InternalAssessmentTest2–November 2025

Sub:	DATAENGINEERING AND MLOPS					Sub Code:	BAD714C	Branch:	AIML/CSE AIML	
Date:		Duration:	90min	Max Marks:	50	Sem/Sec:	VII/A,B&C		OBE	
Answer any FIVE FULL Questions								**MARKS**	**CO**	**RBT**
1	With a suitable use case , explain how an application moves from development to production environment?							10	CO4	L3
2	In MLOps, Explain how CI/CD pipeline works? How do artifacts, testing pipelines, and automation contribute to a smooth transition from development to production?							10	CO4	L2
3	Explain Model Risk Evaluation. Discuss the major sources of risk that can cause an ML model to fail in real-world environments.							10	CO4	L2
4	Explain what is model degradation? Describe how monitoring ground truth, logging, and online evaluation help identify when a machine learning model needs to be retrained?							10	CO5	L2
5a	Mention any two causes of data drift with examples.Explain any one drift detection technique used in practice.							5	CO5	L2
5b	What is a feedback loop in machine learning? How does logging support continuous monitoring? When a model should be retrained based on monitoring signals?							5	CO5	L2
6	A recommendation model has not been retrained for 10 months and user behavior has changed.Describe how you would apply a feedback loop mechanism to decide whether the model must be retrained now?							10	CO5	L3

USN | | | | | | | | | | |

Sub:	DATAENGINEERING AND MLOPS					Sub Code:	BAD714C
Date:	**2-12-25**	Duration:	**90 minutes**	Max Marks:	**50**	Sem/Sec:3 A,B,CSE-AIML	

SCHEME AND SOLUTIONS

1		With a suitable use case, explain how an application moves from development to production environment? The lifecycle of an application can be explained using a practical scenario such as a **fraud detection system** used by a bank. In the **development phase**, data scientists collect historical transaction data and clean it to remove errors or missing values. Features such as transaction amount, device type, and location are extracted. A machine learning model (e.g., XGBoost) is trained to classify transactions as fraud or not fraud. The model is validated using test datasets to ensure good accuracy. Next, the application enters the **pre-production phase**, where the model is prepared for deployment. It may be converted into a format compatible with the production environment, optimized for latency, and tested under realistic load conditions. Quality assurance checks such as handling extreme values, missing values, and stress testing are performed. Secure data access is configured so the model can read live transaction data. After successful validation, the system is **deployed to production**. Initially, it may run in **shadow mode**, where the new model receives real-time data but does not influence decisions. This allows comparison with the existing system. Once stable, the new model becomes the active version and starts blocking or flagging suspicious transactions. In the next phase, **continuous monitoring** is performed. Engineers monitor system metrics like CPU usage, latency, and throughput. They also track model performance to detect data drift—e.g., if user transaction patterns change due to new payment apps. Alerts are raised when performance degrades. Finally, the **feedback loop** begins. Newly collected labeled data is used to retrain the model. The updated model is evaluated, tested, and deployed again as a newer version. This cycle repeats to ensure the fraud detection system remains accurate and adapts to emerging fraud patterns.
2	a)	In MLOps, Explain how CI/CD pipeline works, How do artifacts, testing pipelines, and automation contribute to a smooth transition from development to production? In MLOps, the CI/CD pipeline (Continuous Integration and Continuous Deployment) is designed to automate the movement of machine learning models from development to production in a reliable and reproducible manner. Unlike traditional software CI/CD, MLOps handles not only code but also data, models, and training pipelines. This ensures consistent delivery of high-quality ML systems. a) Continuous Integration (CI) This stage focuses on automatically building, validating, and versioning ML components whenever changes occur in:

- Model code
- Data preprocessing scripts
- Training configurations
- Datasets

Whenever a developer commits changes, the CI pipeline triggers:

A. **Code linting & static checks**
B. **Unit testing for data and model functions**
C. **Data validation tests**
D. **Model training pipeline execution**
E. **Model evaluation and comparison with baseline**

If the new model performs better than the current baseline, an artifact is generated and pushed to the model registry.

b) Continuous Deployment (CD)

CD automates deployment to staging or production environments.

Key steps include:

A. **Fetching the best model artifact from the registry**
B. **Containerization (e.g., Docker image creation)**
C. **Integration testing in staging**
D. **Canary rollout / A-B testing / shadow deployment**
E. **Full production deployment if tests pass**

Modern CD ensures that every validated model version can be deployed with minimal risk using automated workflows.

Role of Artifacts in CI/CD

Artifacts are packaged versions of:

- Trained ML models
- Preprocessing pipelines
- Environment dependencies (Docker images)
- Metadata (metrics, parameters, version info)

Why artifacts matter:

A. **Reproducibility:**
   The same artifact deployed in staging is guaranteed to behave the same in production.
B. **Version control:**
   Every model version is tracked, allowing rollbacks if needed.
C. **Environment consistency:**
   Helps avoid issues like “works on my machine.”
D. **Auditability:**
   Artifacts store metadata needed for debugging, monitoring, and compliance.

Thus, artifacts act as the single source of truth across development, testing, and deployment.

Role of Testing Pipelines

Testing pipelines ensure that any change to code, data, or model does not break the system.

Types of tests in MLOps CI/CD:

- **Unit tests:** Validate feature engineering, transformations, and algorithms.
- **Data validation tests:** Schema checks, missing values, drift checks.
- **Model quality tests:** Accuracy, precision/recall, AUC comparisons with baseline.
- **Stress and load tests:** Ensure that inference meets latency requirements.
- **Integration tests:** Confirm the model interacts properly with APIs and databases.
- **Fairness and bias tests:** Check for model stability across subgroups.

Why testing is essential:

- Ensures reliability of models under real-world data
- Prevents regressions when new models are trained
- Validates that performance meets business thresholds
- Reduces risk during deployment

Testing serves as the safety gate before a model moves to production.

Role of Automation in MLOps CI/CD

Automation ties the entire CI/CD process together.

How automation ensures smooth transition:

A. **Eliminates manual errors:**
   Automated pipelines ensure consistent and error-free execution of workflows.
B. **Reduces deployment time:**
   Deployments that used to take weeks can now occur in minutes.
C. **Enforces standards:**
   Every model version passes through the same validation, testing, and quality checks.
D. **Supports continuous retraining:**
   Pipelines can automatically retrain models when new data arrives or drift is detected.
E. **Enables scalable ML systems:**
   Multiple models can move through development → production without human bottlenecks.

Automation ensures that ML models remain up-to-date, robust, and aligned with changing business needs.

3		**Explain Model Risk Evaluation. Discuss the major sources of risk that can cause an ML model to fail in real-world environments.** Model risk evaluation refers to the process of identifying and assessing all factors that can cause an ML model to behave incorrectly once deployed in real-world environments. Since ML models approximate reality and rely heavily on data, they carry unique risks that must be understood before production. The goal of model risk evaluation is to ensure that the model is safe, reliable, and does not

		create financial, operational, legal, or ethical harm. A major source of risk is **errors or bugs in the model design or implementation**. Mistakes in data preprocessing, feature engineering, training logic, or evaluation metrics can lead to flawed predictions. Another important source is **runtime or environment-related issues**, such as differences between the development and production environments, library incompatibilities, model conversion errors, or containerization issues that make the deployed model behave unexpectedly. A significant category of risk comes from **low-quality or biased training data**. If data is noisy, incomplete, outdated, or not representative of the real population, the model will fail when exposed to new scenarios. Related to this is the risk of a high **difference between training data and production data**. When the real world changes (data drift or concept drift), the model's accuracy drops, sometimes sharply. Another risk arises from **expected model errors that have greater consequences than anticipated**. Even a small misclassification rate can be dangerous in sensitive domains like healthcare, credit scoring, or fraud detection. There is also a risk of **misuse or misinterpretation**, where users depend too heavily on the model or apply it outside its intended scope. **Adversarial attacks** present another threat. Attackers can manipulate inputs to force incorrect predictions or attempt to poison the training data. ML models can also unintentionally **leak sensitive information**, especially if trained on personal data. Finally, risks are amplified in environments that involve **rapid changes**, **broad deployment**, or **interactions between multiple models**. When many models depend on each other, unexpected behavior in one can cascade across systems. Overall, model risk evaluation ensures that ML systems are tested thoroughly, validated carefully, and monitored continuously so failures are prevented and real-world harm is minimized.
4		**Explain model degradation and describe how monitoring ground truth, logging, and online evaluation help identify when a machine learning model needs to be retrained.** Model degradation refers to the decline in a model's performance after deployment due to changes in real-world data, user behavior, environment, or unseen patterns. Because machine learning models assume that future data resembles training data, any shift—such as new fraud patterns, different customer behavior, or seasonal changes—causes prediction accuracy to drop. To detect degradation early, MLOps systems rely on three major mechanisms: **ground truth monitoring, logging, and online evaluation**. To detect degradation, MLOps uses **ground truth monitoring**, which compares the model's predictions with actual outcomes once labels become available. If the difference between the original training performance and current performance becomes large, it indicates the model is no longer aligned with real-world data and needs retraining. **Logging** also plays a major role. By recording inputs, outputs, feature values, and system actions from the production environment, engineers can detect data drift. If the distribution of incoming data begins to differ from the training distribution, it signals that the model might soon degrade even before ground truth arrives. **Online evaluation** further helps by comparing the production model with a new model in real time

		sing methods like shadow testing or A/B testing. If the challenger model performs better on live data, it shows the existing model is degraded and must be retrained or replaced. Together, ground truth monitoring, logging, and online evaluation provide continuous insight into model behavior and help identify the right moment to retrain the machine learning model.
5	a	**Mention any two causes of data drift with examples.Explain any one drift detection technique used in practice.** **Two causes of data drift with examples** • **Sample Selection Bias** This happens when the training data is not representative of the real-world population. *Example:* A discount recommendation model trained only on data from premium customers will fail when used for all customers. • **Non-stationary Environment (Concept Drift)** Real-world patterns change over time, causing relationships between features and labels to shift. *Example:* A fraud detection model trained before the rise of UPI payments will miss new fraud patterns using those methods. **Example for drift detection technique** **Univariate Statistical Tests** Each feature in the production data is compared with the same feature from the training data using tests like the Kolmogorov–Smirnov test (for continuous features) or Chi-square test (for categorical features). If the distribution difference is statistically significant, it indicates drift. This helps detect which features are changing and may affect model accuracy.
5	b	**What is a feedback loop in machine learning? How does logging support continuous monitoring? When a model should be retrained based on monitoring signals?** **What is a feedback loop in machine learning?** A feedback loop is the process where production data, predictions, and later ground truth are continuously sent back to the model development stage. This helps evaluate performance, detect degradation, and prepare improved models. It ensures the system keeps learning from new patterns rather than relying on outdated behavior. **How logging supports continuous monitoring** Logging records all important production information — inputs, model predictions, timestamps, system actions, feature values, and explanations. By analyzing logs, teams can detect data drift, monitor model accuracy, observe unusual patterns, and identify failures early. Logging provides the evidence needed to understand when and why the model begins to degrade. **A model should be retrained when:** ● Ground truth performance drops beyond a set threshold ● Statistical drift is detected in important features

		● Online evaluation shows challenger models performing better ● Logging indicates consistent changes in data distribution ● Business metrics (e.g., fraud loss, incorrect approvals) worsen Retraining is triggered when these signals show the model no longer reflects real-world data or business needs. **What is a feedback loop in machine learning?** A feedback loop is the process where production data, predictions, and later ground truth are continuously sent back to the model development stage. This helps evaluate performance, detect degradation, and prepare improved models. It ensures the system keeps learning from new patterns rather than relying on outdated behavior. **How logging supports continuous monitoring** Logging records all important production information — inputs, model predictions, timestamps, system actions, feature values, and explanations. By analyzing logs, teams can detect data drift, monitor model accuracy, observe unusual patterns, and identify failures early. Logging provides the evidence needed to understand when and why the model begins to degrade. **A model should be retrained when:** • Ground truth performance drops beyond a set threshold • Statistical drift is detected in important features • Online evaluation shows challenger models performing better • Logging indicates consistent changes in data distribution • Business metrics (e.g., fraud loss, incorrect approvals) worsen Retraining is triggered when these signals show the model no longer reflects real-world data or business needs.
	b	**A recommendation model has not been retrained for 10 months and user behavior has changed.** Describe how you would apply a feedback loop mechanism to decide whether the model must be retrained now? To apply a feedback loop for a recommendation model that has not been retrained for 10 months, the first step is to collect recent production data, including model inputs, predictions, and actual user actions such as clicks, views, and purchases. Using this ground truth, the model's performance metrics—like click-through rate, ranking accuracy, and conversion rate—are recalculated and compared with the metrics the model achieved during its last training. A noticeable drop indicates potential degradation. Next, logging data is analyzed to check whether user behavior has drifted. This includes changes in browsing sessions, new content categories, shifts in item popularity, or different time-of-day usage patterns. If the feature distributions in logs differ significantly from the training distributions, this confirms that the recommendation environment has evolved. As part of the feedback loop, a new candidate model is trained using the latest data and evaluated offline. The current production model and the new model are then compared through online evaluation, such as shadow testing or A/B testing, to observe their performance on real traffic. If the challenger model consistently gives higher engagement or better ranking quality, it shows the current model is outdated. Finally, using the combined signals—ground truth performance decline, detected data drift in logs, and online evaluation results—the feedback loop determines whether retraining is needed. If all three indicators point to degradation, the model should be retrained immediately and the improved version deployed.

CI **CCI** **HOD**

----------All the Best----------