# 1 Libraries and Their Usage

To implement our restaurant recommendation system, we will utilize several Python libraries that provide essential functionality for data processing, graph-based ranking, and visualization. Below, we outline the key libraries we intend to use, their role in our project, and why they are appropriate.

## 1.1 pandas (Data Processing)

**Usage**:

- Load, clean, and manipulate restaurant data.
- Filter restaurants based on user preferences (e.g., cuisine type, price range, and minimum rating).
- Store intermediate results such as PageRank scores and MST relationships.

**Key Functions**:

- `pandas.read_csv()`: Load the dataset.
- `pandas.DataFrame.query()`: Filter restaurants based on user inputs.
- `pandas.DataFrame.sort_values()`: Sort recommendations by ranking score.

**Why It's Appropriate?** `pandas` provides powerful tools for tabular data manipulation, making it easy to filter and process restaurant data efficiently.

## 1.2 networkx (Graph Construction & Ranking Computation)

**Usage**:

- **Manually construct a graph representation of restaurants**, where:
  - **Nodes** represent individual restaurants.
  - **Edges** are added based on **computed similarity scores** between restaurants (e.g., based on cuisine type, rating, and pricing).
- Implement a **custom similarity function** to define edge weights.
- Apply **PageRank** on the constructed graph to determine restaurant ranking.
- Compute **Minimum Spanning Tree (MST) on the similarity graph** to refine recommendation consistency.

**Key Functions**:

- `networkx.Graph()`: Stores the manually constructed graph.
- `networkx.pagerank()`: Runs PageRank on our graph, **which we construct manually**.
- `networkx.minimum_spanning_tree()`: Helps extract a refined recommendation set based on similarity connectivity.

**Why It's Appropriate?** `networkx` provides an **efficient way to store and manipulate graphs**, but we **define the graph structure, edge weights, and similarity functions** ourselves. This ensures that our ranking and MST results are tailored to the problem rather than relying on pre-built generic implementations.

## 1.3 plotly (Visualization)

**Usage**:

- Display restaurant rankings using bar charts.
- Provide interactive visualizations of the ranking results.

**Key Functions**:

- `plotly.express.bar()`: Generate a bar chart to display PageRank scores.
- `plotly.graph_objects.Figure.update_layout()`: Customize visualization layout.

**Why It's Appropriate?** `plotly` allows us to create **interactive and visually appealing** ranking displays, helping users interpret the recommendation results.

## 1.4 (Optional) streamlit (Interactive UI)

**Usage**:

- Create a **simple web interface** where users can input preferences (e.g., cuisine type, budget, and rating).
- Display **real-time recommendation updates** based on user selections.

**Key Functions**:

- `streamlit.selectbox()`: Allow users to choose cuisine type.
- `streamlit.slider()`: Let users set budget and rating preferences.
- `streamlit.dataframe()`: Display recommendations in a table.

**Why It's Appropriate?** `streamlit` simplifies the process of **creating interactive interfaces** without requiring extensive front-end development.

### 1.5 Conclusion

Each of these libraries plays a crucial role in our project:

- `pandas` for efficient data filtering and processing.
- `networkx` for graph-based restaurant ranking and MST refinement.
- `plotly` for clear and interactive visualization of ranking results.
- **(Optional)** `streamlit` for an interactive user interface.

This combination ensures an efficient, scalable, and user-friendly recommendation system.