Milestone 2 Report: Feature Engineering, Selection, and Data Modeling

Course: CAP5771 - Spring 2025
Name: Rushang Sunil Chiplunkar
Date: April 3, 2025

Introduction and Project Overview

Project Objective:
The primary objective of this project is to develop a recommendation tool for restaurants in Bangalore. This involves aggregating and analyzing diverse datasets from Swiggy, Zomato, and a general Indian Restaurants dataset. The goal is to provide users with comprehensive insights and personalized recommendations based on preferences such as cuisine, location, and cost, addressing the current lack of a consolidated platform. The final tool will be an interactive dashboard built with Streamlit.

Tool Type:
This project is focused on developing a restaurant recommendation engine that aggregates data from multiple sources and delivers personalized recommendations. The final deliverable is an interactive dashboard built with Streamlit, which enables users to filter recommendations based on criteria such as location, cuisine preferences, and budget.

Data to Be Used:
The project utilizes three datasets sourced from Kaggle:
• Swiggy Restaurants Dataset: Swiggy Restaurants on Kaggle
• Zomato Bangalore Dataset: Zomato Bangalore on Kaggle
• Indian Restaurants Dataset: Indian Restaurants 2023 on Kaggle
These datasets provide comprehensive details on restaurant names, addresses, ratings, costs, and cuisines, which are essential for constructing my recommendation system.

Tech Stack:
The project is built using Python and uses a suite of libraries:
• Pandas and NumPy for data manipulation and preprocessing
• Scikit-Learn for feature engineering, model training, and evaluation
• Matplotlib and Seaborn for data visualization

• Streamlit for developing an interactive and user-friendly dashboard

Milestone 1: Data Collection, Preprocessing, and EDA

- **Feb 5, 2025 – Feb 21, 2025:**
  • Data collection from Kaggle sources and initial data extraction (notebooks 1_data_extraction.ipynb and 2_data_processing.ipynb).
  • Data cleaning, standardization, and exploratory analysis (documented in 3_data_exploration.ipynb).

Milestone 2: Feature Engineering, Feature Selection, and Data Modeling

- **Feb 22, 2025 – Mar 28, 2025:**
  • **Feb 22 – Mar 1, 2025:** Feature Engineering
  - Enrich data via the Google Places API and create new features (content_features and popularity_score) as detailed in 4_feature_engineering.ipynb.
  • **Mar 2 – Mar 8, 2025:** Feature Selection
  - Evaluate and justify feature inclusion/exclusion; implement implicit dimensionality reduction (using TF-IDF limits and SVD) as shown in 5_data_merging.ipynb.
  • **Mar 9 – Mar 28, 2025:** Data Modeling
  - Train and compare at least three recommendation approaches (content-based, collaborative filtering, and a hybrid model) with detailed evaluation (documented in 6_model.py).

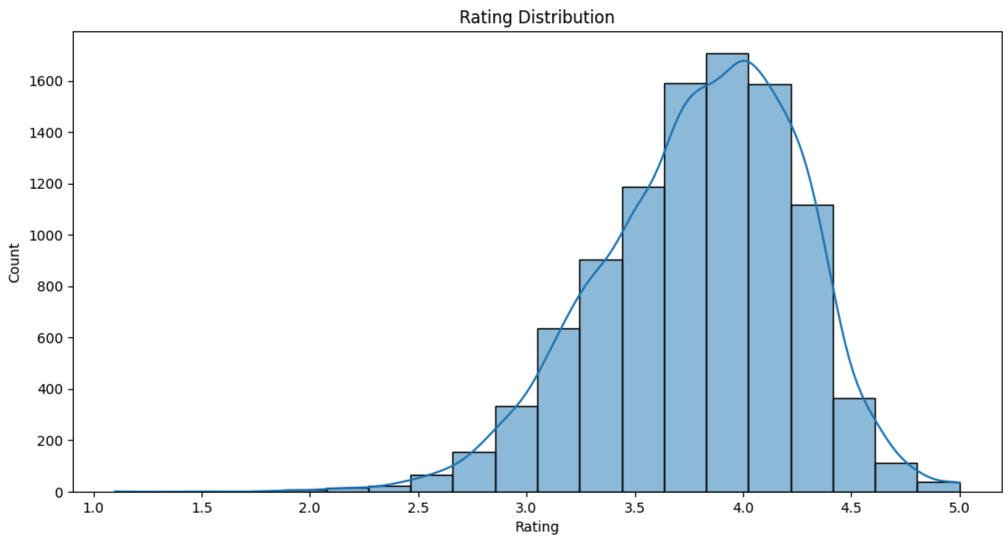
Milestone 3: Model Evaluation, Interpretation, Tool Development, and Final Presentation

- **Mar 29, 2025 – Apr 30, 2025:**
  • **Mar 29 – Apr 5, 2025:** Model Evaluation and Interpretation
  - Refine evaluation metrics (rating precision, cuisine match rate), perform additional testing, and analyze model trade-offs.
  • **Apr 6 – Apr 20, 2025:** Tool Development
  - Develop an interactive dashboard using Streamlit to integrate and visualize the recommendation engine.
  • **Apr 21 – Apr 30, 2025:** Final Presentation Preparation
  - Finalize the project report, prepare presentation slides, and create a short demo video showcasing the tool’s functionality.

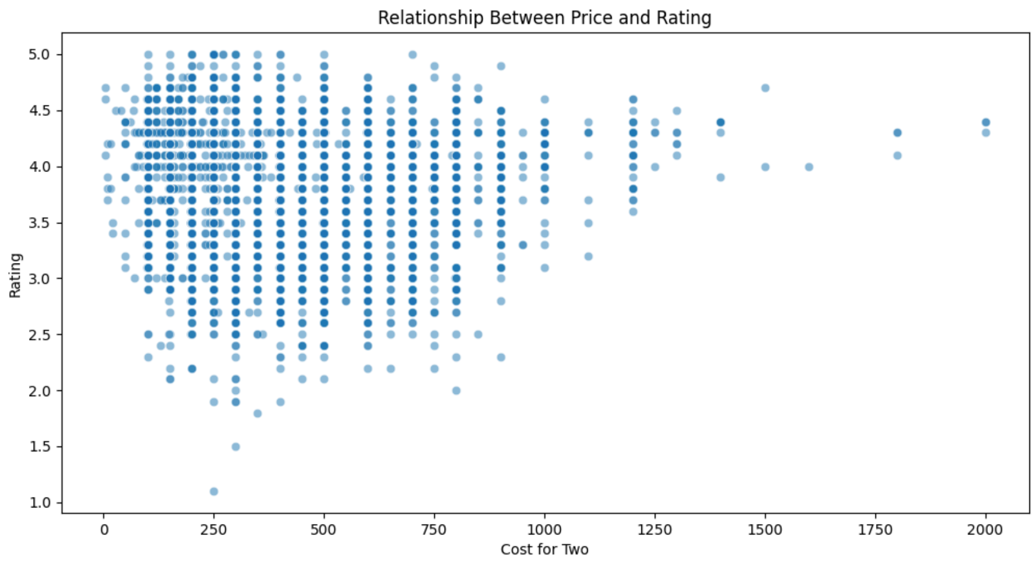
Exploratory Data Analysis (EDA) Recap:

Key findings from the initial EDA phase (detailed in 3_data_exploration.ipynb), which informed the subsequent feature engineering and modeling steps, include:

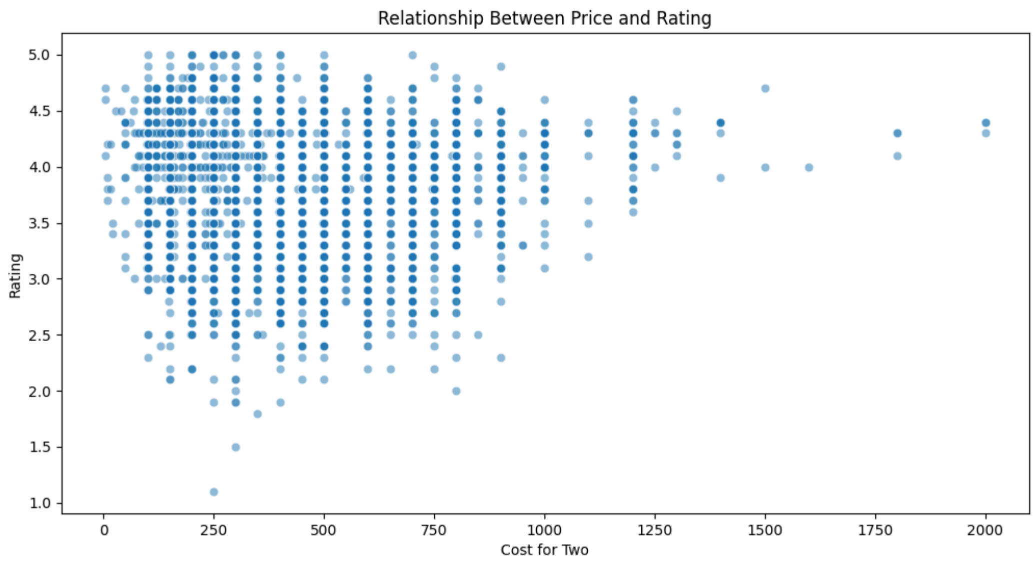
- Successful integration and cleaning of data from three distinct sources, resulting in the merged_restaurant_data.csv dataset used for further analysis.
- Restaurant ratings predominantly cluster between 3.5 and 4.5.



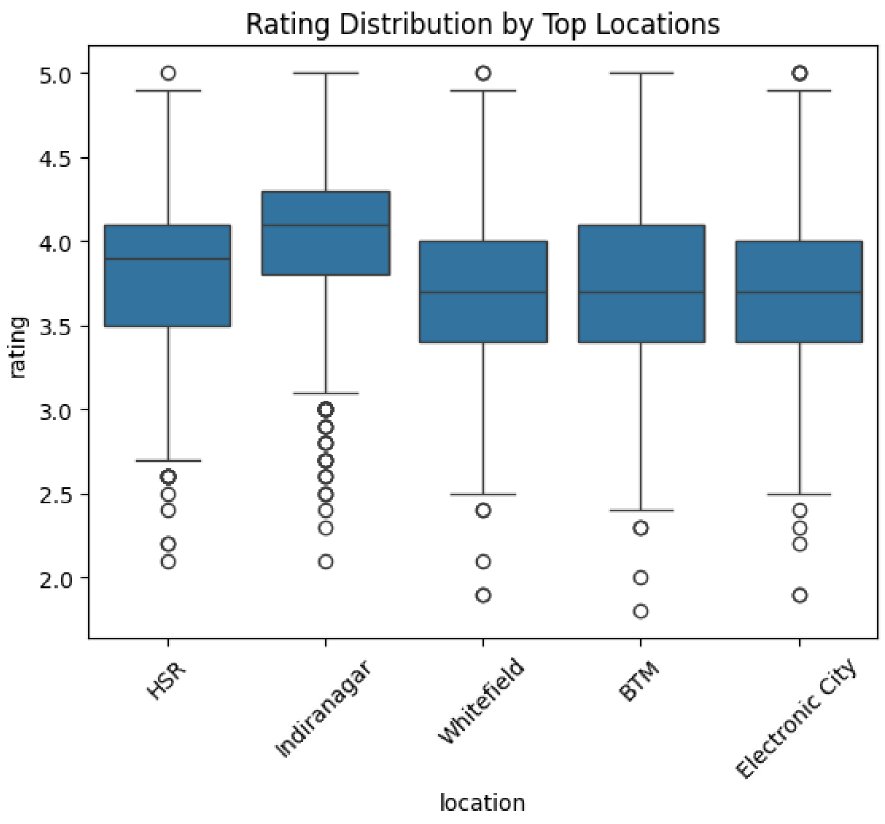
- The typical cost for two people ranges from approximately 125 INR to 500 INR, with a right-skewed distribution.



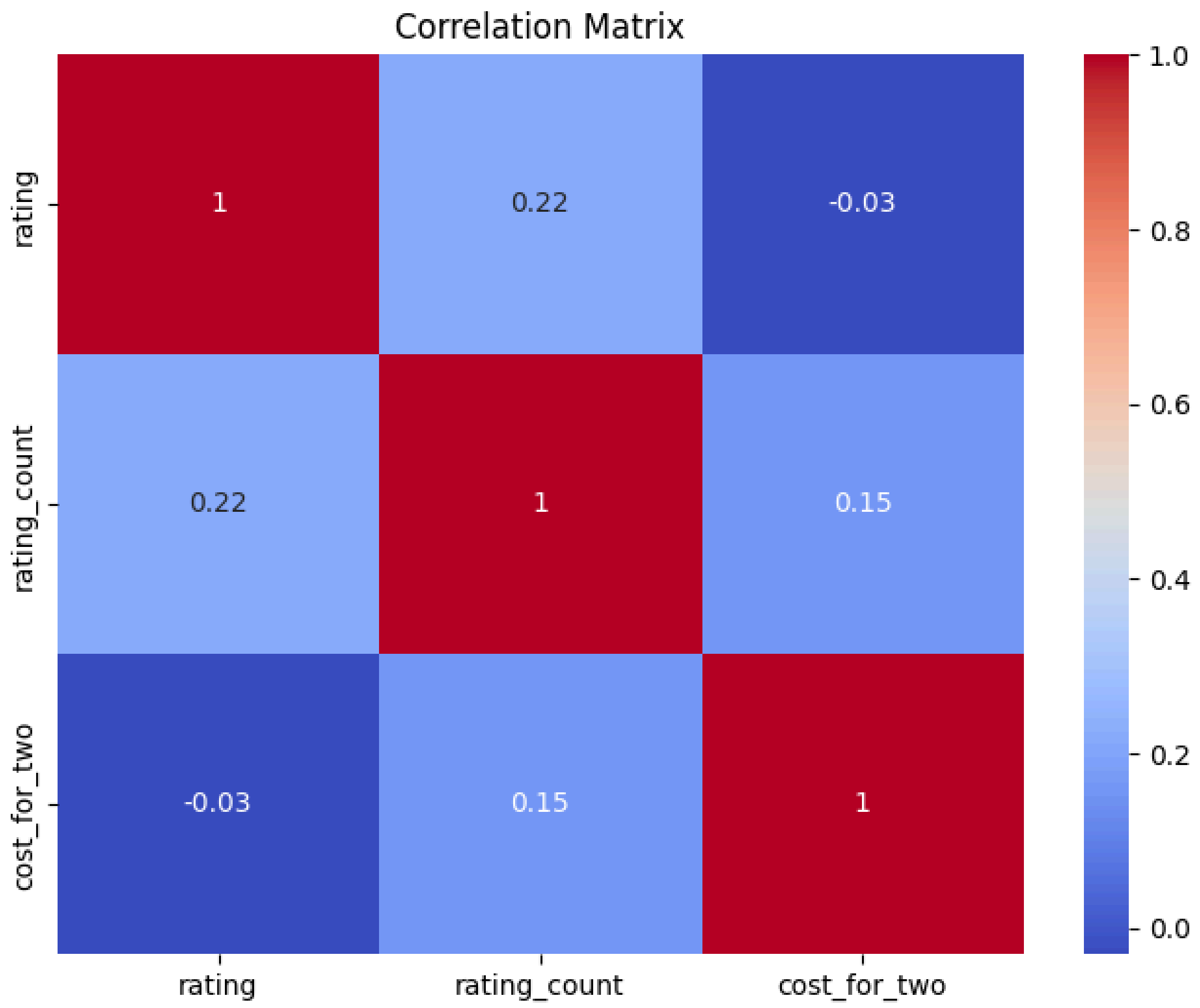
- A weak positive correlation exists between restaurant cost and rating, suggesting higher-rated establishments may be slightly more expensive.



- Analysis of ratings by location revealed variations, highlighting cuisine types.



- Correlation analysis showed weak to moderate relationships between numerical features (rating, rating_count, cost_for_two).



Feature Engineering

The process of creating meaningful features involved several steps executed across different notebooks:

1. Google Places API Enrichment (4_feature_engineering.ipynb):
To enhance the dataset with standardized and richer location information, the Google Places API (searchText endpoint) was used. For each restaurant listed in merged_restaurant_data.csv, an API call was made using its name and address to retrieve:

- place_id: A unique, stable identifier for the establishment.
- formatted_address: A standardized address string.
- place_types: A list of categories describing the establishment (e.g., 'restaurant', 'cafe', 'indian_restaurant').

This step was crucial for improving data quality, enabling reliable deduplication, and providing detailed categorical features. Error handling and rate limiting (time.sleep(0.2)) were implemented to manage the API call process effectively. The results, including a success flag, were saved to all_place_ids_results.csv.

```
# Illustrative snippet for fetching Place ID details
# (Full function definition in 4_feature_engineering.ipynb)
def get_place_id(name, address, api_key, ...):
  base_url = "https://places.googleapis.com/v1/places:searchText"
  search_query = f"{name}, {address}, Bangalore" # Simplified example
  headers = {
    "Content-Type": "application/json",
    "X-Goog-Api-Key": api_key,
    "X-Goog-FieldMask": "places.id,places.displayName,places.formattedAddress,places.types"
  }
  data = {"textQuery": search_query}
  try:
    response = requests.post(base_url, headers=headers, json=data)
    response.raise_for_status()
    result = response.json()
    if 'places' in result and len(result['places']) > 0:
      place = result['places'][0]
      return place['id'], place['formattedAddress'], place['types']
    else:
      return None, None, None
  except Exception as e:
        # ... (error logging) ...
        return None, None, None

# Processing the dataframe
# all_results_df = process_data(df, api_key, is_sample=False) # df is merged_restaurant_data.csv
# all_results_df.to_csv('all_place_ids_results.csv', index=False)
```

2. Data Merging and Cleaning (5_data_merging.ipynb):
The output from the Places API enrichment was processed to prepare the final dataset for modeling:

- The enriched data (all_place_ids_results.csv) was filtered to retain only records where the API call was successful.
- Duplicate entries were removed based on the unique place_id obtained from the API, ensuring each distinct establishment was represented only once.
- This cleaned and deduplicated data (containing place_id, formatted_address, place_types) was merged back with the original merged_restaurant_data.csv using name and address as keys. This step was essential to combine the newly acquired standardized information with the original restaurant attributes like rating, rating_count, cost_for_two, and cuisines.
- The final, integrated dataset was saved as final_df.csv.

```
# Illustrative snippet for merging enriched data
# (Code from 5_data_merging.ipynb)
df_enriched = pd.read_csv('../Data/feature_engineered_data.csv') # Assumed API output file
name
df_ratings = pd.read_csv('../Data/merged_restaurant_data.csv') # Original data with ratings/cost

# Filter for successful API calls and deduplicate by place_id
df_enriched = df_enriched[df_enriched['success'] == True]
df_enriched.drop_duplicates(subset=['place_id'], inplace=True)
df_enriched.reset_index(drop=True, inplace=True)
df_enriched.drop(columns=['success','index'], inplace=True) # Drop helper columns

# Merge enriched data with original ratings data
df_final = pd.merge(df_enriched, df_ratings, on=['name', 'address'], how='inner')
df_final.to_csv('../Data/final_df.csv', index=False)
```

3. Feature Creation (6_model.py):
Two key features were engineered from the final_df.csv specifically for use in the recommendation models:

- **content_features**: A composite textual feature created by concatenating the cuisines, location, and place_types (obtained from the Places API) for each restaurant. Missing values were handled by filling with empty strings. This feature aggregates key descriptive elements for content-based similarity calculations.
- **popularity_score**: A numerical score designed to represent a restaurant's general popularity and value proposition. It was calculated by:
    - Normalizing rating and rating_count using MinMaxScaler.
    - Normalizing cost_for_two and inverting the scale (so lower cost gets a higher score).
    - Combining these normalized scores with weights: 60% for rating, 30% for rating count, and 10% for inverse cost. This score provides a baseline recommendation metric independent of specific content similarity.

```
# Snippet for creating modeling features
# (Code from Notebooks/6_model.py)
df['content_features'] = df['cuisines'].fillna('') + ' ' + df['location'].fillna('') + ' ' +
df['place_types'].fillna('')
df['content_features'] = df['content_features'].astype(str)

# Popularity Score Calculation (details in 6_model.py)
# scaler = MinMaxScaler()
# df[['normalized_rating', 'normalized_rating_count']] = scaler.fit_transform(df[['rating',
'rating_count']])
# df['normalized_cost'] = 1 - scaler.fit_transform(df[['cost_for_two']])
# df['popularity_score'] = (df['normalized_rating'] * 0.6 +
#                     df['normalized_rating_count'] * 0.3 +
#                     df['normalized_cost'] * 0.1)
```

Categorical Variable Encoding (6_model.py):
The primary method for encoding the textual content_features was Term Frequency-Inverse Document Frequency (TF-IDF) vectorization. This technique converts the text descriptions into a sparse matrix of numerical values, where each value represents the importance of a term (word or n-gram) within a specific restaurant's description relative to the entire corpus. TF-IDF was chosen because it effectively captures the semantic meaning of text data, handles variations in description length, and is well-suited for calculating cosine similarity between documents (restaurants in my case).

```
# Snippet for TF-IDF Vectorization
# (Code from Notebooks/6_model.py)
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(min_df=2, # Ignore terms appearing in only one document
                        max_features=5000, # Limit vocabulary size
                        strip_accents='unicode',
                        analyzer='word',
                        token_pattern=r'\w{1,}', # Token pattern
                        ngram_range=(1, 2), # Use unigrams and bigrams
                        stop_words='english') # Remove common English words

tfidf_matrix = tfidf.fit_transform(df['content_features'])
# tfidf_matrix is now a sparse matrix (restaurants x features)
```

Code Quality and Documentation:

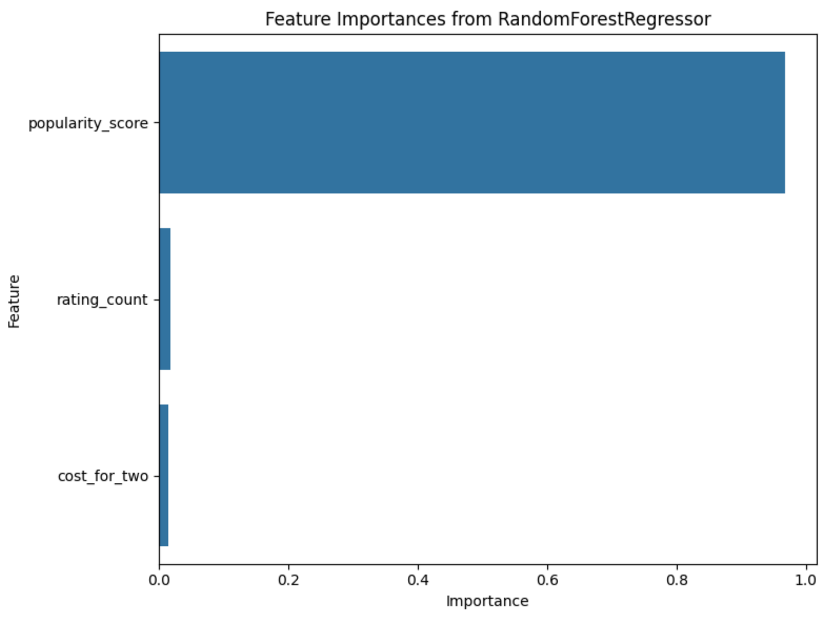
Code for all stages (data extraction, processing, EDA, feature engineering, merging, modeling) is organized into Jupyter Notebooks (1 through 6) and Python scripts (6_model.py). The code includes comments explaining logic and functionality. Key snippets illustrating the feature engineering process are included within this report, accompanied by explanations of their purpose and context.

Feature Selection

Feature Importance Evaluation:

Features like place_id, formatted_address, and place_types were deliberately engineered via the Google Places API because standardized identifiers and detailed establishment types were considered crucial for improving data quality, enabling reliable deduplication, and providing richer input for content analysis.

- **Modeling Input Selection:** The features directly fed into the recommendation models (6_model.py) were chosen based on their expected relevance:
  - cuisines, location, place_types: Combined into content_features as they represent core aspects users consider for similarity.
  - rating, rating_count, cost_for_two: Combined into popularity_score as they reflect overall quality, popularity, and value.



- Above bar chart represents the feature importances. Notably, popularity_score dominates the model, accounting for the vast majority of the predictive power. This outcome aligns with the design, since popularity_score encapsulates multiple aspects of a restaurant's quality and appeal (rating, rating_count, and affordability).
- Meanwhile, rating_count shows a moderate level of importance, confirming that the number of reviews is still a useful standalone predictor. cost_for_two contributes only a small portion to the overall prediction, suggesting that while cost can influence a restaurant's perceived quality, it is less impactful once popularity_score is included.
- These findings validate the feature engineering strategy: popularity_score effectively captures multiple relevant signals, making it the single strongest predictor of a restaurant's rating. Consequently, this feature's dominant importance highlights the value of combining raw attributes into a well-designed composite metric.
- **Implicit Importance (TF-IDF & SVD):**
  - TF-IDF inherently assigns higher weights to terms that are frequent within a specific restaurant's description but relatively rare across all descriptions, thus identifying terms important for distinguishing that restaurant.
  - Singular Value Decomposition (SVD) used in collaborative filtering identifies latent factors (underlying patterns) in the location-restaurant rating matrix, implicitly focusing on the most important dimensions of variation.

Feature Selection and Dimensionality Reduction:

- **Included Features:** The final set of features influencing the recommendations generated by 6_model.py includes cuisines, location, place_types (via content_features), rating, rating_count, and cost_for_two (via popularity_score). The place_id played a critical role in data preparation (deduplication) even if not directly used in the final scoring algorithms shown.
- **Excluded Features:** Features like the original address string (superseded by formatted_address and place_id), the source of the data, and the raw name were not directly incorporated into the core similarity or scoring logic of the models presented in 6_model.py. The name is, however, essential for presenting the final recommendations.
- **Dimensionality Reduction:**
  - **TF-IDF:** The max_features=5000 parameter in TfidfVectorizer limits the vocabulary size, effectively reducing the dimensionality of the text feature space by selecting the most frequent terms.
  - **SVD:** In the collaborative filtering component, SVD reduces the dimensionality of the utility matrix by projecting it onto a lower-dimensional latent factor space (controlled by the parameter k, set to min(50, num_factors)), capturing the most significant user-item interaction patterns.

Data Modeling

Data Splitting Strategy:
A traditional train/test split methodology was not employed for training the primary recommendation models (Content-based, Popularity, Hybrid). These models leverage the entire dataset to build their respective representations (e.g., TF-IDF matrix, popularity scores). This approach is common in recommendation systems where the goal is to recommend items from the full available catalog based on similarity or predicted preference.

For *evaluation*, a different strategy was used:

- A subset of restaurants (e.g., 100 random samples) was selected from the dataset.
- For each sampled restaurant, recommendations were generated using the different trained models.
- Custom metrics were calculated by comparing the recommended items to the properties of the sampled source restaurant.
  This simulates real-world recommendation scenarios and allows for comparison of model performance based on relevance, without requiring a hold-out set during the primary model construction phase. The Collaborative Filtering model (SVD) also utilized the full

utility matrix derived from the dataset.

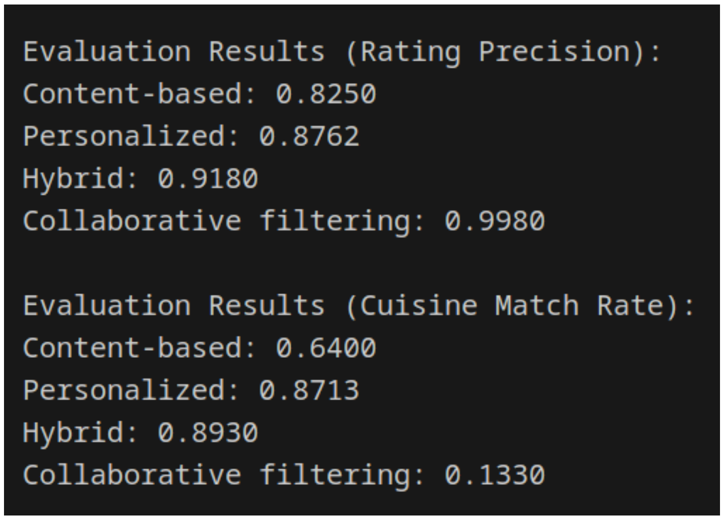
Model Training and Selection:
Several recommendation techniques were implemented to explore different signals within the data:

1. **Content-Based Filtering:** Recommends items similar to a source item (or user profile) based on shared content features. This model calculates the cosine similarity between the TF-IDF vectors (tfidf_matrix) of restaurants. Restaurants with higher cosine similarity scores to a target restaurant are recommended.
2. **Popularity-Based Filtering:** Provides a non-personalized baseline recommendation by simply ranking all restaurants based on the pre-calculated popularity_score (descending).
3. **Collaborative Filtering (CF):** Leverages patterns in user-item interactions. In this implementation:
    - A utility matrix was constructed with locations as 'users' and restaurants as 'items', populated with rating values.
    - Singular Value Decomposition (SVD) was applied to this matrix to decompose it into latent factor representations for locations and restaurants.
    - Predicted ratings for location-restaurant pairs were generated by reconstructing the matrix from the decomposed factors.
    - Recommendations for a given location are based on restaurants with the highest predicted ratings.
4. **Personalized Filtering:** Extends content-based filtering by incorporating explicit user preferences (e.g., preferred cuisine, location, budget range, previously liked items) to first filter candidate restaurants and then rank them based on a combination of content similarity and popularity.
5. **Hybrid Model:** Combines recommendations from multiple approaches (Content-based, Popularity, Collaborative Filtering) using weighted scores. This allows leveraging the strengths of different techniques (e.g., content similarity for specific tastes, popularity for general quality, CF for uncovering latent preferences). The weights (content_weight, pop_weight, cf_weight) can be tuned.

Model Evaluation and Comparison:
The evaluate_recommendations function within 6_model.py was designed to compare the performance of the different models based on the random sampling evaluation strategy described earlier.

- **Evaluation Metrics:** Custom metrics focused on relevance were used:
    - cuisine_match: The proportion of recommended restaurants that share at least one

cuisine type with the source (sampled) restaurant. This measures relevance based on cuisine similarity.
    - precision: The proportion of recommended restaurants whose rating is within a defined threshold (e.g., +/- 0.5) of the source restaurant's rating. This serves as a proxy for recommending items of comparable quality level.
- **Comparison:** The evaluation function calculates these metrics for recommendations generated by the Content-based, Personalized, Hybrid, and Collaborative Filtering models for each restaurant in the random sample. The average scores for cuisine_match and precision across all samples provide a basis for comparing the effectiveness of the different approaches.

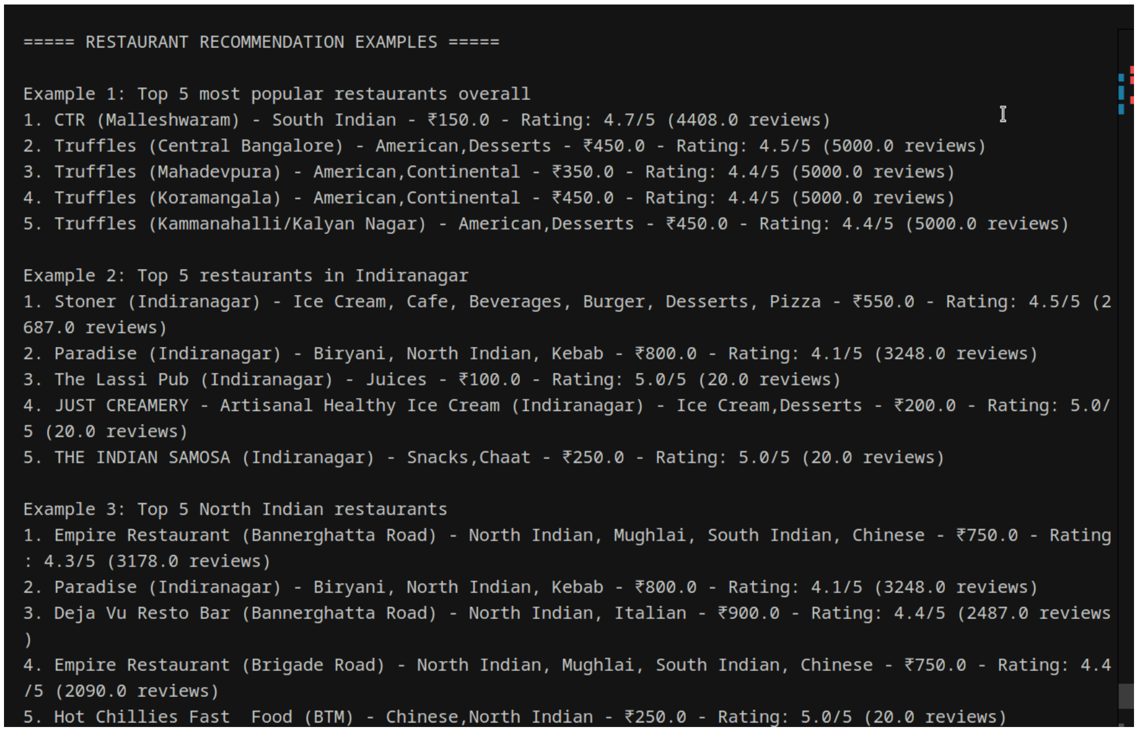


```
Evaluation Results (Rating Precision):
Content-based: 0.8250
Personalized: 0.8762
Hybrid: 0.9180
Collaborative filtering: 0.9980

Evaluation Results (Cuisine Match Rate):
Content-based: 0.6400
Personalized: 0.8713
Hybrid: 0.8930
Collaborative filtering: 0.1330
```

Based on the evaluation results, while the collaborative filtering model achieves an almost perfect rating precision of 0.9980, its extremely low cuisine match rate (0.1330) indicates that it fails to align the recommendations with the culinary preferences of the users. On the other hand, the hybrid model, with a rating precision of 0.9180 and an excellent cuisine match rate of 0.8930, demonstrates a much better balance between recommending high-quality restaurants and matching user-preferred cuisines. Although the personalized model performs reasonably well in both metrics, the hybrid approach—with its optimal weights (Content-based: 0.60, Collaborative filtering: 0.10, Popularity-based: 0.30)—delivers the most well-rounded recommendations. Thus, despite collaborative filtering's superior rating precision, the hybrid model is ultimately preferred as it offers recommendations that are both of high quality and closely aligned with the user's culinary tastes. Separately, drawing from my experience in Bangalore's restaurant scene, I found the recommendations to be highly relevant. Example outputs are shown in the next page.

EXAMPLE OUTPUTS:

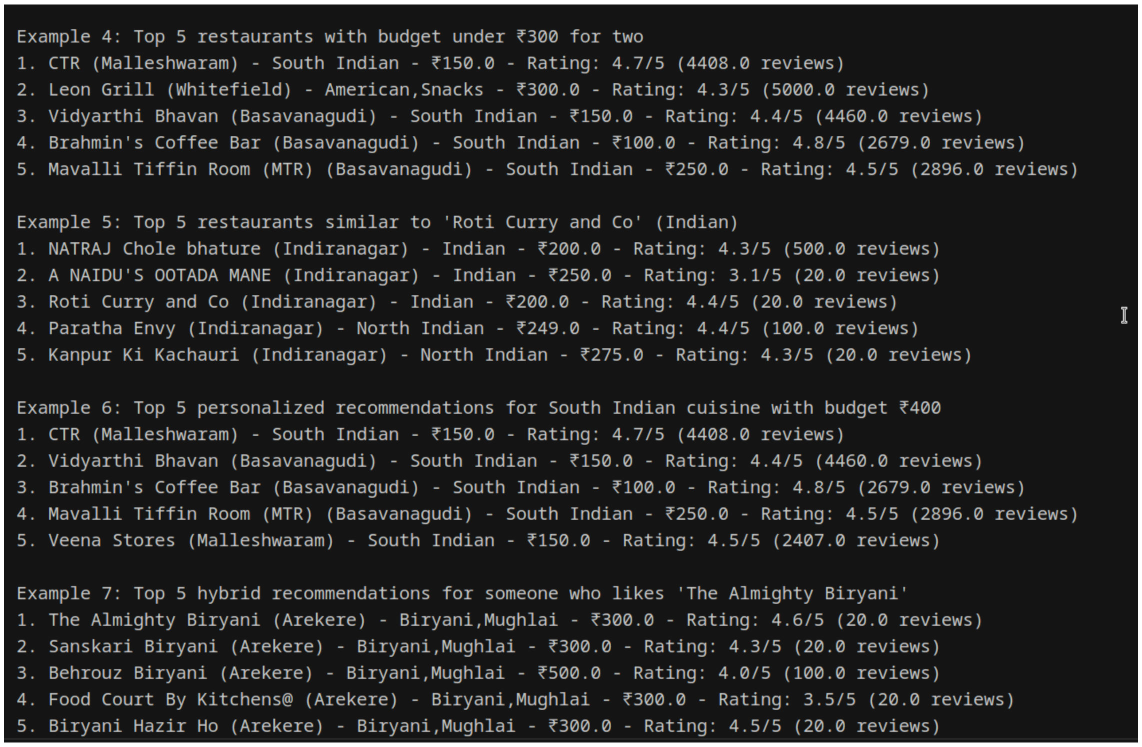


```
===== RESTAURANT RECOMMENDATION EXAMPLES =====

Example 1: Top 5 most popular restaurants overall
1. CTR (Malleshwaram) - South Indian - ₹150.0 - Rating: 4.7/5 (4408.0 reviews)
2. Truffles (Central Bangalore) - American,Desserts - ₹450.0 - Rating: 4.5/5 (5000.0 reviews)
3. Truffles (Mahadevpura) - American,Continental - ₹350.0 - Rating: 4.4/5 (5000.0 reviews)
4. Truffles (Koramangala) - American,Continental - ₹450.0 - Rating: 4.4/5 (5000.0 reviews)
5. Truffles (Kammanahalli/Kalyan Nagar) - American,Desserts - ₹450.0 - Rating: 4.4/5 (5000.0 reviews)

Example 2: Top 5 restaurants in Indiranagar
1. Stoner (Indiranagar) - Ice Cream, Cafe, Beverages, Burger, Desserts, Pizza - ₹550.0 - Rating: 4.5/5 (2
687.0 reviews)
2. Paradise (Indiranagar) - Biryani, North Indian, Kebab - ₹800.0 - Rating: 4.1/5 (3248.0 reviews)
3. The Lassi Pub (Indiranagar) - Juices - ₹100.0 - Rating: 5.0/5 (20.0 reviews)
4. JUST CREAMERY - Artisanal Healthy Ice Cream (Indiranagar) - Ice Cream,Desserts - ₹200.0 - Rating: 5.0/
5 (20.0 reviews)
5. THE INDIAN SAMOSA (Indiranagar) - Snacks,Chaat - ₹250.0 - Rating: 5.0/5 (20.0 reviews)

Example 3: Top 5 North Indian restaurants
1. Empire Restaurant (Bannerghatta Road) - North Indian, Mughlai, South Indian, Chinese - ₹750.0 - Rating
: 4.3/5 (3178.0 reviews)
2. Paradise (Indiranagar) - Biryani, North Indian, Kebab - ₹800.0 - Rating: 4.1/5 (3248.0 reviews)
3. Deja Vu Resto Bar (Bannerghatta Road) - North Indian, Italian - ₹900.0 - Rating: 4.4/5 (2487.0 reviews
)
4. Empire Restaurant (Brigade Road) - North Indian, Mughlai, South Indian, Chinese - ₹750.0 - Rating: 4.4
/5 (2090.0 reviews)
5. Hot Chillies Fast  Food (BTM) - Chinese,North Indian - ₹250.0 - Rating: 5.0/5 (20.0 reviews)
```



```
Example 4: Top 5 restaurants with budget under ₹300 for two
1. CTR (Malleshwaram) - South Indian - ₹150.0 - Rating: 4.7/5 (4408.0 reviews)
2. Leon Grill (Whitefield) - American,Snacks - ₹300.0 - Rating: 4.3/5 (5000.0 reviews)
3. Vidyarthi Bhavan (Basavanagudi) - South Indian - ₹150.0 - Rating: 4.4/5 (4460.0 reviews)
4. Brahmin's Coffee Bar (Basavanagudi) - South Indian - ₹100.0 - Rating: 4.8/5 (2679.0 reviews)
5. Mavalli Tiffin Room (MTR) (Basavanagudi) - South Indian - ₹250.0 - Rating: 4.5/5 (2896.0 reviews)

Example 5: Top 5 restaurants similar to 'Roti Curry and Co' (Indian)
1. NATRAJ Chole bhature (Indiranagar) - Indian - ₹200.0 - Rating: 4.3/5 (500.0 reviews)
2. A NAIDU'S OOTADA MANE (Indiranagar) - Indian - ₹250.0 - Rating: 3.1/5 (20.0 reviews)
3. Roti Curry and Co (Indiranagar) - Indian - ₹200.0 - Rating: 4.4/5 (20.0 reviews)
4. Paratha Envy (Indiranagar) - North Indian - ₹249.0 - Rating: 4.4/5 (100.0 reviews)
5. Kanpur Ki Kachauri (Indiranagar) - North Indian - ₹275.0 - Rating: 4.3/5 (20.0 reviews)

Example 6: Top 5 personalized recommendations for South Indian cuisine with budget ₹400
1. CTR (Malleshwaram) - South Indian - ₹150.0 - Rating: 4.7/5 (4408.0 reviews)
2. Vidyarthi Bhavan (Basavanagudi) - South Indian - ₹150.0 - Rating: 4.4/5 (4460.0 reviews)
3. Brahmin's Coffee Bar (Basavanagudi) - South Indian - ₹100.0 - Rating: 4.8/5 (2679.0 reviews)
4. Mavalli Tiffin Room (MTR) (Basavanagudi) - South Indian - ₹250.0 - Rating: 4.5/5 (2896.0 reviews)
5. Veena Stores (Malleshwaram) - South Indian - ₹150.0 - Rating: 4.5/5 (2407.0 reviews)

Example 7: Top 5 hybrid recommendations for someone who likes 'The Almighty Biryani'
1. The Almighty Biryani (Arekere) - Biryani,Mughlai - ₹300.0 - Rating: 4.6/5 (20.0 reviews)
2. Sanskari Biryani (Arekere) - Biryani,Mughlai - ₹300.0 - Rating: 4.3/5 (20.0 reviews)
3. Behrouz Biryani (Arekere) - Biryani,Mughlai - ₹500.0 - Rating: 4.0/5 (100.0 reviews)
4. Food Court By Kitchens@ (Arekere) - Biryani,Mughlai - ₹300.0 - Rating: 3.5/5 (20.0 reviews)
5. Biryani Hazir Ho (Arekere) - Biryani,Mughlai - ₹300.0 - Rating: 4.5/5 (20.0 reviews)
```