**Problem Introduction**

To properly establish what pieces of data we can capture, I would look at what the customer journey looks like, from inspiration to flight booking to the actual day of travel. This will shed light on what kind of events or data will be useful to analyze customer behavior, identify any errors, and provide insight on future purchases.

**User Journey**

1. Inspiration for flight
2. Website landing page
3. Inputting flight information
4. Search Results
5. Booking
6. Travel

**Available Data**

1. Inspiration for flight
   1. Impression data from marketing campaigns (email, online ad, etc…)
2. Website landing page
   1. Choose extra options, which has upgrade
   2. Time spent on page
   3. User sign in -> member information
3. Inputting flight information
   1. Flight itinerary information
4. Search Results
   1. Flight availability
   2. Click events
      1. Click details on which flights
      2. What filters did the customer use
      3. Did the customer change search criteria
      4. Clicked compare fare type, and which flights did they compare
   3. Time spent on page
5. Booking
   1. Passenger information
   2. Travel insurance
   3. Gift card
   4. Payment information
6. Travel
   1. Check-in details
      1. Self-serve kiosk
      2. Online check-in
      3. Booth check-in
   2. Questions/Concerns (seating, upgrade request, etc…)

**Customer Interaction Questions**

1. **What is the conversion rate for an individual flight search result? For definition, an ‘individual flight search result’ is the following:**

Using the clickstream data that we’ve gathered for the session, we can determine if the customer ended up booking a flight. The event indicator would be if they clicked the Purchase button

1. **How does the cost in miles affect the guests’ willingness to purchase? For this question, assume that the conversion space is already tagged successfully, and this data is already accessible.**

Initially for a naïve approach, you can view the percentage of conversions for bin ranges of different mile costs.

For a more refined approach, if you consider mile cost as a feature in a model (classification fits for conversion), you can look at factor importance. Lower factor importance would mean that cost in miles wasn’t a big motivator for a customer to end up booking.

1. **How many times does a guest attempt to search for a destination that we do not serve for the given airport or date? This can be demoed through the following flight search:**

This can be obtained easily from the session click data. There will be some call to action to show the “not available” screen, and this can be captured as part of the data model. We would query for a count of this type of event per session.

**Data Storage**

**Storage on Cloud**

Following an Azure cloud architecture, Azure SQL Database is a great option to store the tagging/event data from users. The service allows loading data that follows a JSON schema. It also integrate well with Databricks, which can clean and prep the data for analysis from Blob, and load it back into SQL Database.

**Data Warehouse**

For data warehousing, the first option I propose is Azure SQL Data Warehouse as part of the Azure infrastructure. You can use Azure SQL DB as a warehouse, but it’s not optimized for analytics. It performs better with CRUD operations such as many INSERT, UPDATES, and DELETES. However, SQL DW does have its limitations. It doesn’t allow cross database queries and the query times and costs won’t be competitive with other options.

The second warehousing solution I would suggest is Snowflake. Snowflake was built and optimized for the cloud – this means better compression leading to faster query times, incredibly resilient data fail-safes if any data was changed or deleted, and reduced cost from compute. Data cloning and sharing capabilities allow for easy control on what is shared. There are also considerable cost savings, as you pay almost nothing for storage, just compute. One downside to straying away from a complete cloud-based solution is that you’ll have to connect Snowflake to Azure, but Snowflake has useful tools that can make data ingestion very easy, such as Snowpipe. Snowpipe allows continuous data inflow from Azure blob storage using an external stage. Another consideration is the initial setup and account management needed for Snowflake. Whereas in Azure, the IAM provisioning might already exist for the other services, it will take some time and resources getting Snowflake ready for use by cross-functional teams.

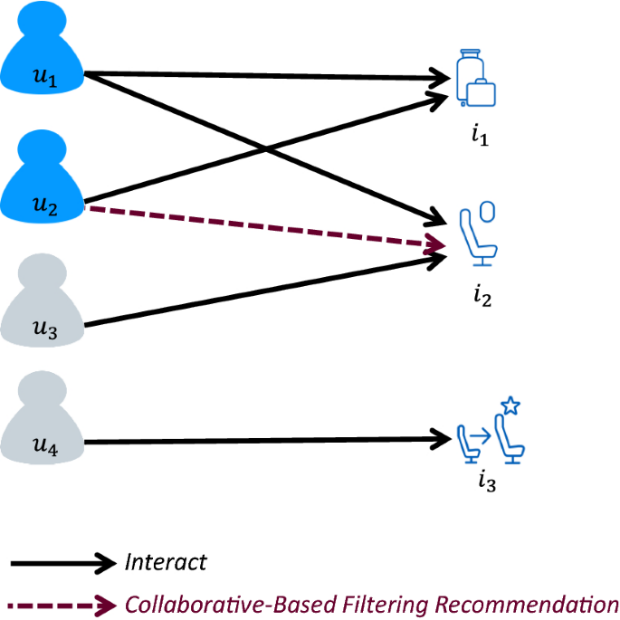
**Modeling**

If Azure SQL Data Warehouse is chosen as the warehousing solution, I would use Databricks as the main ML application of choice. Since all the data sits in Azure, we can use PySpark to both query and model the data.

If choosing Snowflake as the warehousing solution, there are many options you can select for your analysis. As mentioned before, Snowflake is optimized for cloud computing, so it has many useful connectors for the popular analytical tools. You can use Databricks using a Spark connector, a Python notebook using a Python connector, or even use Java using a JDBC connector.

A recommendation system is a great solution for maximizing revenue with ancillary additions such as upgrade seating, seat selection, baggage options, insurance, etc… By using the customer profile to identify what they’ve purchased in the past, and matching those preferences with other similar users, we can recommend extra options to them besides the main flight ticket. However, the main issue with recommender systems is that of the cold start problem. If there isn’t enough past information for a user to build a profile for them, the resulting predictions can be inaccurate. Customers with AlaskaAir accounts will most likely have previous flight profiles that we can build a traditional recommender system with.

I would recommend a collaborative filtering approach to this problem. In my opinion, choosing an add-on is a binary decision. Although add-ons can be similar to each other and so similar items can be recommended, I think we can extract better results from using user choices rather than item similarity. We can always model for both and compare results.



I would start off with a nearest neighbors and a matrix factorization approach to compute the similarity between users and items. One thing to be aware of first is that collaborative filtering works best with rated items. Currently, we only have enough information to provide a binary of whether the item was purchased or not. I would propose creating a heuristic of some sort to rate the items. We can leverage the data that we’ve captured from the user journey, such as number of times that they bought an item, how quickly they selected it, and what other similar items did they buy to come up with an item rating.

In the nearest neighbors approach, each user’s choices would be represented as a vector. The cosine similarity would be computed between the user vecor and all other user vectors, and the closest x neighbors who are most similar would be recommended for that user.

In the matrix factorization approach, it will try to identify the latent factors that exist between users and add-on item preferences. We will have to test how many latent variables would best explain the user/item interactions, and reduce our dimensionality accordingly. We can get the customer’s preference for an item by taking the dot product of the user-feature and feature-item matrices and recommend items by order of preference