Data Science & ML Engineering Interview Preparation

SECTION 1: ROLE-SPECIFIC TECHNICAL QUESTIONS

A. Machine Learning Fundamentals

Q1: What's the difference between classification and regression? Can you give examples from your projects? A: Classification predicts categorical outcomes (fraud vs. non-fraud, claim denial vs. approval), while regression predicts continuous values. In your Claim Denial Risk Prediction project, you used Logistic Regression for binary classification (denial probability). For healthcare, you could also do regression to predict claim value or processing time.

Q2: What is imbalanced data and how did you handle it in your claim denial project? A: Imbalanced data occurs when one class is much rarer than others. Healthcare claims often have far fewer denials than approvals. Handling strategies: use stratified train-test splits, adjust class weights in models, try SMOTE for synthetic oversampling, use metrics like F1-score or AUC-ROC instead of accuracy, or apply threshold tuning. With your Logistic Regression, you could adjust the probability threshold to optimize for precision/recall tradeoff based on business impact.

Q3: Explain your feature engineering process. What features did you create for claim denial prediction? A: Feature engineering transforms raw data into meaningful predictors. For healthcare claims, examples: claim amount (ratio to typical claims), claim type (emergency vs. routine), time-to-adjudication, provider history (claim approval rate), member age, diagnosis complexity. You extracted 10+ features—discuss which ones had highest importance, how you validated them, and which were domain-specific insights.

Q4: What's the difference between training, validation, and test sets? Why can't you use test data for hyperparameter tuning? A: Training set (60-70%) trains the model, validation set (15-20%) tunes hyperparameters and prevents overfitting, test set (15-20%) evaluates final performance. Using test data for tuning causes data leakage—the model "sees" test data indirectly, inflating its reported performance. True test performance will be worse in production.

Q5: What metrics did you use to evaluate your models? Why these metrics for claim denial prediction? A: For imbalanced classification: Precision (false positives cost—incorrect denials upset customers), Recall (false negatives—missed fraud costs money), F1-score (harmonic mean), ROC-AUC (threshold-independent), and confusion matrix. For healthcare claims, false positives harm customer trust while false negatives cost revenue. Choose based on business priorities.

Q6: What is overfitting? How do you detect and prevent it? A: Overfitting occurs when a model learns training data noise rather than true patterns, performing well on training data but poorly on unseen data. Detection: large gap between training and validation metrics. Prevention: use simpler models, regularization (L1/L2), cross-validation, early stopping, more training data, or dropout in neural networks.

Q7: Explain cross-validation. Why is it better than a single train-test split? A: Cross-validation (typically k-fold, k=5 or 10) splits data into k parts, training k times with different validation sets. It provides robust performance estimates by reducing variance from a single random split. Essential for small datasets and ensures your model generalizes across different data subsets.

B. Data Science & Analytics

Q8: Walk us through your claim denial risk prediction project end-to-end. A: Stakeholder requirement→ exploratory data analysis to understand claim patterns → feature engineering (10+ healthcare-specific features) → train Logistic Regression on balanced data → validate on hold-out set with F1/ROC-AUC metrics → deploy via FastAPI for real-time predictions → batch scoring daily via MLflow-tracked workflows → monitor performance in production. Discuss business impact (e.g., "enabled proactive denial prevention, saving $X annually").

Q9: How did you approach fraud detection in your automated fraud claims project? A: Started with EDA to identify patterns in fraudulent vs. legitimate claims (claim amount outliers, frequency anomalies, provider behavior). Engineered domain-specific features capturing suspicious patterns. Applied multiple models (Logistic Regression, Random Forest, XGBoost) and compared via MLflow. Fraud detection prioritizes recall (catching fraud) over precision, so optimized F1-score accordingly. Batch scoring ensures all claims are scored daily.

Q10: What is exploratory data analysis (EDA) and why is it crucial? A: EDA explores data before modeling to uncover patterns, distributions, outliers, missing values, and relationships. For healthcare claims, EDA reveals: claim volume trends, denial patterns, provider variations, seasonal effects. It informs feature engineering, identifies data quality issues, and ensures you understand the problem domain before building models. Always start with EDA—it prevents bad model choices downstream.

Q11: How do you handle missing data? A: Strategies depend on missingness pattern and proportion. For MCAR (Missing Completely at Random), deletion is acceptable if <5%. For MAR/MNAR, imputation is better: mean/median for continuous, mode for categorical, or model-based imputation. In healthcare,

missing diagnosis codes might indicate incomplete claims—create a "missing" indicator feature. Document your approach for reproducibility.

Q12: Describe your data cleaning and validation process in production. A: At Infosys, you performed validation (outlier checks, consistency analysis across sources), cleaning (removing duplicates, standardizing formats), and quality checks. For ML pipelines, validation should occur at every stage: raw data validation → cleaned data validation → feature validation → prediction validation. Automated checks prevent bad data from corrupting models.

Q13: You used Power BI for reporting. How did you design dashboards for operational analytics? A: Created dashboards tracking claim volumes, processing status, SLA metrics. Good dashboard design: clear KPIs aligned to business goals, drill-down capability (summary → details), filters for slicing, automated refresh schedules. Dashboards translate model insights into actionable intelligence for stakeholders.

Q14: What is statistical analysis? Give an example from your work. A: Statistical analysis validates hypotheses and quantifies uncertainty. Example: "Are claim denial rates significantly different across providers?" Use t-tests or chi-square tests to answer this. Statistical rigor prevents false conclusions and ensures business decisions are data-driven.

C. ML Engineering & MLOps

Q15: Explain your end-to-end ML pipeline architecture. How did you ensure reproducibility? A: Pipeline flow: data ingestion → validation → feature engineering → model training → evaluation → deployment → monitoring. Reproducibility tools: MLflow tracks experiments (parameters, metrics, artifacts), Docker containerizes the environment (same dependencies everywhere), version control (Git) tracks code changes, and scheduled batch jobs run deterministically. This ensures the same code produces the same results across development, staging, and production.

Q16: What is MLflow and how did you use it in your projects? A: MLflow is an open-source platform for ML lifecycle management. Components: Tracking (logs parameters, metrics, artifacts), Projects (packages code reproducibly), Models (stores trained models), Registry (manages versions). You used it for experiment tracking in claim denial and fraud projects—enabling team collaboration, comparing model versions, and ensuring auditability.

Q17: Tell us about your FastAPI service for real-time predictions. How does it differ from batch processing? A: FastAPI is a modern Python web framework for building APIs. Real-time API: client sends request → model predicts immediately → returns response (latency-sensitive, e.g., milliseconds). Batch

processing: processes thousands of records scheduled (latency-tolerant, e.g., daily). Your fraud project used both: FastAPI for on-demand scoring and batch workflows for daily scheduled scoring.

Q18: Explain Docker and why containerization is important for ML. A: Docker packages your application (code, dependencies, configuration) in a container—isolated, portable environment. Benefits: reproducibility across machines, simplified deployment, easy scaling, and version control of entire environments. You containerized your claim denial pipeline, ensuring it runs identically in development, staging, and production (Render cloud).

Q19: How do you handle model versioning and experiment tracking? A: MLflow Registry stores model versions with metadata (parameters, performance metrics). Tags mark production vs. staging models. This enables rollback if a new model underperforms and maintains an audit trail. Pair with Git for code versioning and Docker for environment versioning—complete reproducibility.

Q20: What monitoring would you implement in production for your ML models? A: Monitor prediction distribution (detect data drift), model performance metrics (degradation signals), prediction latency (API performance), error rates, and data quality (validate input schema). Set alerts if metrics deviate from baseline. Example: if claim denial rate prediction suddenly becomes very certain, that signals distribution shift. Automated monitoring enables rapid issue detection.

Q21: What is data drift and how do you detect it? A: Data drift occurs when input data distribution changes (e.g., new claim types, different provider mix). This causes model performance degradation. Detection: compare feature distributions between training and production data using statistical tests (Kolmogorov-Smirnov test) or visualization. For healthcare, drift might indicate new fraud patterns or claim types requiring model retraining.

Q22: How would you structure an ML project for scalability? A: Modular architecture: separate data pipelines, feature engineering, model training, evaluation, and serving. Use configuration files (YAML) for parameters instead of hardcoding. Implement logging and monitoring. Use cloud platforms (Azure, AWS, GCP) for scalable compute. Version everything (code, data, models). Your projects demonstrate this—FastAPI, Docker, MLflow, and scheduled batch jobs are scalable patterns.

D. Data Engineering & SQL

Q23: Explain ETL (Extract, Transform, Load). How does it differ from ELT? A: ETL: extract raw data → transform (clean, validate, engineer features) → load

into data warehouse. ELT: extract → load raw → transform in warehouse (modern approach, leverages warehouse compute). At Infosys, you performed ETL tasks: extracting from FACETS, transforming via SQL/Python, loading operational reports. ELT is preferred for big data as warehouses scale better.

Q24: Walk us through a complex SQL query you optimized. What was the performance improvement? A: Discuss specific queries at Infosys: perhaps a join-heavy reconciliation query or aggregation across claim transactions. Optimization techniques: add indexes on filter/join columns, use CTEs for readability and query optimization, window functions for ranking/running totals instead of self-joins, materialized views for frequently-accessed aggregations, avoid SELECT * and unnecessary subqueries. Quantify improvement: "reduced query runtime from 30 seconds to 2 seconds."

Q25: What are window functions and when would you use them? A: Window functions compute aggregate/rank values over data subsets without collapsing rows. Examples: ROW_NUMBER() (rank claims by amount), LEAD/LAG (compare current vs. previous claim), SUM() OVER (running total), RANK(). Claim workflow example: "Find members with abnormally high claim frequency" using COUNT() OVER (PARTITION BY member_id). Window functions replace complex self-joins, improving readability and performance.

Q26: Explain indexing strategy for healthcare claims databases. A: Create indexes on frequently-filtered columns (claim_id, member_id, claim_date, status) and foreign keys. Multi-column indexes for common WHERE clauses (claim_date + status). Avoid over-indexing—each index slows writes. For your claims data, indexes on claim_date enable fast date-range queries; member_id indexing speeds member-specific analyses.

Q27: How do you handle data quality issues in production pipelines? A: Implement data validation checks: schema validation (correct column types), completeness (nulls), uniqueness (no duplicates), referential integrity (foreign keys valid), and business logic validation (claim amount > 0). Log quality metrics—if checks fail, alert engineers and halt the pipeline to prevent bad data downstream. You did this at Infosys with validation scripts.

Q28: Explain stored procedures and when they're beneficial. A: Stored procedures are pre-compiled SQL routines stored in the database, reducing network traffic and improving performance. Beneficial for complex, frequently-executed logic (monthly claim processing). They're harder to version-control than application code, so use sparingly. You used them for reconciliation and operational reporting at Infosys.

Q29: What's the difference between INNER JOIN, LEFT JOIN, and FULL OUTER JOIN? A: INNER JOIN returns rows matching in both tables, LEFT JOIN includes unmatched left rows (with NULLs), FULL OUTER JOIN includes all

rows. For claims: INNER JOIN member + claims (only members with claims), LEFT JOIN members + claims (all members, including those without claims for attrition analysis), FULL OUTER JOIN for data reconciliation (find claims with missing member records).

Q30: How would you optimize a slow data warehouse query? A: Diagnose with EXPLAIN PLAN to see execution strategy. Optimize: add indexes, rewrite joins efficiently, partition large tables (by date), use materialized views for common aggregations, avoid full table scans, consider data denormalization for analytics. Healthcare claims queries often scan millions of rows—partitioning by claim_date is critical.

SECTION 2: BEHAVIORAL & SITUATIONAL QUESTIONS

Q31: Tell us about a complex data problem you solved at Infosys. What was your approach? A: Describe a specific issue: "Healthcare claims batch failures were causing SLA breaches. I performed root cause analysis, identified missing member records in upstream system, developed SQL validation script to detect gaps preemptively, and coordinated with stakeholders for resolution. Reduced incidents by 40%." Show: problem-solving, ownership, impact.

Q32: Describe a time you proposed an automation or process improvement. A: "I noticed manual reconciliation checks consumed 8 hours weekly. I developed a Python utility automating validation across data sources, reducing manual effort to 1 hour. This freed team capacity for strategic analysis." Shows: initiative, technical capability, business sense.

Q33: How do you stay current with ML/data science trends? A: "I learn through portfolio projects applying latest tools—I recently used Streamlit for building data quality tools and MLflow for experiment tracking. I read Towards Data Science articles, follow Kaggle competitions, and experiment with new libraries." Shows: continuous learning, practical engagement.

Q34: Tell us about a time you had to explain technical concepts to non-technical stakeholders. A: "I presented claim denial prediction model to claims management team unfamiliar with ML. I explained using business terminology: 'The model identifies high-risk claims before processing, enabling proactive outreach to prevent denials.' I showed dashboard, not math, focusing on business impact." Shows: communication, stakeholder awareness.

Q35: Describe a conflict with a team member and how you resolved it. A: "Disagreed with data engineer on pipeline architecture. We discussed tradeoffs, agreed on hybrid approach combining both ideas, and it performed better. Learned that collaboration beats ego." Shows: teamwork, humility, problem-solving.

Q36: Why are you transitioning from System Engineer to Data Science/ML Engineering? A: "My Infosys role analyzing healthcare claims exposed me to the power of data-driven decisions. I grew passionate about predictive modeling and built portfolio projects demonstrating this shift—fraud detection, claim denial prediction, and a data quality tool using ML. These projects synthesize my domain knowledge (healthcare) with new ML skills, positioning me to drive more impact." Shows: purpose, growth trajectory.

Q37: What's your biggest professional weakness and how are you addressing it? A: "I'm relatively new to ML engineering. To address this, I've built multiple end-to-end projects with production tooling (Docker, MLflow, FastAPI, Streamlit). I'm pursuing certifications and learning from Kaggle competitions. I'm confident in statistical foundations and now solidifying ML systems design." Shows: self-awareness, proactive growth.

Q38: Tell us about your most impactful project. A: "My Claim Denial Risk Prediction pipeline. I engineered 10+ healthcare-specific features, trained and optimized a model for imbalanced data, deployed via FastAPI for real-time scoring and MLflow-tracked batch jobs, and containerized with Docker for reproducibility. Business impact: enables claims teams to intervene before denials, potentially saving millions in appeal costs." Quantify impact when possible.

Q39: How do you approach learning a new tool or technology? A: "I learn by doing. When I needed FastAPI and Docker, I researched documentation, built a small project, then applied them to my claim denial pipeline. For Streamlit, I built a data cleaning tool. I combine tutorials, documentation, and hands-on projects for deep learning." Shows: resourcefulness, practical approach.

Q40: Describe your ideal team environment. A: "I thrive in collaborative environments where data-driven decisions are valued. I appreciate mentorship, feedback, and opportunities to grow technically. I enjoy working cross-functionally—partnering with data engineers on pipelines, analysts on insights, and business stakeholders on impact. I prefer autonomy with clear goals." Shows: teamwork, ambition.

SECTION 3: DOMAIN-SPECIFIC QUESTIONS (Healthcare/Insurance)

Q41: What is a healthcare claim and what are common rejection reasons? A: A claim is a request for payment submitted by providers to insurers for patient services. Rejection reasons: incorrect member ID, out-of-network provider, non-covered service, missing documentation, policy exclusions, duplicate processing. Your denial prediction model learns these patterns from historical data.

Q42: What is a claims adjudication process? A: Adjudication is the workflow: claim submission → validation (basic eligibility checks) → processing (pricing, coordination of benefits) → determination (approval/denial) → payment or adjustment. Your projects touch this workflow—fraud detection screens for suspicious claims, denial prediction identifies likely rejections early.

Q43: What metrics matter most in healthcare claims operations? A: Cycle time (days to adjudicate), claim approval rate, denial rate (by reason), SLA compliance (processing deadlines), cost per claim, fraud detection rate. At Infosys, you tracked these operationally. ML adds: prediction accuracy, false positive rate (incorrect denials harm customers), model performance stability.

Q44: How would an ML model be used in a claims organization? A: 1) Fraud detection: real-time scoring prevents suspicious claims from entering system. 2) Denial prediction: flags high-risk claims for review before adjudication. 3) Cost prediction: estimates claim cost for budget planning. 4) Member profiling: identifies high-utilization members for intervention. 5) Prioritization: accelerates high-priority claims. Your projects cover #1 and #2.

SECTION 4: ROLE-SPECIFIC FINAL QUESTIONS

For Data Scientist Roles:

Q45: What's the difference between a data scientist and a data analyst? A: Data analysts focus on descriptive/diagnostic analytics (what happened, why), answering specific business questions with existing tools (SQL, BI). Data scientists add predictive/prescriptive analytics (what will happen, what should we do), building models and deploying them. You're transitioning from analyst (operational reporting at Infosys) to scientist (predictive models in projects).

Q46: How do you measure success of a data science project? A: Technical metrics: model accuracy, precision, recall, deployment success. Business metrics: revenue impact, cost savings, operational efficiency, decision quality improvement. "My fraud detection model catches 85% of fraud (recall) with 90% precision, reducing false positives that frustrate members." Always tie to business outcomes.

For ML Engineering Roles:

Q47: Explain the difference between ML Engineering and Data Science. A: Data science focuses on problem formulation, exploratory analysis, and model development. ML engineering focuses on taking trained models into production: scalability, reliability, monitoring, retraining pipelines. You're learning both—your projects show model building (DS) and production deployment (MLOps).

Q48: How would you design a real-time fraud detection system? A: Architecture: API endpoint receives claim → feature pipeline transforms raw claim into 10+ features → model predicts fraud score (milliseconds latency) → return risk level (high/medium/low). Scalability: horizontal scaling (multiple API instances behind load balancer), caching frequent predictions, batch pre-computation of features. Monitoring: track latency, accuracy, distribution shifts. Your FastAPI implementation is a good start.

For MLOps Roles:

Q49: What is MLOps and why is it important? A: MLOps applies DevOps principles to ML: automation, monitoring, reproducibility, scalability. It bridges data scientists (who build models) and engineers (who deploy systems). Important because: models decay over time (data drift), require continuous retraining, and are complex to maintain. Infrastructure as code, CI/CD pipelines, automated testing, and monitoring are central. Your Docker, MLflow, and batch scheduling demonstrate MLOps thinking.

Q50: Design an ML CI/CD pipeline for claim denial prediction. A: 1) Developers commit code → Git trigger. 2) Automated tests run (data validation, model tests). 3) Build container (Docker image). 4) Train model on latest data, log with MLflow. 5) Validation suite evaluates performance vs. baseline. 6) If passes, promote to staging, run integration tests. 7) If passes staging, deploy to production. 8) Monitor performance, alert on drift, trigger retraining if needed. This is enterprise ML deployment.

SECTION 5: QUICK REFERENCE - YOUR PROJECTS TALKING POINTS

Claim Denial Risk Prediction

- **Stakeholder value: Enables proactive intervention before denials occur**
- **ML approach: Logistic Regression optimized for imbalanced healthcare data**
- **Scale: Trained on millions of historical claims**
- **Features: 10+ healthcare-specific engineered features (claim amount, type, member age, provider history)**
- **Deployment: FastAPI real-time endpoint + MLflow batch scoring (daily automated runs)**
- **Operations: Docker containerization (Render cloud), experiment tracking**
- **Metrics: AUC-ROC for threshold optimization, F1-score for balanced evaluation**

Automated Fraud Claims Scoring

- **Business problem: Prevent fraudulent claims from entering system**
- **Approach: EDA-driven feature engineering capturing anomalous patterns**
- **Models: Logistic Regression, Random Forest, XGBoost (compared via MLflow)**
- **Scoring: Batch pipeline (daily automated scoring of all claims)**
- **Optimization: Model versioning with MLflow, reproducible results**
- **Impact: High recall (catch fraud) with acceptable precision tradeoff**

AI Data Cleaning Tool

- **Problem: Manual data preprocessing consumes 80% of analyst time**
- **Solution: Streamlit web app using Claude LLM API to auto-detect 15+ issue types**
- **Impact: 95%+ improvement in data quality, 80% reduction in manual time**
- **Deployment: Streamlit Cloud (serverless, free)**
- **Tech: Python, Streamlit, Anthropic Claude API integration**
- **Value: Demonstrates practical LLM application and full-stack data tool building**

SECTION 6: PREPARATION TIPS FOR VERBAL INTERVIEWS

1. **Use the STAR method: Situation-Task-Action-Result. For each question, structure your story.**
2. **Quantify everything: "Improved query performance 10x" beats "made it faster."**
3. **Practice out loud: Record yourself answering these questions. Verbal fluency matters.**
4. **Know your resume cold: Interviewers will deep-dive into every bullet point.**
5. **Prepare technical examples: Have 2-3 concrete examples from your Infosys and portfolio work ready.**
6. **Ask thoughtful questions back: Show genuine interest in the role and company culture.**

7. **Discuss tradeoffs: "Model A was more accurate, Model B faster—we chose B for real-time constraints." Maturity matters.**
8. **Be honest about gaps: "I'm new to MLOps but have built Docker/MLflow projects and I'm eager to deepen this." Honesty + agency.**

SECTION 7: YOUR STRENGTHS & WEAKNESSES FOR THIS TRANSITION

Strengths:

✅ Domain expertise: 3+ years healthcare claims (beats many junior DS candidates) ✅ End-to-end project experience: Full ML lifecycle (data → model → deployment) ✅ Production tooling knowledge: Docker, MLflow, FastAPI, Streamlit—not just Jupyter notebooks ✅ SQL mastery: Critical for data work; many weak here ✅ Mathematical foundation: MCA degree covering ML, algorithms, databases ✅ Communication: Translating between technical and business stakeholders

Areas to strengthen before interviews:

⚠ Advanced ML theory: Study regularization, hyperparameter tuning strategies, ensemble methods in depth ⚠ Advanced data structures: Hash tables, trees, graphs (may be tested in coding rounds) ⚠ Statistics depth: Probability distributions, hypothesis testing, Bayesian thinking ⚠ Big data tools: Spark, Hadoop (less critical for smaller companies, but good to mention) ⚠ A/B testing: Experiment design, statistical significance testing (increasingly important)