AIOps

# Table of Contents

1. Part 1: Problem Statement
2. Part 2: Standard ML Solution
3. Part 3: Ops Challenges
4. Part 4: AIOps with mlflow
5. Part 5: Analysis and Interpretation

# Part I Problem Statement

**Problem Statement: Predicting Quality of Service (QoS) Metrics for 5G Network Optimization**

**Overview**

Telecom operators face challenges in maintaining optimal Quality of Service (QoS) for their users due to varying signal conditions, network congestion, and application-specific demands. The provided dataset includes time-series data on signal strength, latency, bandwidth requirements, allocated bandwidth, and resource allocation for different application types. The goal is to build a machine learning model to predict QoS metrics such as **latency** and **resource allocation efficiency**, enabling proactive network management and improved user experience.

**Objective**

Develop a machine learning model to predict:

1. **Latency**: Estimate the latency for a given user, application type, and signal condition.
2. **Resource Allocation Efficiency**: Predict the ratio of allocated bandwidth to required bandwidth under varying network conditions.

The predicted values will help identify potential QoS degradation before it occurs, allowing for targeted network adjustments.

**Key Business Questions**

1. **Latency Prediction**: Can we accurately predict latency based on signal strength, application type, and bandwidth requirements?
2. **Resource Allocation Optimization**: Can we predict whether the allocated bandwidth will meet user demand efficiently?
3. **Proactive QoS Management**: Can these predictions highlight potential QoS bottlenecks for specific users or applications?

**Detailed Problem Description**

1. **Latency Prediction**:
   * Latency is a critical metric for QoS, particularly for applications like emergency services and online gaming that require real-time responses.
   * Variations in signal strength and bandwidth availability often cause unpredictable latency spikes.
   * The model will predict latency based on:
     + Signal strength.
     + Application type.
     + Required and allocated bandwidth.
2. **Resource Allocation Efficiency Prediction**:
   * Resource allocation efficiency measures how effectively network bandwidth meets application demands.
   * An efficiency ratio below a threshold indicates potential resource under-provisioning, affecting QoS.
   * The model will predict resource allocation efficiency using:
     + Application type.
     + Required bandwidth.
     + Allocated bandwidth.
     + Historical resource utilization patterns.
3. **Data Patterns and Dependencies**:
   * Signal strength below a certain threshold may correlate with higher latency and lower resource allocation efficiency.
   * Specific application types (e.g., video calls) may consistently experience resource constraints.
   * Temporal patterns in data (e.g., peak vs. non-peak hours) could impact QoS metrics.

**Dataset Fields and Their Roles**

|  |  |  |
| --- | --- | --- |
| Field Name | Description | Role in Prediction |
| Timestamp | Date and time of the record | Temporal analysis (peak/non-peak times) |
| User\_ID | Unique identifier for the user | Not directly used but useful for grouping if required |
| Application\_Type | Type of application (e.g., video call, gaming) | Key feature for predicting QoS metrics |
| Signal\_Strength | Signal strength in dBm | Predictor for latency and resource allocation efficiency |
| Latency | Observed latency in milliseconds | Target variable for latency prediction |
| Required\_Bandwidth | Bandwidth required by the application | Predictor for resource allocation efficiency |
| Allocated\_Bandwidth | Bandwidth allocated to the application | Predictor and part of target (efficiency calculation) |
| Resource\_Allocation | Efficiency of resource allocation (0-1 scale) | Target variable for efficiency prediction |

**Potential Insights**

1. **Latency Predictors**:
   * Identify key contributors to high latency for different application types (e.g., weak signal strength, high bandwidth demand).
2. **Bandwidth Efficiency Trends**:
   * Discover patterns of inefficient resource allocation and suggest improvements.
3. **Application-Specific QoS Challenges**:
   * Highlight applications that consistently experience QoS degradation and the factors contributing to it.

**Assumptions and Limitations**

1. **Assumptions**:
   * Data is clean, consistent, and representative of the overall network usage patterns.
   * Signal strength and bandwidth metrics are accurately recorded in real-time.
2. **Limitations**:
   * Geographic and user-specific metadata are not included, which could limit personalized predictions.
   * Predictions will be based solely on the provided features without additional contextual data (e.g., network topology).

# Part 2: Proposed Solution

**Proposed Approach**

1. **Exploratory Data Analysis (EDA)**:
   * Identify trends, correlations, and anomalies in key features (e.g., signal strength, latency).
   * Analyze the distribution of QoS metrics by application type and signal strength ranges.
2. **Feature Engineering**:
   * Create derived features:
     + **Bandwidth Efficiency**: Allocated Bandwidth/Required Bandwidth\text{Allocated Bandwidth} / \text{Required Bandwidth}Allocated Bandwidth/Required Bandwidth
     + **Signal Quality Category**: Group signal strength into ranges (e.g., strong, moderate, weak).
     + **Temporal Features**: Extract time-related features (e.g., hour of day, day of week).
3. **Machine Learning Models**:
   * **Supervised Learning**:
     + Target Variables:
       1. **Latency**: A regression task to predict latency in milliseconds.
       2. **Resource Allocation Efficiency**: A regression task to predict bandwidth efficiency as a ratio.
     + Input Features:
       1. Signal strength, application type, required bandwidth, allocated bandwidth, and temporal features.
   * **Candidate Models**:
     + Linear Regression, Random Forest, Gradient Boosting (e.g., XGBoost, LightGBM), and Neural Networks.
   * Evaluate models using metrics like MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error).
4. **Validation and Testing**:
   * Split data into training, validation, and test sets.
   * Use cross-validation to ensure robust performance metrics.

# Part 3: Current Challenges for Ops

**Instructions:**

**1. Role-Playing**

* Take on a role (e.g., Data Scientist, Operations Engineer, or Stakeholder).
* Discuss challenges your role faces in ML workflows (e.g., tracking, deployment, monitoring). Relate to current use case.
* Share key pain points and their impact.

**2. Scenario Exploration**

* Scenario: A deployed model’s performance degrades after two weeks.
* Discuss:
  + Possible causes of degradation.
  + Metrics needed to investigate.
  + Solutions to prevent this in the future.

**3. Pain Point Mapping**

* List the steps in the ML workflow (e.g., data prep, training, deployment).
* Identify pain points for each step (e.g., manual processes, lack of tracking).
* Share how addressing these could improve outcomes.

**4. Challenge Prioritization**

* Review and rank the top three challenges from discussions.
* Explain why they’re critical and suggest possible solutions.
* Discuss tools or practices (e.g., tracking systems, automation) to resolve them.

1. **Relate Challenges to Your Experience**:
   * Reflect on any past machine learning projects you’ve worked on. Think about the pain points during model development, deployment, or monitoring.
   * Discuss examples of manual processes or inefficiencies that could benefit from automation or tracking.
2. **Break Down the Workflow**:
   * Think about the end-to-end machine learning lifecycle, from data preparation to deployment. Identify areas where tracking, monitoring, or scaling were difficult.
3. **Focus on Outcomes and Impact**:
   * Consider what happens when models fail in production.
   * Discuss scenarios where issues like lack of reproducibility, inconsistent results, or system bottlenecks negatively impacted project outcomes.
4. **Challenge Current Practices**:
   * Explore if the current methods provide enough transparency, scalability, and collaboration.
   * Think critically about what information was missing when troubleshooting or improving a model.
5. **Prioritize Scalability and Maintenance**:
   * Imagine your solution being deployed in a real-world, high-scale environment. Discuss potential risks or challenges in maintaining the system.

## Hint (Examples):

**Experimentation and Tracking**

1. *"How do you currently keep track of the parameters, datasets, and metrics used during training? Have you ever struggled to reproduce a result from a previous experiment?"*
2. *"What happens when someone on the team modifies a model or pipeline? How do you ensure these changes are tracked and logged?"*

**Model Versioning and Governance**

1. *"Imagine deploying an updated model to production and it performs worse than the previous version. How would you roll back to the earlier version? Do you currently have a system for managing multiple versions?"*
2. *"How do you ensure that every version of a model is linked to the exact dataset and code used to train it?"*

**Monitoring Operational Metrics**

1. *"Once your model is deployed, what operational metrics would you track to ensure smooth functioning? Would you monitor things like CPU usage, memory, or latency?"*
2. *"How do you know if your deployed model is drifting in terms of accuracy or encountering degraded performance over time?"*

**Collaboration and Transparency**

1. *"How do team members currently share experiment results, model artifacts, or deployment processes? Is it easy to collaborate across roles?"*
2. *"Have you ever faced issues where one team member's changes to a pipeline broke something for someone else? How could better transparency have helped?"*

**Automation and Workflow Management**

1. *"What aspects of your current machine learning workflow are repetitive or manual? How could automating these processes help improve efficiency?"*
2. *"Imagine retraining and deploying a model every week with new data. What would make this process more streamlined and reliable?"*

**Scaling and Real-Time Use**

1. *"What happens when your model needs to handle 10x more data or user queries? Have you thought about how resource constraints might affect scalability?"*
2. *"If you needed to deploy multiple models for different tasks (e.g., latency and efficiency), how would you manage dependencies and scaling?"*

**Success Metrics**

1. *"How do you currently measure the success of your deployed models? Are these metrics tracked consistently across experiments and in production?"*
2. *"Have you ever struggled to define clear performance metrics for a model? How could a standardized system help?"*