University of Victoria

CSC 501

Report (Assignment - 2)

Submitted by:

Name	Student ID
SIDDHARTH CHADDA	V00947906
DHRUVRAJ SINGH	V00970352
ABHISHEK KATHURIA	V00959831

Introduction

At any given time in United States there are thousands of aircrafts in the sky, this makes it extremely important to monitor the skies for possible aircraft collisions, predict the probability of a bad weather event inorder to preemptively take appropriate actions like issuing alerts, redirecting flights or grounding flights as needed to protect passengers, pilots, crew members, and aircraft themselves.

The goal of our investigation is to come up with appropriate data modeling, data storage and data visualization techniques to help us analyze the large spatial-temporal flight and weather data in an efficient and scalable way. We also wish to analyze the impact of severe weather conditions on flight trajectory of aircrafts and to produce efficient techniques to monitor the possibility of aircraft collisions in real time.

Data Modelling

To facilitate our investigation, we are using the general aviation dataset provided by Carnegie Melon University. The datset itself contains two types of data: Weather data (containing various weather measurements from different weather stations) and 7 different zip files containing the flight data across 7 days.

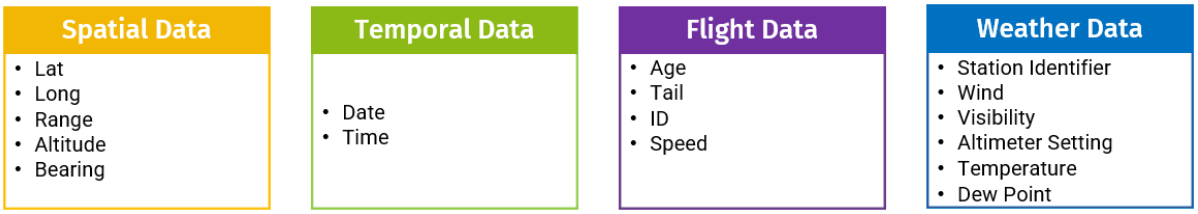
For data cleaning we have used the Pandas python library to perform basic data cleaning operations like, removing Null values from the database etc.

Conceptual Data Modelling

Preliminary Basic Data Modelling:

To produce one coherent flight and weather dataset, we first need to merge the 7 individual flight datasets into 1 dataset called flight_dataset. After producing the dataset containing the flight data for all of the 7 days, now we wish to connect or 'join' this flight dataset with the Weather dataset, this will enable us to generate some meaningful insights from the connected data. We can join the combined Flight datset with the Weather dataset by doing a Join operation the "Metar" column of the 2 datasets. Post this we will be left with 1 dataset containing the information from both the Flight dataset as well as the Weather dataset.

On further analysis we can categorize the data attributes in the following categories:



This modelling strategy has 2 advantages: (1) First it allows us to preserve the maximum amount of information from the original datasets and (2) It introduces modularity in our dataset and allows us to conduct an independent and more holistic investigation to uncover the underlying hidden relationships present in the dataset.

Spatial - Temporal Conceptual Data Modelling

Building the Conceptual Vector Data Model:

In Vector data modelling we define the features as discrete points, lines, and polygons.

Reason for Using Vector Based Spatial Modeling:

To better model the Spatial data we must first come up with a strategy to understand the aircraft spatial data in the real word 3D scenario. In the real-world scenario, each of the aircraft can be thought of as a vector of 5 spatial dimensions namely: Latitude, Longitude, Altitude, Range and Bearing.

Using these 5 spatial dimensions as radar intrinsic parameters, we can spatially imagine the aircraft in the physical world and by connecting the spatial data points of the aircraft we can plot the trajectory of the aircraft in real time and space. The below figure more vividly illustrates this point.

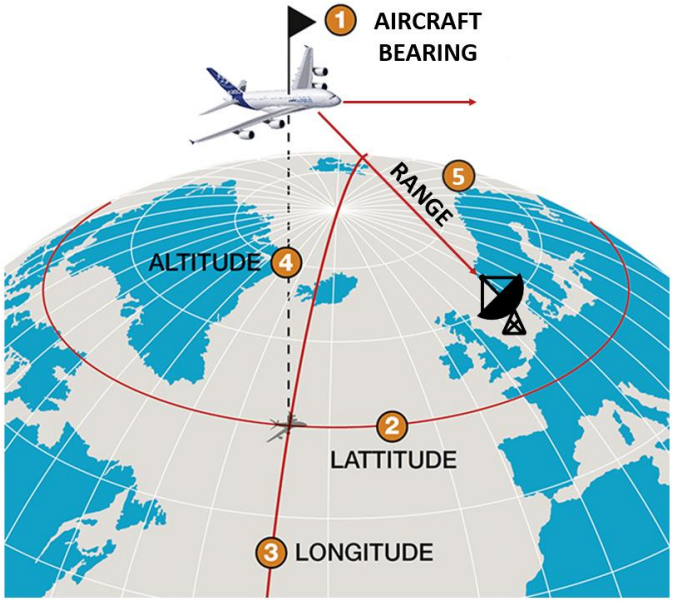


Fig.1 Representing an aircraft as a vector of 5 spatial dimensions

Using the above modelling strategy, we can now conceptually model the layers, vectors, points and arcs of our Spatial data. Given the nature of our problem we will use the Vector based technique for spatial modelling.

Reason for Not Using Raster Based Spatial Modeling:

Using Raster based spatial modelling in this case was not appropriate because Raster data model is used to represent the landscape as a rectangular matrix of square cells, whereas in our case we are just concerned with the vector dimensions of the aircraft for spatial modelling. Raster data modelling is more suitable in cases when you are dealing with an image type dataset and have to store the information in matrix type manner, this is does not apply in our case.

Defining the vectors:

I. **Points:** For points we can use the Lat, Long, Altitude, Bearing and Range as radar intrinsic paraments to represent the discreet spatial locations on the map.

II. **Arcs:** By connecting the spatial-temporal data points of an individual plane we can plot out the trajectory arcs of that plane. As illustrated in the figure below.

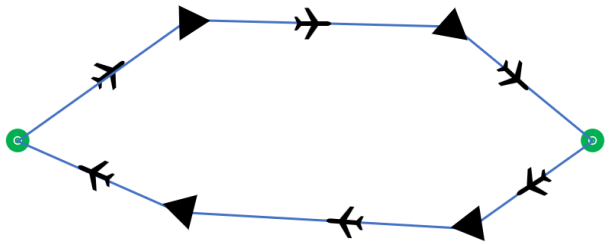


Fig.2 Forming an Arc by connecting the Aircraft spatial datapoints

Introducing Temporal Dimension in the Spatial Data Model

To build an even better model we can introduce the dimension of time to monitor the aircraft trajectory in real time. We can implement date time alongside our spatial data model and build stacked maps of the flying aircrafts. We can understand this with an analogy of pages in a book, where the book represents the complete overall spatial-temporal data model and the pages in the book represent the state of the map at that instance of time.

By adding the dimension of time, we can also connect the weather data with the plane flight trajectory data. For example, we can visualize the impact of strong wind or rainfall on the trajectory of a particular aircraft at a definite instance of time.

Logical Data Modelling

The overall purpose of our above spatial modelling was to detect the nearest neighbors of our selected aircraft and by doing so we can design a tool for preemptively detecting aircraft collisions. The data structure most suitable to implement our vision is Kdtrees.

Reason for choosing KDtrees despite the “Curse of Dimensionality” limitation:

One of the major limitations that often impact Kdtrees is the condition of “Curse of Dimensionality”. The “Curse of Dimensionality” refers to the phenomenon when arranging a large volume of data becomes extremely challenging because the dataset has too many features/dimensions.

However, in our case we have just 5 spatial dimensions, and in case of KDTrees we need more than 2^K datapoints to escape the curse of Dimensionality. That means 2^K=2^5 = 32, thus we just need more than 32 data points to escape this condition, whereas in our datset we have more than 1 million datapoints.

Hence, we can efficiently use Kdtrees to calculate the nearest neighbors to our aircraft vector in a higher dimensional space without encountering the curse of Dimensionality.

The below figure illustrates how we can use KDTree to find nearest neighbors to aircraft in Higher dimensional plane.

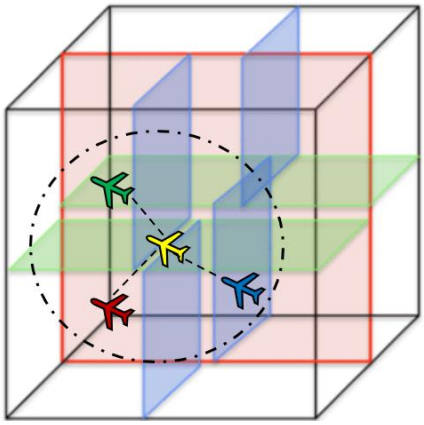


Fig.3 Using Kdtree to calculate Nearest Neighbors of the aircraft in higher dimension

Alternatives to Kdtrees considered:

The most common spatial data structures are:

I. Kdtree
II. Quad Trees
III. Rtrees

Reason for Not choosing Quad Trees or Rtrees over Kdtrees:

1. The objective of spatial data modelling is to find the nearest neighbors to our aircraft and thus preemptively detect collisions, for this task Kdtree is more suitable as compared to Quad or Rtree.
2. KDTrees have easy implementation with good library support, whereas the same cannot be said about Quad Trees or Rtrees. Good library support helps in faster prototyping.
3. Rtree partition the data into rectangles/areas whereas Kdtrees do a binary split on the data. Thus, Rtrees are better suited for holding rectangle/ polygon or area-based data, where as in our case we dealing point-based vector data and hence Kdtree becomes a better alternative to Rtree.
4. Quad trees are not suitable because quad trees split the data in all dimensions and thus when we are dealing a large datset with large dimensions this type of splitting becomes computationally expensive.
5. Searching for a data point in Kdtree is easier than as compared to Rtree or Quad tree. Thus, Nearest neighbor queries are much more efficient in Kdtrees.
6. Visualizing a Kdtree in higher dimension is much more intuitive than Quad Tree or Rtree.

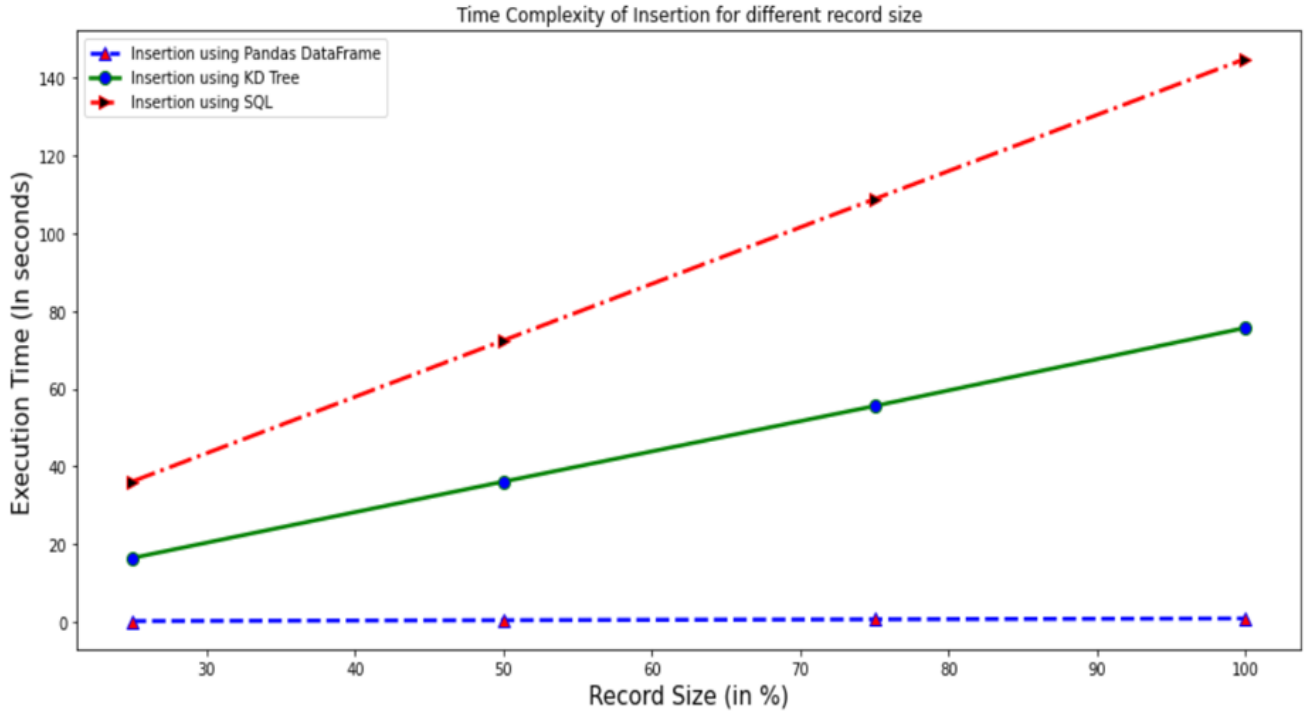
Experiments on Scalability

Experiment 1: Comparing the Execution Time for Insertion of records of different sizes using KD Trees, simple SQL Queries and Pandas based DataFrame Technique.

Insights: It can be clearly observed from the below graph that as the record size of the dataset increases, the execution time for insertion of records in the table using SQL queries (indicated in red) increases steeply as compared to the execution times for insertion of record in the table using Pandas DataFrame (indicated in blue) and KD Tree (indicated in green). Considering just the 25% of the total records, it is evident that the time for insertion using Pandas DataFrame is 0.202 seconds and for KD Trees it is 16.4 seconds whereas for SQL queries, it is 36.5 seconds which is way higher for insertion of small chunk of records in the table. We can also observe that the insertion time using Pandas DataFrame remains nearly constant for the different number of records which compliments its optimality for insertion of data.

It is also noticeable from the figure that when all the records were inserted in the table, the time taken by Pandas DataFrame was just 0.86 seconds but for the KD Trees and SQL queries, it is 75.6 and 144.8 seconds respectively.

This shows that for insertion of records in the table, Pandas DataFrame is more optimized as it uses multithreading.

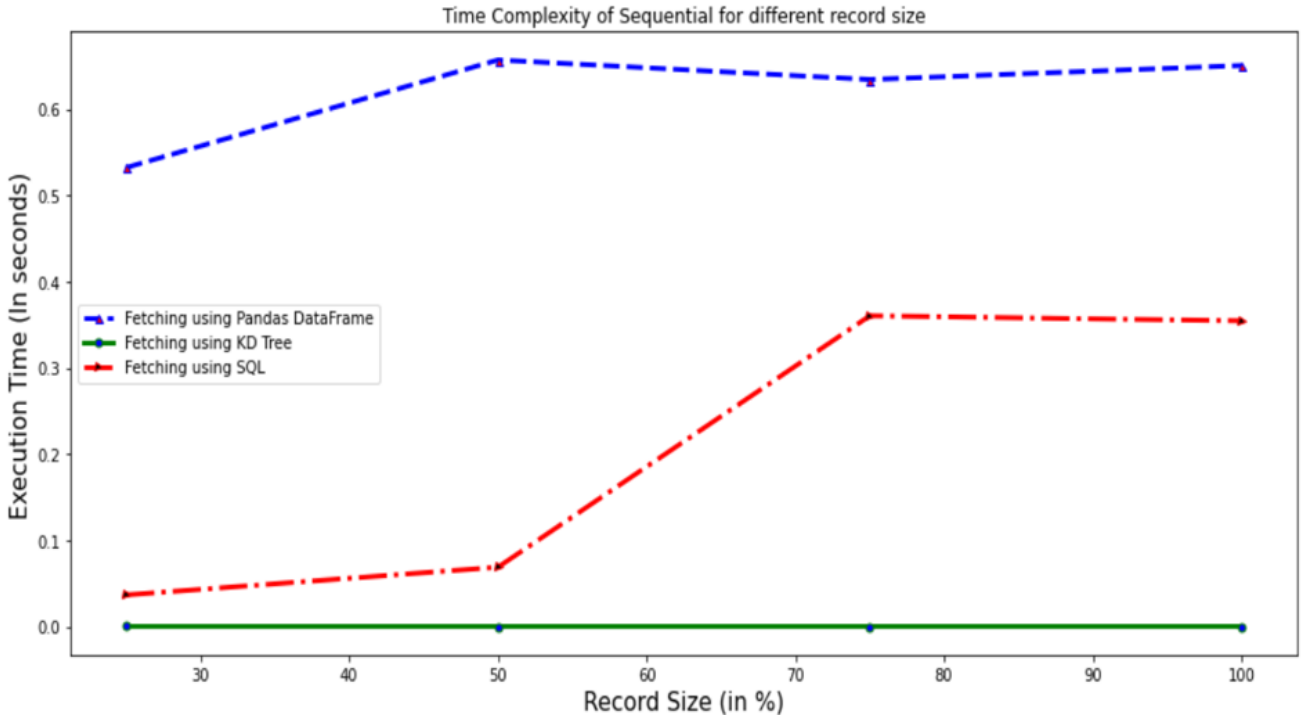


Graph. 1 Time Complexity of Insertion for different record size

Experiment 2: Comparing the Execution Time for sequential Fetching of records of different sizes using KD Trees, simple SQL Queries and Pandas based DataFrame Technique.

Insights: The below figure graph the time taken to fetch the data sequentially from local database using KD Trees, SQL queries as well as Pandas DataFrame. It can be noted that Pandas DataFrame has the highest execution time for fetching 50% of records with the value of 0.65 seconds. As the dataset size increases, the time taken fetching records using Pandas DataFrame and SQL drastically increases with the increase in volumetric data. But, on the other hand, as the percentage of the records fetched increases, the execution time for fetching using KD Trees remains almost linear and within the range of 0.06 seconds to 0.03 seconds. This graph clearly shows that although the time for fetching using SQL as well as Pandas DataFrame increases, but the slope of fetching data through SQL is much higher than Pandas DataFrame.

This shows that for fetching the records sequentially, we should prefer using K-Dimensional Trees as they are able to handle large volumetric spatial data.



Graph. 2 Time Complexity of Sequential Fetching for different record size

Execution Time for Insertion

% of Records inserted	Execution time using Pandas Dataframe	Execution time using SQL queries	Execution time using KD Trees
25%	0.22 seconds	36.16 seconds	16.45
50%	0.43 seconds	72.43 seconds	36.08
75%	0.64 seconds	100.78 seconds	55.52
100%	0.86 seconds	144.87 seconds	75.65

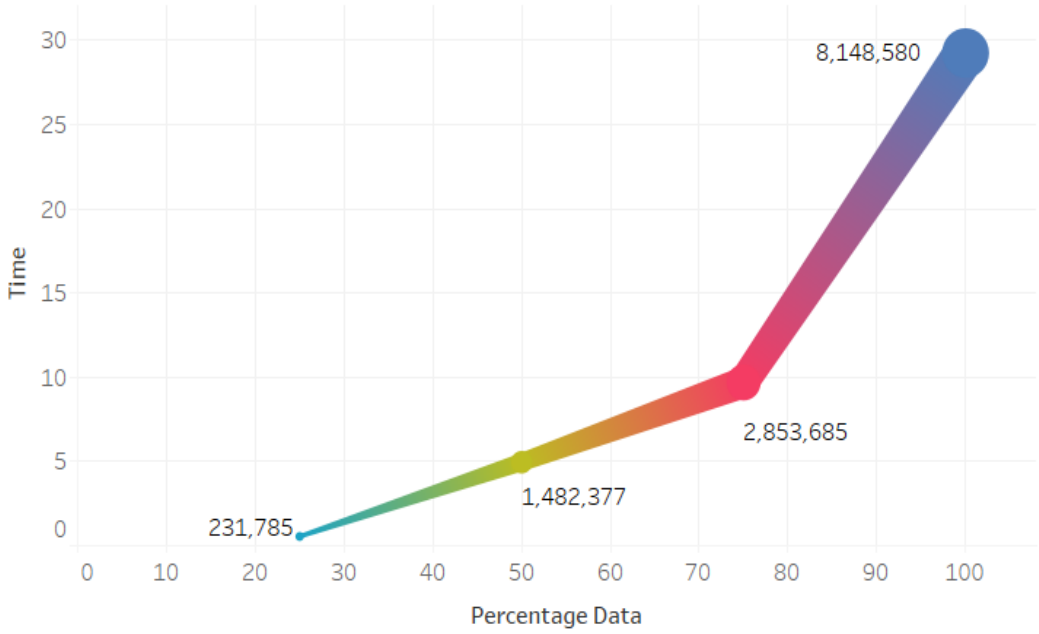
Execution Time for Fetching

% of Records fetched	Execution time using Pandas Dataframe	Execution time using SQL queries	Execution time using KD Trees
25%	0.53 seconds	0.03 seconds	0.030 seconds
50%	0.65 seconds	0.06 seconds	0.032 seconds
75%	0.63 seconds	0.36 seconds	0.034 seconds
100%	0.66 seconds	0.37 seconds	0.063 seconds

Experiment 3: Study the behavior of KD Tree algorithm for a large dataset

Insights: The below graph shows the behavior of the K-dimensional tree algorithm for a large dataset. Our goal was to study the scalability of our model by deliberately choosing a large radius value which would in turn incorporate more nearest neighbor points. Through our experiment, we found that even if we increased our radius to an extreme value and captured an exponential number of neighboring points, our algorithm was able to compute this within 30 seconds with a fast computational performance.

In the below graph, we can clearly see that as time increases, the percentage of neighboring points captured also increases steeply. Here, green represents the small percentage (25%) of the data points passed into our model which were able to capture 231785 neighboring points. Now, as the dataset size increases, the number of neighbor points also increases exponentially. We can observe this as when the dataset size was 50%, the number of neighboring points captured were 2,853,685 (indicated in red), and when 100% of the dataset was provided, we were able to capture 8,148,580 neighboring points (indicated in blue colour) in just 30 seconds.



Graph.3 Behavior of KDTree algorithm for large dataset

Experiment 4: Calculating the memory size of the Spatial data

SQL Tables used: Days Combined

Insights: In the below table, we show the comparison between the memory size of the Spatial data in csv format, Spatial Data in SQLite3 database and Spatial Data modelled using KD Trees.
The memory size for Spatial database in SQLite3 database was the least with 12.27 MB and for Spatial Data in csv, it was 80.2 Megabytes. Using K- dimensional Trees, a new database was created by first converting it into an n-dimensional Numpy array and then storing it in a tree format. Here, we used two formats for storing the KD Tree data, that is, using serialized pickled spatial data and compressed spatial data.This resulted in a compressed database of 27.3 MB and a new pickled database with a memory size of 129.1 MB which is much larger than the original database (the csv format) to store spatial data. But this is done so that it gives us an advantage of faster fetching of data from the database.

Memory Size of Spatial Data

Type of Database	Memory size
Spatial Data (In CSV format)	80.2 MB
Spatial Data in SQLite3 database	12.27 MB
Serialized/Pickled Spatial Data using KD Trees	121.1 MB
Compressed Spatial Data using KD Trees	27.3 MB

Exploratory Data Analysis

Plot 1: Flight Altitude Map

Question: How can we identify at which altitude does maximum airplanes fly near the Airport?

Insight: In Fig. 4, the green color represents high altitude and the red color depicts the low altitude at which the airplanes near the airport are flying. When the airplanes are taking off or landing, we see that the color of the lines change from red (low altitude) to green (high altitude) or vice-versa. As we can observe from the figure, most of the lines are red near the airport (represented by the blue marker) which implies that either airplanes are flying at a low altitude or the airplanes are descending to land on the airport. If we add the visibility feature to the map in Fig. 4, we get something like Fig. 5. The more clustered areas with larger sizes marked in red represent the locations with low visibility for the airplanes.

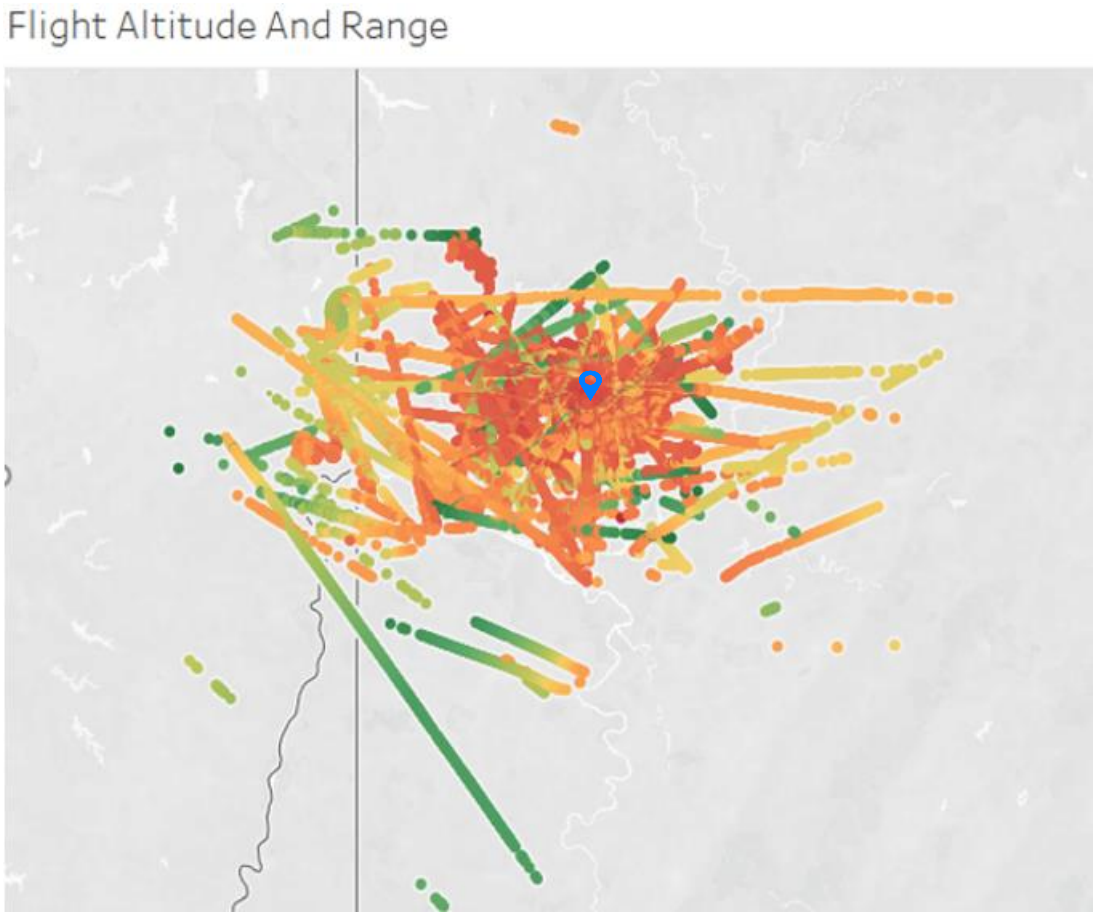


Fig. 4 Flight path visualized along with Flight Altitude

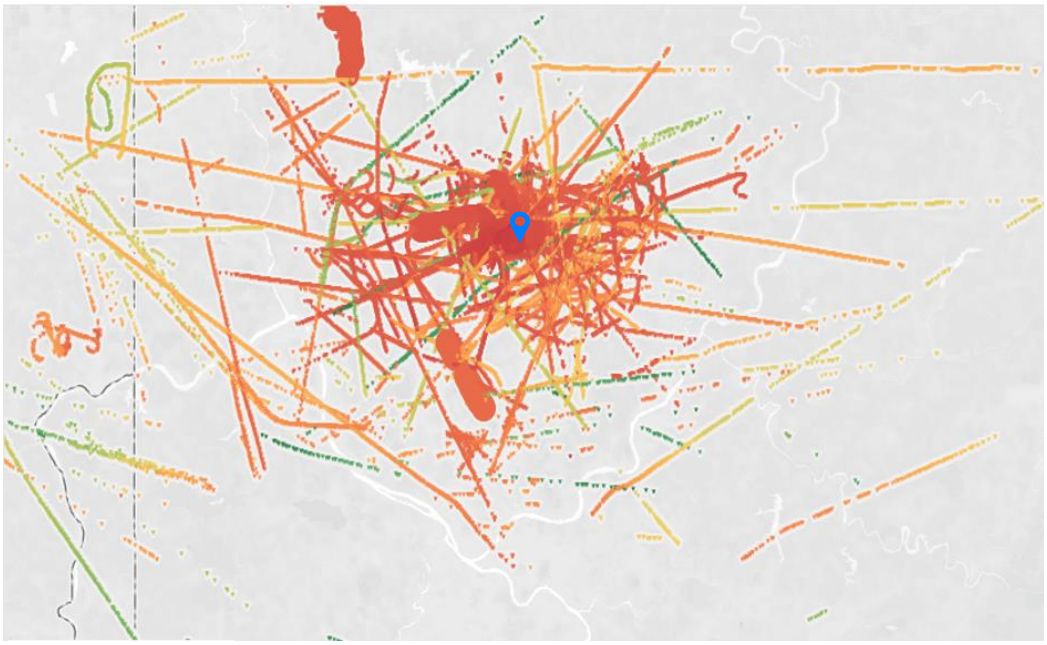


Fig. 5 Flight Altitude and Range with Visibility

Plot 2: Altitude Vs Flight Speed

Question: What is the correlation between Flight speed and Altitude? Also, does an aircraft being within range of the airport affect its flight speed?

Insight: As we can observe from Fig. 6, the speed of the aircraft increases with the altitude. When airplanes take off and start ascending, their speed increases as well. The color range from red to green shows the altitude from low to high. Furthermore, we can also see that when the airplane is within the range of the airport, its speed starts decreasing. The size of the boxes represents the range of the airport. Small boxes show that the range between the airport and the airplane is small i.e. the airplane is about to reach the airport or the airplane has just left the airport and so its speed is still low. As the range increases (size of the boxes increases), the speed increases as well.

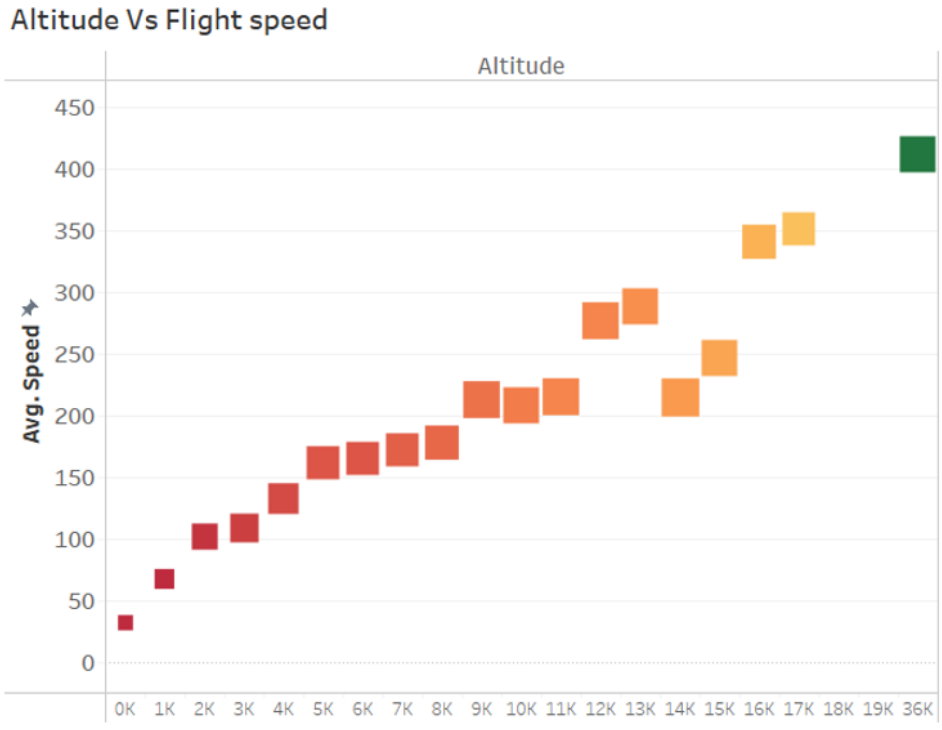


Fig. 6 Altitude Vs Flight Speed

Plot 3: Rainfall Vs Visibility

Question: How does rainfall and wind speed affect the visibility of the airplanes at different hours of the day?

Insight: In Fig. 7, we can see the correlation between the Avg. Precipitation (Rainfall) on the y-axis and Hours of the day on the x-axis. The color range from red to green represents visibility from low to high respectively and the size of the circles represents the wind speed. We can see that during times when Avg. Precipitation is high, the visibility is low. Also, during the hours when sunlight is low and rainfall is high (i.e. Dawn at 5 and 5 a.m.), the visibility is very low. From hours 15 to 20, we can see that even though there is some rainfall, the visibility is high (green color). This can be due to high wind speed which sweeps the rainfall in some particular direction which is represented by larger circles.

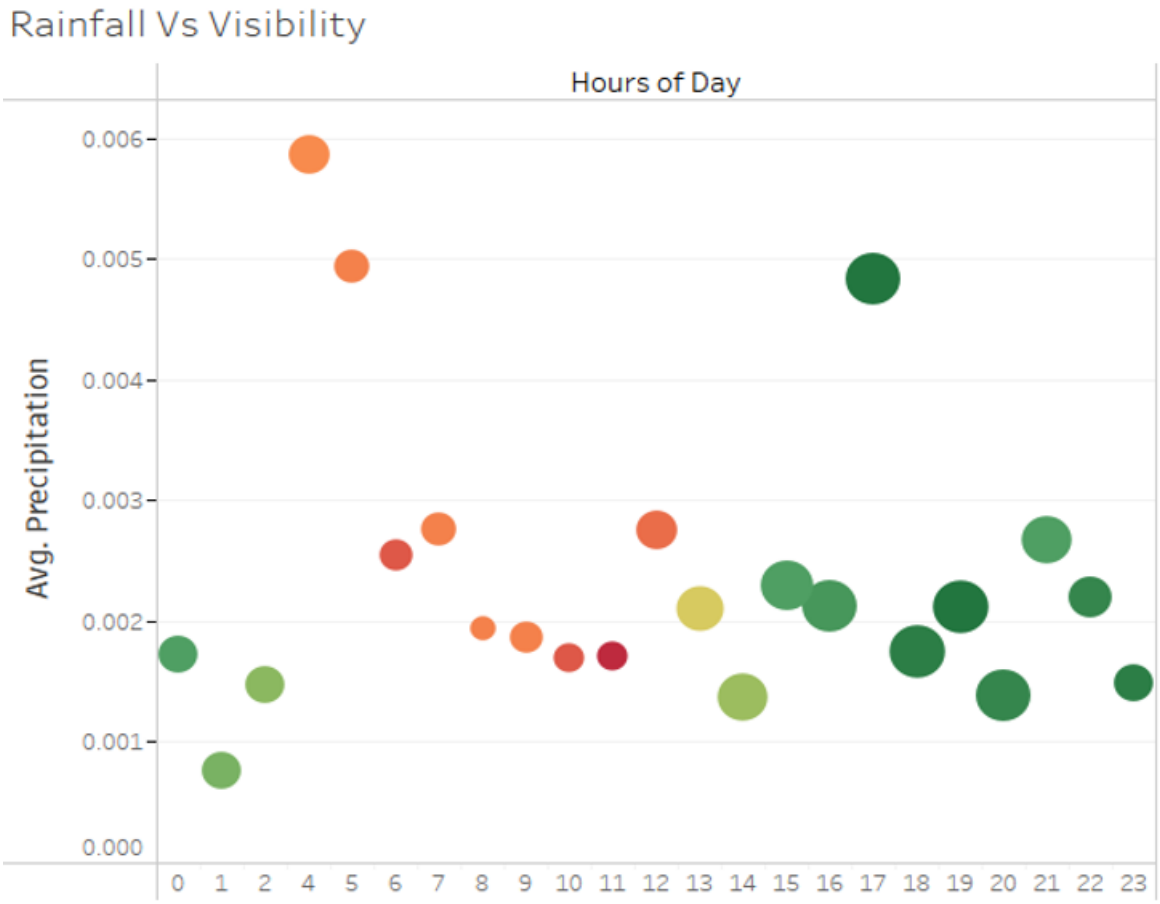


Fig. 7 Rainfall Vs Visibility

Plot 4: Hours of the Day Vs Avg. Temperature

Question: What is the correlation between the Avg. Temperature and Hours of the Day?

Insight: The Line graph in Fig. 8 represents the correlation between Temperature and Hours of the Day. From hours 1 to 11, the temperature is 37 to 43 Fahrenheit but the temperature starts to rise from around 12 to 18 hours. Then again, we observe the fall in temperature till the 23rd hour. That is obvious since the temperature usually drops during the nighttime. The interesting thing is that the relative humidity is more during the dawn hours (1 to 11) rather than the noon hours.

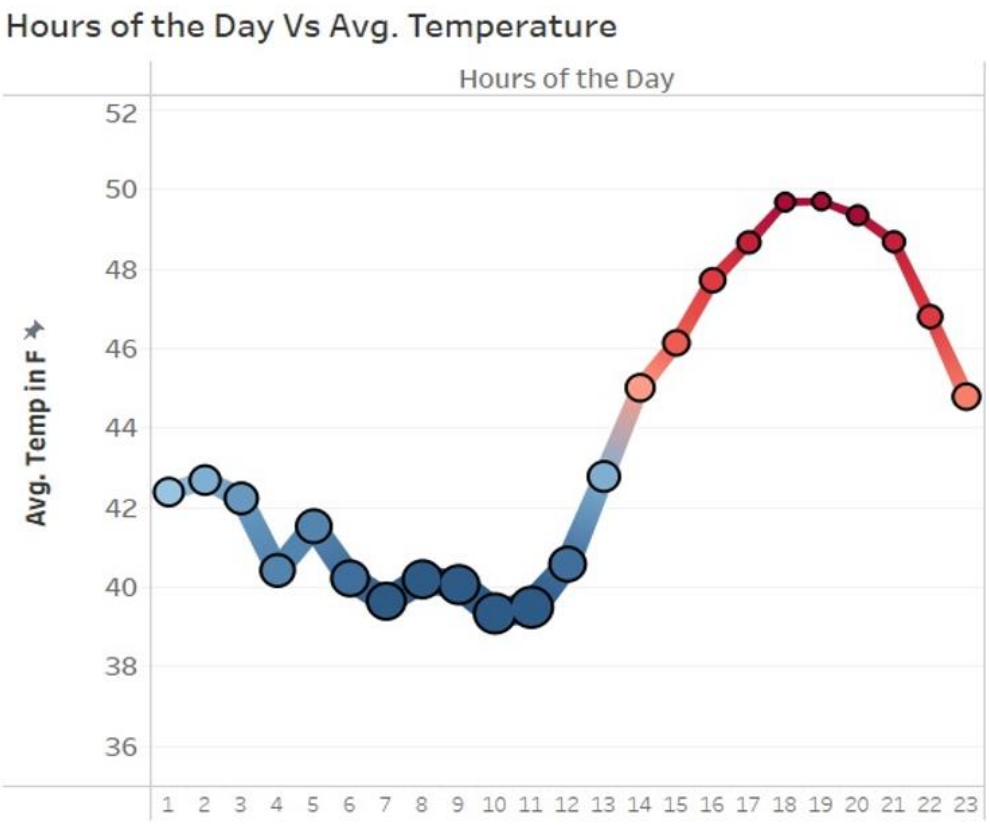


Fig. 8 Hours of the Day Vs Temperature