Task 1: Restaurant Rating Prediction
Built ML models (Linear Regression, Random Forest, Gradient Boosting) to predict ratings based on features like cuisine, price, and location. Visualized patterns and feature importance.

Task 2: Restaurant Recommendation System
Created a content-based recommendation engine using feature similarity (e.g., price, cuisine) to suggest restaurants based on user preferences.

Task 3: Cuisine Classification
Used models (Logistic Regression, SVM, Random Forest, Gradient Boosting) to classify restaurants by cuisine type. Included hyperparameter tuning and clustering.

Task 4: Geographic Analysis
Mapped restaurant locations using latitude/longitude. Analyzed rating trends, cuisine distribution, and city-wise restaurant density.

🧠 Interview Questions & Answers

Q1: What was your project about?
A: I built a full-stack ML system for restaurant analytics—predicting ratings, recommending restaurants, classifying cuisines, and analyzing geographic patterns.

Q2: What business problems does it solve?
A: It helps restaurant owners understand rating drivers, automates classification, supports location decisions, and improves user experience via personalized recommendations.

🔧 Technical Implementation

Q3: How did you preprocess the data?
A: Cleaned missing values, created features like cuisine count and service indicators, and standardized numerical data using StandardScaler.

Q4: Why these ML algorithms?
A: Choose Linear Regression as baseline; Random Forest and Gradient Boosting for their handling of non-linear relationships. Used SVM for classification with tuning.

Q5: How did you evaluate models?
A: Used RMSE, MAE, and R^2 for regression; accuracy, precision, recall, and confusion matrices for classification. Cross-validation ensured robustness.

🧪 Feature Engineering

Q6: What features did you create?
A: Created domain features like cuisine count and binary flags for popular cuisines. Converted service options into binary features.

Q7: How did you handle categorical data?
A: Used binary encoding for simple categories and selected binary indicators for popular cuisines instead of full one-hot encoding to reduce sparsity.

⚙️ Model Selection & Recommendation Challenges

Q8: How did you pick the best model?
A: Compared algorithms on RMSE; considered performance, interpretability, and efficiency. Tuned hyperparameters using GridSearchCV.

Q9: Challenges in the recommender system?
A: Cold-start issue and lack of diversity. Used content-based filtering with feature similarity and randomization for varied suggestions.

📊 Data Analysis & Visualization

Q10: Key insights from EDA?
A: Higher prices are often linked to higher ratings. Italian/Japanese cuisine scored well. Table booking and delivery correlated with better ratings.

Q11: How did you visualize geographic data?
A: Used Plotly maps to show city-wise restaurant density, average ratings by region, and cuisine distribution.

🚀 Scalability & MLOps

Q12: How would you scale this?
A: Use feature stores, model versioning, data drift monitoring, caching for popular queries, and deploy via microservices.

Q13: What metrics to monitor in production?
A: Track latency, accuracy, user engagement, prediction drift, and model confidence. Monitor data quality and system health.

🔬 Advanced Concepts

Q14: How would you improve recommendations?
A: Add hybrid filtering, deep learning, real-time availability, temporal trends, and explainability to improve personalization.

Q15: What would you do with more time?
A: Add neural networks, better feature engineering, automated pipelines (MLOps), Bayesian tuning, and fairness metrics.

📚 Theory & Fairness

Q16: Why did Random Forest outperform Linear Regression?
A: RF captures non-linear interactions that linear models miss. It models complex rules like conditional splits in features.

Q17: How to handle concept drift?
A: Monitor feature/rating shifts with PSI and KS tests. Use sliding windows, retrain models periodically, or apply adaptive learning.

Q18: Limitations in your evaluation?
A: Lacked temporal validation, fairness metrics, and business impact assessment. RMSE alone doesn't capture user satisfaction.

🧱 System Design

Q19: How to design a real-time recommendation API?
A: Use microservices, Redis for caching/feature store, Faiss for similarity search, load balancing, and Kubernetes for scaling.

Q20: What if restaurant features change frequently?
A: Use event-driven pipelines (Kafka), real-time updates to feature stores, online learning, and fallback strategies.

💡 Business & Research

Q21: Is price causing higher ratings or just correlated?
A: Use causal methods like propensity score matching, IVs, natural experiments, and controlled studies to isolate causality.

Q22: Concern about bias against ethnic cuisines?
A: Analyze per-cuisine accuracy, check data balance, assess proxy bias, apply fairness metrics, and retrain with diverse data.

Q23: How to enhance the model with external data?
A: Use social media sentiment, weather, events, economic indicators, and scrape competitor data for richer context.

Q24: Propose a new ML approach for recommendations.
A: Use Graph Neural Networks to model users, cuisines, and locations as nodes, capturing deeper multi-hop relationships.

🛠️ Troubleshooting in Production

Q25: Model degrading in production—what's your approach?
A: Check data pipeline, feature drift, model versioning, prediction patterns, infra bottlenecks, and run A/B tests for comparison.