Custom Chatbot - Restaurant Recommendations

The goal of the custom chatbot is to assist the user to identify the top 3 restaurants within Bangalore based on the following factors:

- Cuisine
- Dietary Restrictions
- Preferred location
- Current location
- Meal type (Occasion)

To build this custom chatbot for recommending restaurants the following steps are carried

1. Collecting the relevant datasets - Finding the appropriate set of dataset to recommend the
2. Data exploration of the chosen dataset
3. System design chatbot
4. Evaluation metrics - user preference scores vs recommendation results reliability

Dataset source

Dataset source: Bengaluru Restaurants

The basic level EDA was performed to verify if the data is suitable for the objective of building the chatbot.

The list of columns in the data:

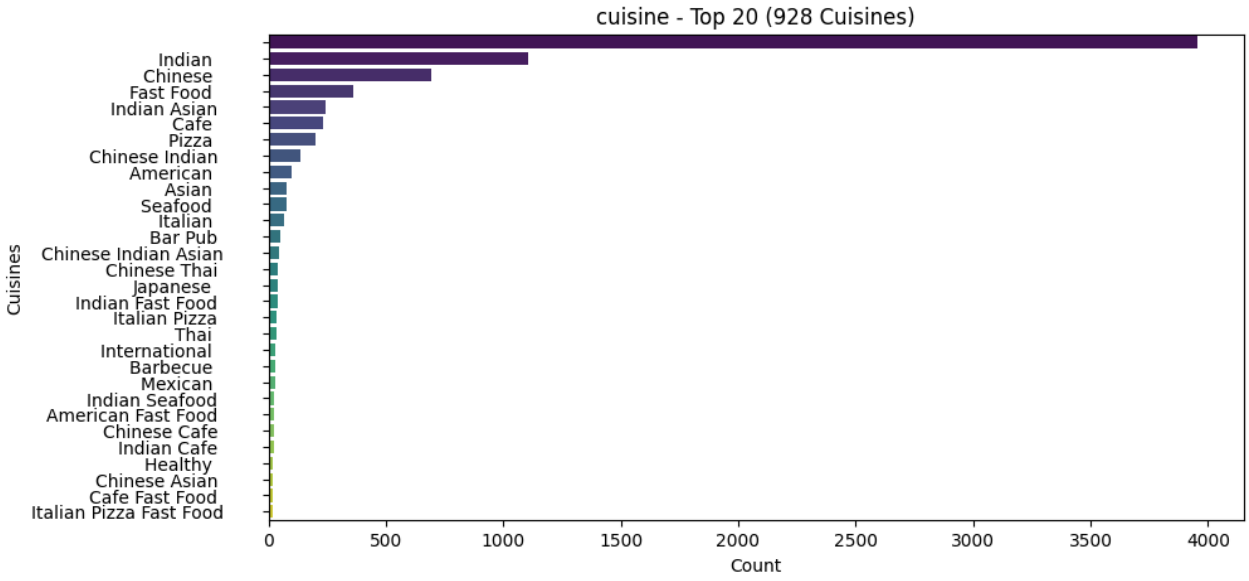
```
List of columns in the dataset:
- name - <class 'str'>
- address - <class 'str'>
- addressObj/country - <class 'str'>
- addressObj/postalcode - <class 'str'>
- addressObj/state - <class 'str'>
- cuisine - <class 'str'>
- description - <class 'str'>
- DietaryRestrictions - <class 'str'>
- Dishes - <class 'str'>
- Features - <class 'str'>
- latitude - <class 'str'>
- longitude - <class 'str'>
- localAddress - <class 'str'>
- MealType - <class 'str'>
- numberOfReviews - <class 'str'>
- phone - <class 'str'>
- rankingDenominator - <class 'str'>
- rankingPosition - <class 'str'>
- rating - <class 'str'>
- rawRanking - <class 'str'>
```

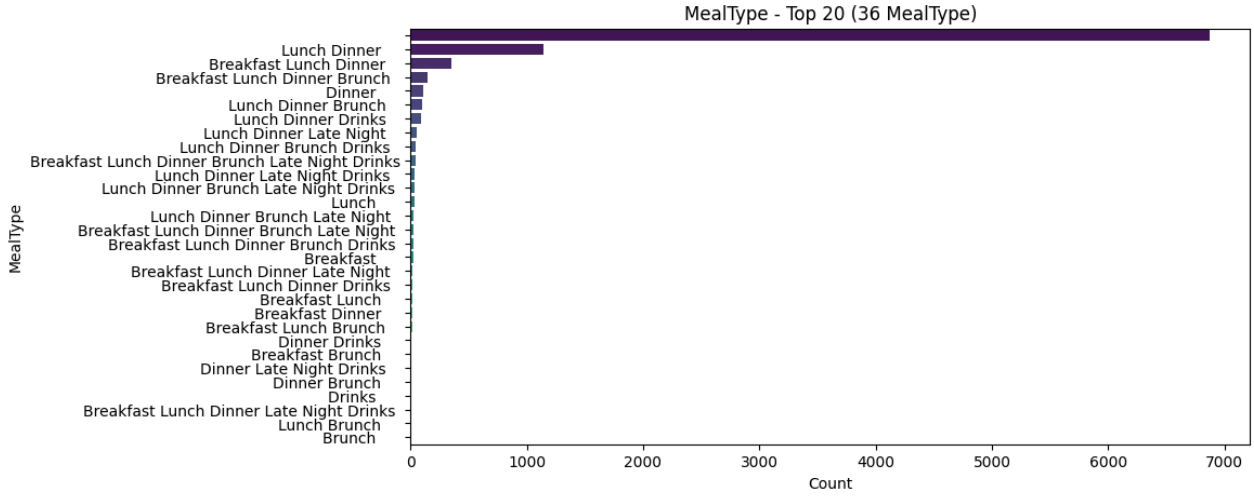
Data Exploration

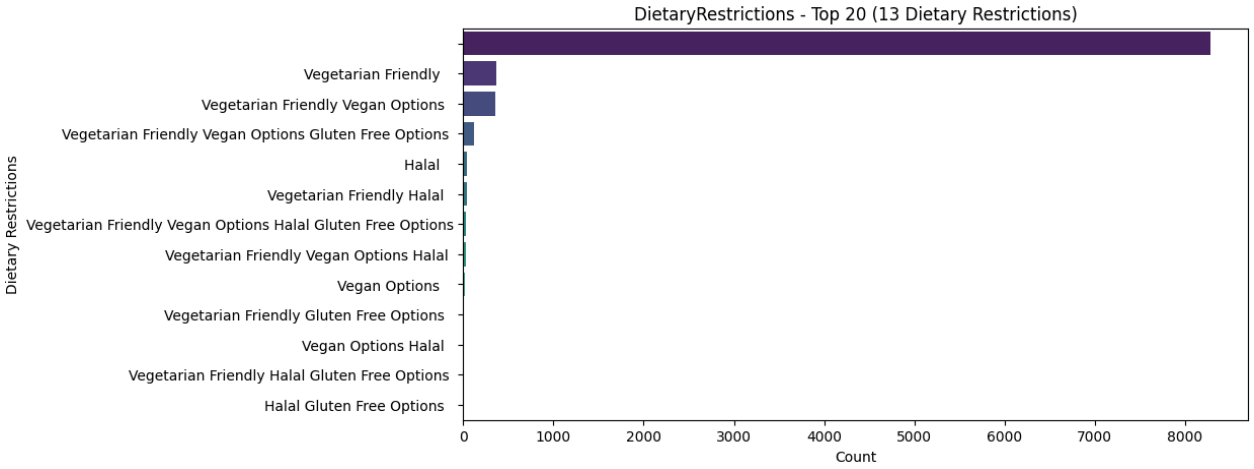
The columns of interest to design the chatbot recommendation:

```
Missing data evaluation:
name                    0.000000
cuisine                 0.000000
DietaryRestrictions     0.000000
localAddress            0.000000
latitude                3.541061
longitude               3.541061
numberOfReviews         0.000000
rating                 41.007427
rankingDenominator     40.856743
rankingPosition        40.856743
rawRanking             40.856743
dtype: float64
```

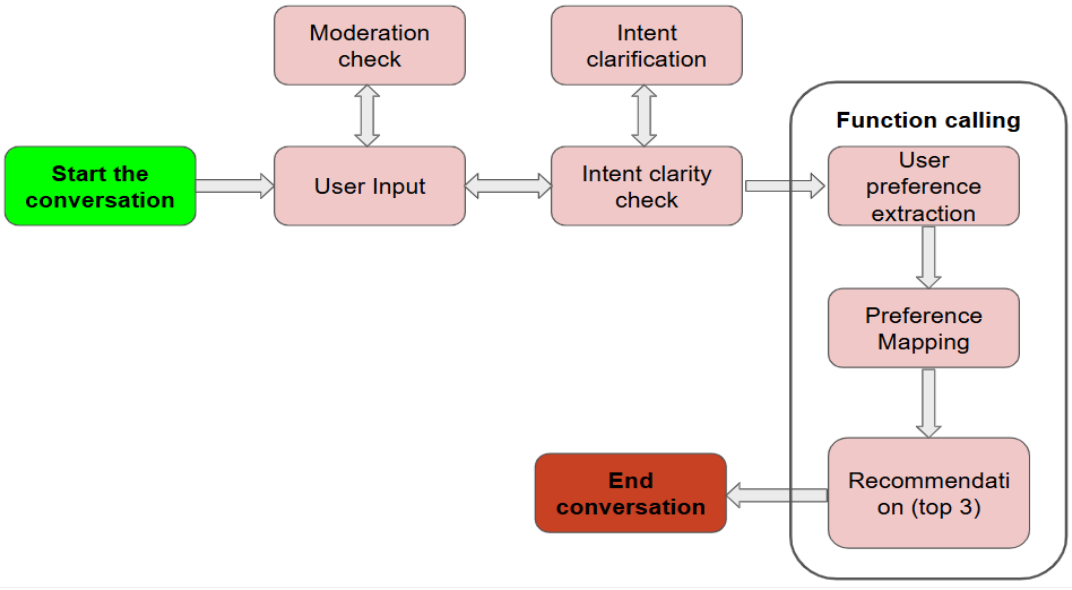
The missing values are not a hindrance for building a chatbot with GenAI as we have several restaurants in the chosen dataset with various cuisines. All the required columns to base the chatbot are very populated with less than 50% missing values.







High-level overview - System Design



Technical Implementations

Stage 1:

- Initialize the conversation by setting up the role, content etc.
- Invoke multi-turn conversation by appending the history of conversation to follow up on the intent of the user to assistant and avoid repetitive questions until the chosen categories are answered by the user
- Set the context by few-shot prompting and chain of thoughts
- Define chat completion function with function_calling argument turned on the “auto” mode
- Implement moderation check to avoid offensive questions

Stage 2:

- Define the product mapping function - flag and filter restaurants that match the user’s preference. Calculate the user preference score against each restaurants in the dataset
- Define the recommendation function that recommends the top 3 restaurants based on the highest user preference score and highest rating
- Define a function calling tool - To plug these functions for final recommendation task into the OpenAI response module

Step 3:

- Validating the output by asking out of the box questions: For instance, suggest some restaurants near Anna nagar chennai. The bot asks to rephrase and commits its limitations
- Stress testing the chatbot to validate the consistency of the response
- Human-like conservation sanity check

Challenges

- Choosing the right set of data with various cuisine options was quite challenging as there were multiple datasets with inconsistent variables to base the chatbot on.
- Intent confirmation layer and initialization layers were redundant with several prompting techniques and an increased set of tokens. Introduction of function calling reduced the number of tokens involved with systematic tooling process
- Experienced inconsistent responses for the same set of preferences. Standardized them using temperature and seed arguments in the chat completion module