# Data Science Challenge

## Part 1. Machine Learning System Design Task

### Diagrama de arquitectura del sistema que ilustra el end-to-end ML pipeline.

Diagrama

Descripción generada automáticamente

### Explicación detallada de cada componente de la arquitectura.

1. **Pipelines de Airflow**:
   * **Apache Airflow** orquesta el pipeline de datos para poblar las tablas de **Snowflake**.
   * Gestiona la programación para el reentrenamiento, las verificaciones de calidad de datos y las predicciones por lotes.
   * DAGs por país aseguran la separación y mantenibilidad.
   * Para controlar la **Calidad de los Datos** se maneja con un paso de validación dentro del pipeline de Airflow. Este paso verifica datos faltantes, valores atípicos y desviaciones de datos. En caso de errores, Airflow activa notificaciones para tomar acciones correctivas.
2. **Snowflake**:
   * Almacén central de datos para el almacenamiento de características y la ingesta de nuevos datos.
3. **Docker & AWS ECS** **& AWS ECR**:
   * **AWS ECS** y **ECR**: Los modelos se encapsulan en contenedores de Docker, se almacenan en **ECR** y se despliegan en **clusters de ECS**. Con **ECS** se gestiona el escalado automático y el balanceo de carga entre los diferentes países y modelos.
   * **5 modelos por País**: Cada país tiene un entorno aislado donde los modelos se alojan como microservicios independientes.
4. **MLflow**:
   * Seguimiento de modelos, control de versiones y registro de métricas.
   * Proporciona transparencia y control sobre la experimentación de modelos y los modelos en producción.
5. **AWS Batch**:
   * Los trabajos de inferencia por lotes distribuidos se ejecutan diariamente para los 50 millones de usuarios.
6. **Bitbucket Pipelines**:
   * CI/CD para la prueba, construcción e implementación automática de modelos.
7. **S3**:
   * **S3** para el registro y almacenamiento de predicciones del modelo y el seguimiento de datos a lo largo del tiempo.

### Estrategias de experimentación, reentrenamiento y seguimiento de métricas.

* 1. **Estrategia de Experimentación:**
  + **MLflow** almacena todos los datos de los experimentos del modelo, incluidos los hiperparámetros, resultados, artefactos, y versiones de modelos.
  + **Bitbucket Pipelines**: Para la integración continua (CI), automatiza las pruebas de nuevas características/modelos. Los nuevos cambios que se suban a la rama principal activan pasos de pipeline que entrenan, evalúan y comparan las métricas del modelo con versiones existentes.
  + **Despliegue canario**: Despliega nuevas versiones de los modelos de manera gradual (por ejemplo, con el 10% del tráfico), mide el rendimiento y aumenta gradualmente.
  1. **Estrategia de Reentrenamiento:**
  + Se puede configurar un pipeline de reentrenamiento en **Airflow**, que se ejecute periódicamente (por ejemplo, semanal o mensualmente) en función del tiempo o cuando las métricas de rendimiento disminuyan (gestionadas a través del **Data Drift**).
  + **Snowflake** almacena los datos más recientes etiquetados para el entrenamiento, y **Airflow** orquesta los trabajos de reentrenamiento en **AWS Batch**.
  + Los modelos reentrenados se almacenan con **MLflow**, y el control de versiones del modelo garantiza retrocesos si es necesario.
  1. **Seguimiento de Métricas y Data Drift:**
  + Realizar un seguimiento de las métricas de rendimiento del modelo (precisión, recall, AUC, etc.) usando **MLflow**. Registrar cada ejecución de predicción e implementar la detección de **Data Drift** comparando las predicciones del modelo a lo largo del tiempo con los resultados reales (etiquetas).
  + Las métricas de rendimiento en tiempo real se registran y se muestran en **Grafana**, que recibe sus datos de **Prometheus**. **Grafana** mostraría tableros en tiempo real para los modelos de cada país, facilitando la detección de tendencias, degradación del rendimiento y derivaciones de datos.
  + Usa **Airflow** para programar verificaciones periódicas de **Data Drift** mediante la medición de cambios en la distribución de las características de entrada utilizando pruebas estadísticas (por ejemplo, **KS-Test** o **Jensen-Shannon Distance**).
  + Almacenar registros, predicciones y métricas en **S3** para almacenamiento persistente.

### Enfoque para manejar problemas de calidad de datos y deriva de datos.

* 1. **Manejo de Problemas de Calidad de Datos:**
  + Implementar pasos de validación de datos en **Airflow** para verificar anomalías en los datos, valores faltantes y formatos inconsistentes.
  + Registrar los resultados de la validación de datos en **S3** y genera alertas si surgen problemas significativos.
* Implementar reintentos y modelos de respaldo cuando las verificaciones de calidad de datos fallan.
  1. **Manejo de Data Drift:**
  + Monitorear en tiempo real las entradas de datos y las salidas de predicción utilizando pruebas estadísticas de para detectar **Data Drift** (por ejemplo, **KS-Test** o **Jensen-Shannon Distance)**.
  + Al detectar **Data Drift,** se generan alertas y se inicia el reentrenamiento a través de **Airflow**. Realizar seguimiento de la distribución de características clave y las predicciones a lo largo del tiempo y compáralas con los datos de entrenamiento originales.

### Consideraciones de escalabilidad y soluciones propuestas.

* **AWS Batch**. Para la inferencia por lotes a gran escala, utiliza **AWS Batch** para distribuir los trabajos de inferencia de manera eficiente en múltiples instancias de cómputo. Esto es escalable para manejar predicciones para 50 millones de usuarios. El sistema escala según la demanda de trabajo y los recursos disponibles.
* **Snowflake**. Proporciona almacenamiento de datos escalable y capacidades de consulta eficientes, garantizando una recuperación de datos rápida para 50 millones de usuarios en diferentes países.
* **Escalado automático con AWS ECS**. Los clusters de **AWS ECS** se escalarán automáticamente según el número de solicitudes y la utilización de recursos para garantizar un rendimiento fluido incluso bajo cargas pesadas.