**Overview:**

Makemytrip users searching for the flights make activity in the app to look for the flights and book them. This activity can be across multiple search contexts, search context is defined as combination of from\_city:to\_city:travel\_date. User might be searching for multiple dates and destinations. We have this activity data and would like to identify the search context user would finally book for.

**Problem statement:**

For a sampled point in time of a user, you are given the data of the user activity in the past 7 days and the activity in next 7 days. You need to build a model that scores each of the search context user made in the past 7 days. This score would be used to rank the search contexts and high scored search context should have higher booking chances in the next 7 days compared to the low scored one.

**Data:**

File type: parquet

Each line item is a time in point sample of a user:

{

“context”: <list of user activity in past 7 days>,

“affinity”: <list of user activity in next 7 days>

**}**

Each activity by the user looks like this:

{

“features”: array of size 24,

“from\_id”: <id of the source airport>,

“to\_id”: <id of the destination airport>

}

Array feature definition:

Index 0-5: (lat, lon, area) from and to city features

Index 6: travel date in integer format (days relative to 2023-01-01)

Index 7-9: travel date features day of the week, day of the month, day of the year

Index 10: search timestamp in integer format (days relative to 2023-01-01)

Index 11-14: other search related fields like travel class, pax etc

Index 15: Activity type

0: user viewed listing page for search context

1: user went a step ahead and looked for a flight

2: user went till the add traveler details page

3: user made a booking

Index 16-23: price related features

\*\*To get the search context of an activity, you may concatenate from\_id:to\_id:features[6]

**Analyzing the Problem Statement**:

Source: “You need to build a model that scores each of the search context user made in the past 7 days.”

Learning: The training data for building the model would be the activities in last 7 days.  
If needed a validation split, then it should only be created from within this last 7 days activities.

The activities of the next 7 days, would be used as a Test set to get an approximation of “Best Model’s Performance on Unseen Data”. This test set will be used only once.

Source: “high scored search context should have higher booking chances in the next 7 days compared to the low scored one.”

Learning: The Activity type column has 4 values provided: [0, 1, 2, 3].

3 refers to a booking.

The problem statement could be formulated in 2 ways:

Should we try to learn a Ranking Model, that scores 0 class obs < 1 class obs < 2 class obs < 3 class obs ??

In this approach, we could have more info about the class in which the search context is falling and interventions could be made to move the search context to the next higher class

Another option is to learn a Binary Class Model that scores whether an observation belongs to class 3 or not ?

Here the main goal would be to separate the search context that will complete the booking from all other search contexts, which would not go till the final stage.

Since the problem statement specifically mentions --> “high score --> higher booking chances”, we’ll formuated this as a Binary class problem with ACTIVITY TYPE == 3 as class 1 and all others as class 0.

**Measuring Progress**

For the different models, that we’ll build, there are multiple options to measure if we are improving or not.

More specifically, we can choose to optimise either Precision or Recall.

In this problem statement, looking at the business context, we can define the metrics as below:

**Precision**: Of all the search context that my model identified as “BOOKING”, which ones actually booked the flight.

**Recall**: Of all the search context which are actually going to “BOOKING”, which ones my model can capture correctly.

In this scenario, it makes more sense to tune the models, to achieve **High Recall**.

The justification can be something like this: If we can maximise the amount of actual BOOKINGS that we can capture, then there could be some business opportunity that can be explored --> Like maybe offering additional services such as complementary meals, extra leg space seats other, that we can offer to those search contexts.

**Caution**:

Since we have decided that we want to achieve high recall, a naive way of achieveing this is keeping lowest possible threshold = 0.0

This would mean that we effectively classify all search context as booking and so we’ll capture all actual BOOKINGS too.

However, there would be no intelligence in this model and it’ll have very high False Positives.

So, to make sure that our model is learning something, we’ll monitor **F1 score along with Recall** to make sure that there is actual learning going on.

**PROJECT CODE REPO**: <https://github.com/suka1557/makemytrip_assignment>

**EDA**

the data has been analysed and the notebook with findings has been added to the github repo

**Modelling Approach:**

We tested 2 types of models:

Classical Models - Logistic Regression, Decision Trees, Random Forest, LightGBM and XGBoost.

FeedForward Neural Networks with learning embeddings for FROM-TO-CITY pairs

**Class Imbalance**:

In the data it is seen that BOOKING:NON-BOOKING is roughly in the ratio of 14:86.

To handle this issue, while training the model, we used the following approach:

- Undersample Majority Class for classical models

- Use Class Weights for Neural Nets

**Performance Metrics**:

The best model was a Decision Tree based model which achieved ~66% Recall on test set.

**Inferencing**

The inferencing was added in 2 ways:

* A static script that requires location of a parquet file, in the exact same format as the assignment.parquet. This function will take in this file location, read , process and then make predictions and return a 1-D array of True/False based on optimal threshold found during the experiements.
* A FastAPI endpoint that requires the following inputs:
* **FROM\_CITY** : int
* **TO\_CITY**: int
* **features**: json payload - list of 23 floating point numbers (excluding the value for ACTIVITY\_TYPE)
* **Response**: Predicted probability of Booking for this record

**Deploying API**

* A bash script has been created to run the API using gunicorn on PORT 9001
* A Dockerfile has been created which will created image and run this script inside the container
* A Github Actions Continuous Integration pipeline has been added, so that anytime new changes are pushed in main branch, it will create the new image
* Final deployement is not done, as currently don’t have any actual Vms/Resources

**Improvements in Current Work**

* Lack of computing resources meant that not other NN architectures could be tried out. Even for the ones, which are in the repo, has to use Google Colab to do the model fitting. More complex networks could be tried to see, if we can improve performance given that we’d not fallen into OverFitting range with current architectures
* Integrating MLFlow with experiments for easy tracking of all hyperparameters and performances
* No Error handling has been added in the code as of yet. There could be scenarios for bad/incorrect API inputs
* The final deployement and CD pipeline for the API has not been configured. They could be worked upon

**Regards,**

**Sukant Kumar**

**9599671519**