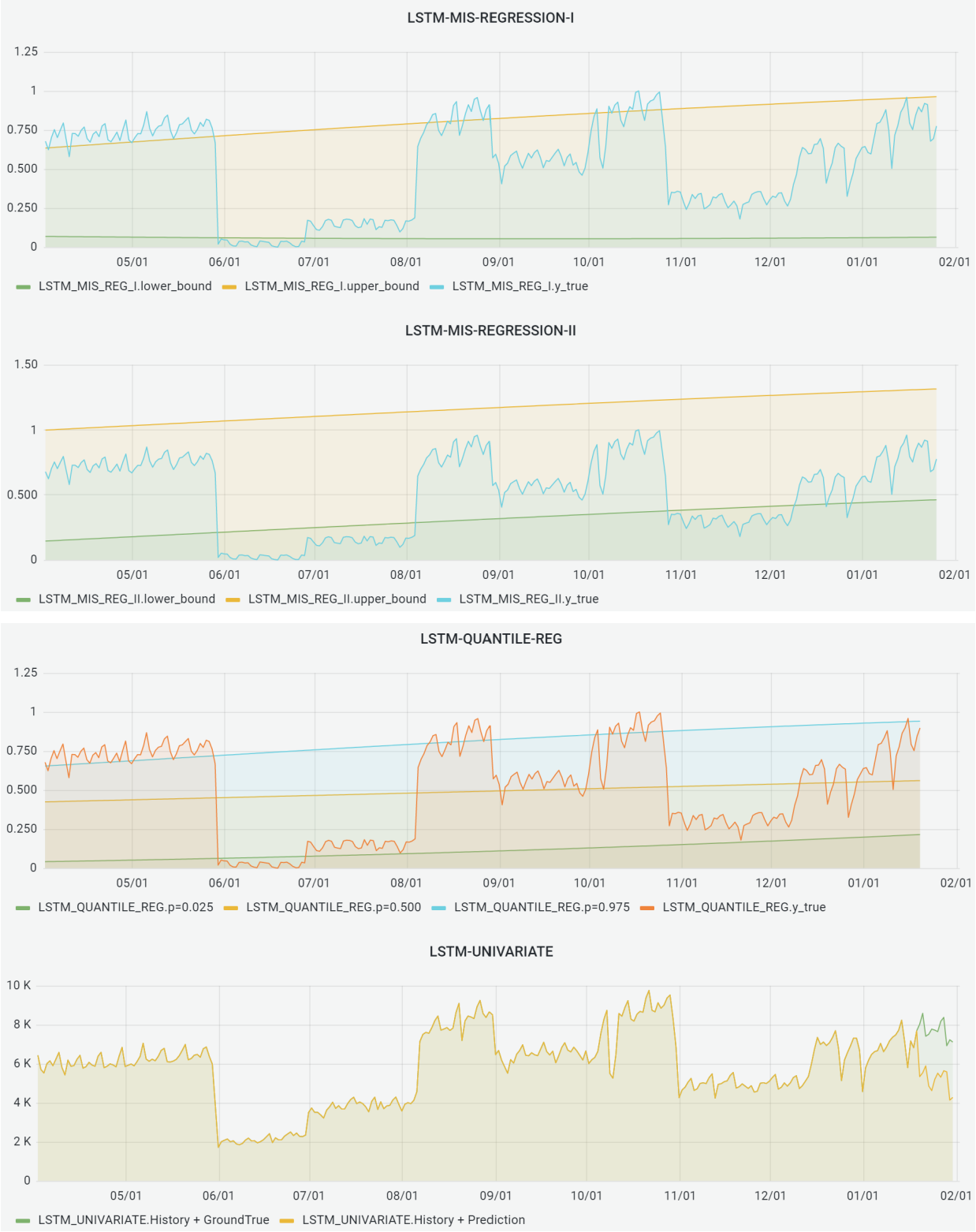


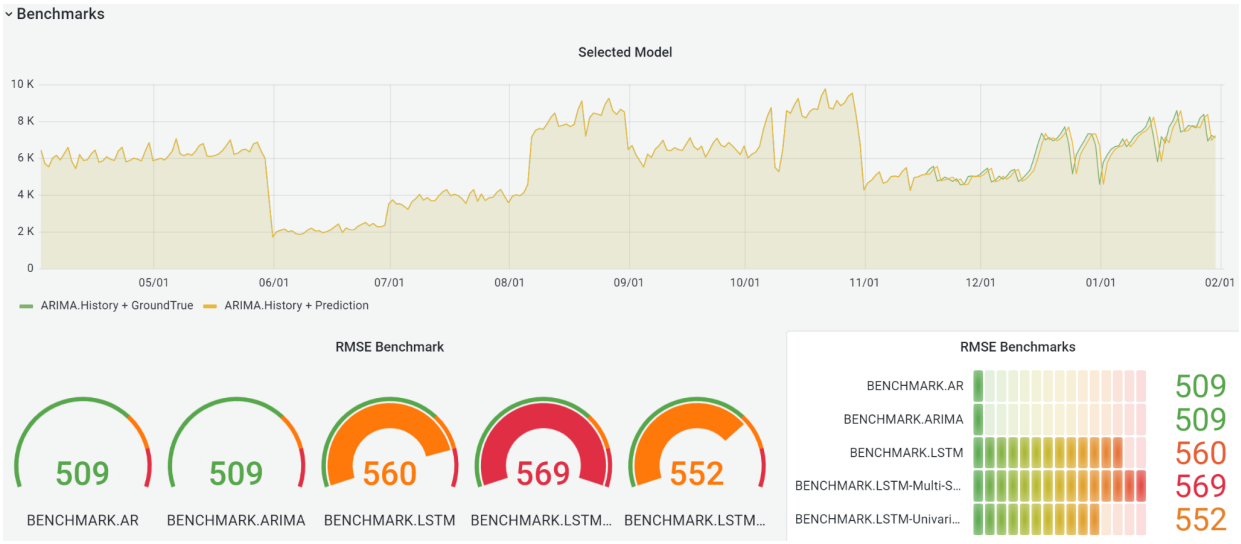


COVID-19 data at the country level is also available in parallel in order for the user to track the number of covid cases in both origin and destination of travel. A user may click and drag the map to change location as well as zoom in and out to see more granular data. The amount of Covid cases are also encoded with a color scale where red is the more severe, while green is very low.

Once the user selects Forecasting, they will see the following figures, which contain the forecasting results from the different models. Each plot can be filtered by date both by using the filters on top and by dragging the blue highlighted window as shown below. Additionally, a user may hover over a point on the plot to reveal the quantitative information behind the data point such as the date, time and number of flights around the world.


Forecasting - Baseline
AR Forecasting
10 K
8 K
6 K
4 K
2 K
05/01
06/01
07/01
08/01
09/01
10/01
11/01
12/01
01/01
02/01
AR.History + GroundTrue
AR.History + Prediction
ARIMA Forecasting
10000
8000
6000
4000
2000
05/01
06/01
07/01
08/01
09/01
10/01
11/01
12/01
01/01
02/01
ARIMA.History + GroundTrue
ARIMA.History + Prediction
Forecasting - Deep Learning
LSTM Forecasting
10000
8000
6000
4000
2000
05/01
06/01
07/01
08/01
09/01
10/01
11/01
12/01
01/01
02/01
LSTM.History + GroundTrue
LSTM.History + Prediction


LSTM-MIS-REGRESSION-I
1.25
1
0.750
0.500
0.250
0
05/01
06/01
07/01
08/01
09/01
10/01
11/01
12/01
01/01
02/01
LSTM_MIS_REG_I.lower_bound
LSTM_MIS_REG_I.upper_bound
LSTM_MIS_REG_I.y_true
LSTM-MIS-REGRESSION-II
1.50
1
0.500
0
05/01
06/01
07/01
08/01
09/01
10/01
11/01
12/01
01/01
02/01
LSTM_MIS_REG_II.lower_bound
LSTM_MIS_REG_II.upper_bound
LSTM_MIS_REG_II.y_true
LSTM-QUANTILE-REG
1.25
1
0.750
0.500
0.250
0
05/01
06/01
07/01
08/01
09/01
10/01
11/01
12/01
01/01
02/01
LSTM_QUANTILE_REG.p=0.025
LSTM_QUANTILE_REG.p=0.500
LSTM_QUANTILE_REG.p=0.975
LSTM_QUANTILE_REG.y_true
LSTM-UNIVARIATE
10 K
8 K
6 K
4 K
2 K
0
05/01
06/01
07/01
08/01
09/01
10/01
11/01
12/01
01/01
02/01
LSTM_UNIVARIATE.History + GroundTrue
LSTM_UNIVARIATE.History + Prediction



7 Solution Architecture, Performance and Evaluation

7.1 Performance Measurement

1. Our findings were evaluated by comparing our predicted results generated by the deep learning models to actual results from our dataset. We incorporated measurements such as R^2, RMSE, and MSE to ensure that our models are performing as optimally as possible.
2. We also created some figures that convey how the prediction accuracy stays within range of our confidence interval, which adds to the reliability aspect of our product.
3. We also created baseline models to compare results against the deep learning modeling results. This comparison can help us demonstrate that the deep learning models perform better.
4. Besides that, we visualized all prediction results together to give users a contrast display to identify the best methodology.

7.2 Scaling and Evaluation of Models

1. More Data

The method to adjust this within our preprocessing script is straightforward, and we also have the means to impute a large amount of missing data.The COVID data will need to be mapped to each airport's country, which can be sourced from a mapping file. The resulting structure of the data that will be ingested by the model will be the same as that at a country level, and further adjustments will need to be made to the model.

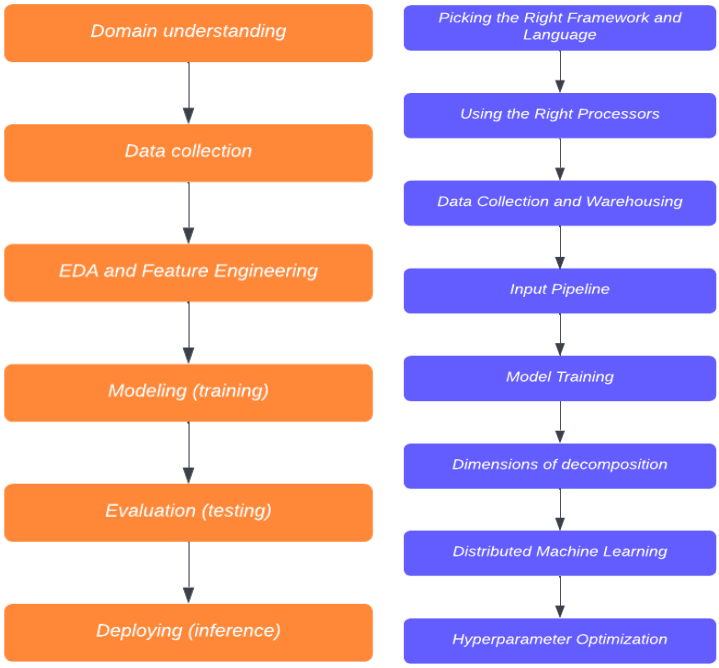
2. More Storage

InfluxDB and S3 would be able to be flexibly upgraded to hold more data; however, we would incur higher costs with S3 as more data is housed there. It will be necessary to eventually find another storage space for the raw data once our AWS allowance is reached.

3. More Processing Power

Once implemented in a Sagemaker instance, the notebook will be able to be run with a processor of our choice. Since the dataset is constantly growing as new data become available, having more processing

power available will help future-proof our product as well. We used CPUs (scalar processors) as our processor.



4. More Algorithms

We tried a bunch of training algorithms and architectures to figure out what fits our use-case the best. And we went back and forth between modeling and evaluation a few times (after tweaking the models) before getting the desired performance for a model. Also, we used loss function to tune performance. And we also track versions and history of our models.

5. Well Chosen Framework and Programming Language

We used Pytorch and Keras as our framework, and Python as our programming language.

6. Well Chosen Input Pipeline

We used Sagemaker, which is a cloud machine-learning platform to auto scaling our system. And we used Amazon S3 to easily manage data at any scale with robust access controls, and backup and restore critical data with robust application features. Also, we used Amazon EC2 Load balancers to distribute network or application traffic.

7. Evaluation

Our evaluation strategy consists of making sure that the same quality of results is achieved. We define our success as being able to match results before and after the scalability update and observing that the system is displaying statistically significant improvements in performance. We implemented each scalability improvement independently of the others to ensure that one enhancement does not negatively impact the others.

7.3 Managing our Budget

For managing our budget, our team made a plan and viewed our first billing information. Based on our first billing information, the majority of our costs are Amazon EC2, Amazon SageMaker, Amazon S3. So we took action below to manage our budget. Our overall limit is $1,000 and we consumed $347.52 up until now, with over $600 left in our account. Our average cost per day is around $10.

- Created Amazon CloudWatch alarms
    - We watched each single CloudWatch metric. And the alarm performs one or more actions based on the value of the metric or expression relative to a threshold over a number of time periods.
- Created Amazon Simple Notification Service (Amazon SNS) notifications
    - We use the Amazon SNS to simplify and reduce costs with message filtering and batching
- Used Amazon EC2 Auto Scaling
    - If a budget action is used to stop an Amazon EC2 instance in an Auto Scaling group, Amazon EC2 Auto Scaling restarts the instance, or launches new instances to replace the stopped instance. Therefore, budget actions are not effective to control cost in this use case. So we kept an eye on the Amazon EC2 periodically to make sure the Amazon EC2 instances

8 Conclusions

The results from our findings are mainly geared towards analyst and management teams at airline companies so that they may accurately predict future demand, optimize future prices, and adjust their fleets of planes as needed.

The team has formulated the following conclusions based on our findings noted above:

- Deep learning models performed quite well and provided forecast results with high accuracy.
- By comparing deep learning modeling results against classical baseline modeling results, we confirmed that the deep learning models performed as well as and better than the baseline models.
- Our deep learning results are also visualized to create benchmarks so that they can easily identify the best options to help them to do decision making.

REFERENCES

1. https://rose-stl-lab.github.io/torchTS/docs/
2. Hu, Weihua, et al. "Open graph benchmark: Datasets for machine learning on graphs." arXiv preprint arXiv:2005.00687 (2020).
3. X. Huang, G. C. Fox, S. Serebryakov, A. Mohan, P. Morkisz and D. Dutta, "Benchmarking Deep Learning for Time Series: Challenges and Directions," 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 5679-5682, doi: 10.1109/BigData47090.2019.9005496.
4. Zhang, Aston, et al. “Dive into Deep Learning.” Dive into Deep Learning - Dive into Deep Learning 0.17.2 Documentation, https://d2l.ai/index.html.

5. 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
6. Influxdb-client-python: https://docs.influxdata.com/influxdb/v2.1/api-guide/client-libraries/python
7. https://medium.com/@gohitvaranasi/how-to-build-robust-data-pipelines-in-the-big-data-ecosystem-806f84d9009f
8. https://paperswithcode.com/paper/diffusion-convolutional-recurrent-neural
9. https://github.com/Rose-STL-Lab/torchTS/

APPENDICES

A DSE MAS Knowledge Applied to the Project

The past two years of the DSE MAS program provided our team with the fundamental knowledge and skills necessary to complete this exciting and thought-fulfilling project from end to end. From extracting and analyzing data, to structuring the data pipeline, to setting up and training a deep learning model, to visualizing our results, we were fortunate to learn from dedicated professionals in the field of Data Science and look forward to applying and expanding this knowledge and skillset in our future endeavors.

- DSE 200: Python for Data Analysis
- DSE 201: Data Management Systems
- DSE 203: Data Integration and ETL
- DSE 210: Probability and Statistics for Data Science
- DSE 220: Machine Learning
- DSE 230: Scalable Data Analysis
- DSE 241: Data Visualization

B Link to the Library Archive for Reproducibility

.

https://library.ucsd.edu/dc/search?f%5Bcollection_sim%5D%5B%5D=Data+Science+%26+Engineering+Master+of+Advanced+Study+%28DSE+MAS%29+Capstone+Projects&id=bb8093534p&sort=object_create_dtsi+desc%2C+title_ssi+asc