**Swiggy Restaurant Recommendation System - Project Report**

**Executive Summary**

I developed a restaurant recommendation system using Swiggy's restaurant data to help users discover restaurants based on their preferences. The system uses machine learning techniques including data preprocessing, one-hot encoding, and cosine similarity to provide personalized restaurant recommendations through an interactive Streamlit web application.

**1. Project Approach**

**1.1 Problem Definition**

The goal was to build a recommendation system that suggests restaurants to users based on their input preferences such as city, cuisine type, minimum rating, and maximum cost. I needed to process raw restaurant data and create a user-friendly interface for getting recommendations.

**1.2 Technical Approach**

I followed a systematic approach divided into three main phases:

**Phase 1: Data Cleaning and Preprocessing**

* Removed duplicate restaurants using unique IDs
* Cleaned numerical columns (rating, rating\_count, cost) by removing special characters
* Handled missing values by dropping incomplete records
* Saved the cleaned dataset for further processing

**Phase 2: Feature Engineering**

* Applied One-Hot Encoding to categorical features (city, cuisine)
* Handled multi-label cuisines by splitting comma-separated values
* Created binary features for each cuisine type
* Combined numerical and encoded categorical features
* Saved the encoder for use in the Streamlit application

**Phase 3: Recommendation Engine Development**

* Implemented cosine similarity to find similar restaurants
* Created a filtering system based on user preferences
* Built a mapping system to return results from the original cleaned dataset
* Developed a Streamlit web interface for user interaction

**1.3 Technology Stack**

* **Python**: Core programming language
* **Pandas**: Data manipulation and analysis
* **Scikit-learn**: Machine learning tools (OneHotEncoder, cosine\_similarity)
* **Streamlit**: Web application framework
* **Pickle**: Model serialization

**2. Data Analysis**

**2.1 Dataset Overview**

The original dataset contained restaurant information with the following key columns:

* **Categorical**: name, city, cuisine, address
* **Numerical**: rating, rating\_count, cost
* **Additional**: id, lic\_no, link, menu

**2.2 Data Quality Issues Identified**

During the data cleaning process, I discovered several issues:

**Missing Values:**

* Some restaurants had missing ratings (marked as '--')
* Cost information was inconsistent with currency symbols
* Rating counts contained non-numeric characters

**Data Inconsistencies:**

* Duplicate restaurant entries with same IDs
* Multi-label cuisines stored as comma-separated strings
* Special characters in numerical fields

**Data Distribution:**

* Ratings ranged from 1.0 to 5.0 with most restaurants having ratings between 3.5-4.5
* Costs varied significantly across different cities and cuisine types
* Popular cuisines included Indian, Fast Food, Chinese, and Continental

**2.3 Preprocessing Results**

After cleaning and preprocessing:

* **Original dataset**: Various entries with inconsistencies
* **Cleaned dataset**: Consistent format with no missing critical values
* **Encoded dataset**: Numerical format ready for machine learning algorithms
* **Feature count**: Expanded significantly due to one-hot encoding of cities and cuisines

**3. Recommendation System Implementation**

**3.1 Algorithm Choice**

I chose **cosine similarity** for the recommendation engine because:

* It works well with high-dimensional sparse data (one-hot encoded features)
* It measures similarity between user preferences and restaurant features
* It's computationally efficient for real-time recommendations
* It handles categorical data well after encoding

**3.2 Recommendation Process**

The system works in the following steps:

1. **User Input Processing**: Convert user preferences into the same format as training data
2. **Filtering**: Apply user constraints (minimum rating, maximum cost)
3. **Encoding**: Transform user's city and cuisine preferences using the saved encoder
4. **Similarity Calculation**: Compute cosine similarity between user vector and restaurant features
5. **Ranking**: Sort restaurants by similarity score and return top N recommendations
6. **Result Mapping**: Map encoded results back to original restaurant information

**3.3 User Interface**

I developed a Streamlit application with:

* **Sidebar Controls**: City selection, cuisine multiselect, rating/cost sliders
* **Interactive Filtering**: Real-time preference adjustment
* **Results Display**: Professional restaurant cards with all relevant information
* **Dataset Overview**: Charts and statistics about the restaurant data

**4. Key Insights and Findings**

**4.1 Data Insights**

**Geographic Distribution:**

* Restaurant availability varies significantly across cities
* Some cities have more diverse cuisine options than others
* Cost patterns differ based on city location

**Cuisine Patterns:**

* Indian cuisine is the most common across all cities
* Fast Food has consistent pricing across locations
* Continental and Chinese cuisines tend to be more expensive

**Rating Patterns:**

* Most restaurants maintain ratings between 3.5-4.5
* Higher-rated restaurants often have fewer reviews
* Cost doesn't always correlate with rating

**4.2 System Performance**

The recommendation system performs well by:

* Providing relevant suggestions based on user preferences
* Handling edge cases like unknown cities or cuisines
* Delivering fast response times through efficient similarity calculations
* Maintaining data consistency between encoded and original datasets

**5. Recommendations and Future Improvements**

**5.1 Current System Strengths**

* **User-Friendly**: Simple interface that anyone can use
* **Flexible**: Allows multiple preference combinations
* **Fast**: Quick recommendation generation
* **Comprehensive**: Shows all relevant restaurant information

**5.2 Suggested Improvements**

**Short-term Enhancements:**

* Add distance-based filtering if location coordinates are available
* Include restaurant images in the display
* Add user rating and review functionality
* Implement sorting options (by rating, cost, popularity)

**Long-term Enhancements:**

* Implement collaborative filtering using user behavior data
* Add machine learning models for better preference learning
* Include seasonal menu recommendations
* Develop mobile application version

**Data Quality Improvements:**

* Regular data updates to maintain current information
* Better handling of new cities and cuisines
* Integration with real-time restaurant data APIs
* User feedback collection for recommendation improvement

**5.3 Business Applications**

This system can be used for:

* **Customer Service**: Help users find restaurants matching their preferences
* **Marketing**: Understand popular cuisine and location combinations
* **Business Intelligence**: Analyze customer preferences for strategic decisions
* **Partner Relations**: Provide insights to restaurant partners about customer demands

**6. Conclusion**

I successfully developed a functional restaurant recommendation system that processes raw restaurant data and provides personalized suggestions through an intuitive web interface. The system effectively combines data preprocessing, machine learning, and web development to create a practical solution for restaurant discovery.

The project demonstrates the complete machine learning pipeline from data cleaning to deployment, showing how theoretical concepts can be applied to solve real-world problems. The modular design allows for easy maintenance and future enhancements, making it a solid foundation for a production recommendation system.

**Key Achievements:**

* Built end-to-end recommendation system
* Created clean, user-friendly web interface
* Implemented efficient similarity-based algorithm
* Maintained data consistency throughout the pipeline
* Delivered within the 7-day project timeline

This project serves as a strong foundation for understanding recommendation systems and can be extended with more advanced features as business needs evolve.