MÓDULO 22: CHECKLIST FINAL

Verificación del Portafolio

Guía MLOps v2.0 | DuqueOM | Noviembre 2025

MÓDULO 22: Checklist Final

Verificación del Portafolio

"La calidad se verifica, no se asume."

Nivel	Duración
Referencia	1 hora

Objetivo

Lista de verificación completa para asegurar que tu portafolio MLOps está listo para presentar.

Checklist Pre-Release

1. Repositorio y Estructura

```
## Estructura del Repo
- [ ] README.md completo con badges
- [ ] LICENSE presente (MIT recomendado)
- [ ] .gitignore apropiado para Python/ML
- [ ] Estructura de carpetas profesional
- [ ] Cada proyecto tiene su propio README

## Versionado
- [ ] Commits con formato convencional
- [ ] Branches organizados (main, develop)
- [ ] Tags para releases
- [ ] CHANGELOG.md actualizado
```

2. Código y Calidad

```
## Código
- [ ] Código modular (src/proyecto/)
- [ ] Configuración con Pydantic (config.py)
- [ ] Logging implementado
- [ ] Type hints en funciones principales
- [ ] Docstrings en clases y funciones públicas

## Estilo
- [ ] Formateado con Black
- [ ] Imports ordenados con isort
- [ ] Sin errores críticos de flake8
- [ ] Sin warnings de mypy (o justificados)
```

3. Datos

```
## Versionado de Datos
- [ ] DVC inicializado
- [ ] Datasets versionados
- [ ] Remote configurado (local o cloud)
- [ ] .dvc files commiteados
- [ ] dvc.yaml con pipeline definido

## Documentación de Datos
- [ ] Data Card presente
- [ ] Descripción de features
- [ ] Distribución de target documentada
```

4. Modelo y Pipeline

```
## Pipeline ML
- [ ] Pipeline sklearn unificado
- [ ] Preprocessor incluido en pipeline
- [ ] Modelo serializable con joblib
- [ ] Reproducibilidad verificada (seeds)

## Tracking
- [ ] MLflow configurado
- [ ] Parámetros logueados
- [ ] Métricas logueadas
- [ ] Modelos registrados como artefactos
```

5. Testing

```
## Tests
- [ ] tests/conftest.py con fixtures
- [ ] Tests unitarios (test_*.py)
- [ ] Tests de integración
- [ ] Tests de API
- [ ] Coverage ≥ 70%

## Ejecución
- [ ] pytest ejecuta sin errores
- [ ] pytest --cov reporta coverage
- [ ] Tests son independientes (no orden)
- [ ] Tests rápidos (<30s total)
```

6. CI/CD

```
## GitHub Actions
- [ ] .github/workflows/ci.yml presente
- [ ] Jobs: lint, test, build
- [ ] Matrix testing (Python 3.11, 3.12)
- [ ] Badge de CI en README
- [ ] Pipeline pasa en verde

## Seguridad
- [ ] Bandit scan sin HIGH severity
- [ ] Gitleaks configurado
- [ ] No secrets en código
- [ ] .env.example presente (no .env)
```

7. Docker

```
## Dockerfile
- [ ] Multi-stage build
- [ ] Usuario no-root
- [ ] .dockerignore presente
- [ ] Imagen < 1GB (idealmente < 500MB)
- [ ] HEALTHCHECK configurado

## Compose
- [ ] docker-compose.yml funcional
- [ ] Servicios se levantan correctamente
- [ ] Health checks pasan
- [ ] Puertos documentados
```

8. API

```
## FastAPI
- [ ] /health endpoint
- [ ] /predict endpoint
- [ ] Validación con Pydantic
- [ ] Documentación en /docs
- [ ] CORS configurado

## Funcionamiento
- [ ] API responde correctamente
- [ ] Predicciones son válidas
- [ ] Errores tienen mensajes claros
- [ ] Latencia < 100ms
```

9. Documentación

```
## README Principal
- [ ] Descripción clara del proyecto
- [ ] Quick Start funcional
- [ ] Instrucciones de instalación
- [ ] Ejemplos de uso
- [ ] Badges de CI, coverage, etc.

## Model Card
- [ ] Model Details completos
- [ ] Intended Use documentado
- [ ] Métricas de performance
- [ ] Limitaciones explícitas
- [ ] Consideraciones éticas
```

10. Demo

```
## Video Demo
- [ ] Duración 3-5 minutos
- [ ] Introducción del problema
- [ ] Muestra estructura del código
- [ ] Demo en vivo funcionando
- [ ] CI/CD pipeline visible
- [ ] Cierre con call-to-action
```

Verificación Final

Comandos de Validación

```
# 1. Clone limpio
cd /tmp
git clone https://github.com/USUARIO/REPO.git
cd REPO

# 2. Instalar y testear
pip install -e ".[dev]"
pytest tests/ -v --cov=src

# 3. Lint
black --check src/
flake8 src/ --select=E9,F63,F7,F82

# 4. Docker
docker build -t test-image .
docker run --rm -d -p 8000:8000 test-image
sleep 10
curl http://localhost:8000/health

# 5. DVC
dvc status
dvc repro --dry

# 6. MLflow
mlflow ui --port 5000 &
# Verificar UI en http://localhost:5000
```

Criterios de Aprobación

Criterio	Mínimo	Ideal
Coverage	70%	85%+
Latencia API	<200ms	<50ms
Tamaño Docker	<1GB	<500MB
Tests	10+	30+
CI tiempo	<10min	<5min

Checklist por Proyecto

BankChurn-Predictor

```
- [ ] src/bankchurn/ modular
- [ ] VotingClassifier implementado
- [ ] Calibración de probabilidades
- [ ] Tests de fairness
- [ ] API /predict funcional
- [ ] Model Card con métricas
```

CarVision-Market-Intelligence

```
- [ ] FeatureEngineer centralizado
- [ ] Pipeline [features, pre, model]
- [ ] Streamlit dashboard
- [ ] FastAPI backend
- [ ] Data leakage prevenido
- [ ] Bootstrap confidence intervals
```

TelecomAI-Customer-Intelligence

```
- [ ] Clasificación multi-estrategia
- [ ] VotingClassifier configurado
- [ ] Pipeline end-to-end
- [ ] Tests de integración
- [ ] API documentada
```

Pre-Push Checklist

Antes de cada push importante:

```
## Quick Check
- [ ] `pytest` pasa
- [ ] `black --check .` pasa
- [ ] `flake8 src/` sin errores críticos
- [ ] `docker build` funciona
- [ ] Commit message es descriptivo
- [ ] No hay archivos sensibles staged
```

Scorecard de Portafolio

Auto-evaluación (0-10 por ítem)

Categoría	Puntuación	Notas
Código	/10	Modularidad, estilo, documentación
Testing	/10	Coverage, variedad de tests
CI/CD	/10	Automatización, velocidad
Docker	/10	Optimización, seguridad
API	/10	Funcionalidad, documentación
Datos	/10	Versionado, documentación
ML	/10	Pipeline, métricas, tracking
Docs	/10	README, Model Cards
Demo	/10	Video, presentación
Profesionalismo	/10	Consistencia, atención al detalle

Total: /100

Niveles

Puntuación	Nivel
90-100	Excepcional - Listo para entrevistas senior
80-89	Excelente - Muy competitivo
70-79	Bueno - Sólido para aplicar
60-69	Aceptable - Necesita mejoras menores
<60	En progreso - Continuar trabajando

Acciones Post-Checklist

Si pasas todo:

1. Publicar en GitHub
2. Añadir a LinkedIn
3. Incluir en CV
4. Compartir en comunidades

Si faltan items:

1. Priorizar items críticos (tests, CI, docs)
2. Crear issues para items faltantes
3. Planificar sprints de mejora
4. Re-evaluar en 1 semana

Navegación

◀ Anterior	Índice	▶ Siguiente
21_PLANTILLAS.md	Índice	23_RECURSOS.md

Módulo 22 Completado