MÓDULO 21: PLANTILLAS REUTILIZABLES

Templates para Proyectos ML

Guía MLOps v2.0 | DuqueOM | Noviembre 2025

MÓDULO 21: Plantillas Reutilizables

Templates para Proyectos ML

"No reinventes la rueda, usa plantillas."

Nivel	Duración
Referencia	Consulta

Objetivo

Proporcionar templates listos para usar que aceleren el desarrollo de tu portafolio MLOps.

1. Template de README.md

~~~
#  [Nombre del Proyecto]

[![CI Pipeline](https://github.com/USUARIO/REPO/actions/workflows/ci.yml/badge.svg)](https://github.com/USUARIO/REPO/actions/workflows/ci.yml)
[![Coverage](https://img.shields.io/badge/Coverage-XX%25-brightgreen.svg)](reports/)
[![Python](https://img.shields.io/badge/Python-3.11%20%7C%203.12-blue.svg)](https://python.org)
[![Docker](https://img.shields.io/badge/Docker-Ready-2496ED.svg?logo=docker)](Dockerfile)
[![License](https://img.shields.io/badge/License-MIT-blue.svg)](LICENSE)

> **Breve descripción del proyecto en una línea.**

---

##  Tabla de Contenidos
- [Descripción](#descripción)
- [Características](#características)
- [Quick Start](#quick-start)
- [Instalación](#instalación)
- [Uso](#uso)
- [Arquitectura](#arquitectura)
- [Testing](#testing)
- [API Reference](#api-reference)
- [Contribución](#contribución)
- [Licencia](#licencia)

---

##  Descripción

[2-3 párrafos describiendo el problema que resuelve, el enfoque técnico, y los resultados principales]

### Métricas del Modelo

| Métrica         | Valor   |
|-----------------|---------|
| **AUC-ROC**     | 0.XX    |
| **F1 Score**    | 0.XX    |
| **Latency P95** | <XXms   |

---

##  Características

-  Pipeline de ML reproducible con sklearn
-  API REST con FastAPI
-  Tracking de experimentos con MLflow
-  Contenerización con Docker
-  CI/CD con GitHub Actions
-  Tests con >70% coverage

---

##  Quick Start

```bash
# Clonar repositorio
git clone https://github.com/USUARIO/REPO.git
cd REPO

# Opción 1: Docker (recomendado)
docker-compose up -d

# Opción 2: Local
pip install -e ".[dev]"
python main.py

# Probar API
curl http://localhost:8000/health
~~~

Instalación

Requisitos

- Python 3.11+
- Docker (opcional)
- Make (opcional)

Instalación Local

```
# Crear entorno virtual
python -m venv .venv
source .venv/bin/activate  # Linux/Mac
# .venv\Scripts\activate   # Windows

# Instalar dependencias
pip install -e ".[dev]"
```

Instalación con Docker

```
docker build -t proyecto:latest .
docker run -p 8000:8000 proyecto:latest
```

Uso

Entrenamiento

```
python main.py train --config configs/config.yaml
```

Predicción

```
python main.py predict --input data/sample.csv --output predictions.csv
```

API

```python
import requests

response = requests.post(
    "http://localhost:8000/predict",
    json={"feature1": 1.0, "feature2": "A"}
)
print(response.json())
```

🏗 Arquitectura

```
proyecto/
├── src/proyecto/          # Código fuente
│   ├── config.py          # Configuración Pydantic
│   ├── training.py        # Pipeline de entrenamiento
│   ├── prediction.py      # Lógica de inferencia
│   └── evaluation.py      # Métricas
├── app/                   # API FastAPI
├── tests/                 # Tests pytest
├── configs/               # Archivos de configuración
├── models/                # Modelos entrenados
└── Dockerfile
```

Testing

```
# Ejecutar todos los tests
pytest tests/ -v

# Con coverage
pytest --cov=src --cov-report=html

# Solo tests rápidos
pytest -m "not slow"
```

API Reference

GET /health

Health check del servicio.

Response:

```json
{"status": "healthy", "version": "1.0.0"}
```

POST /predict

Hacer una predicción.

Request:

```json
{"feature1": 1.0, "feature2": "A"}
```

Response:

```json
{"prediction": 1, "probability": 0.85}
```

Contribución

1. Fork el repositorio
2. Crea una branch (`git checkout -b feature/nueva-feature`)
3. Commit cambios (`git commit -m 'feat: añadir nueva feature'`)
4. Push a la branch (`git push origin feature/nueva-feature`)
5. Abre un Pull Request

Licencia

Este proyecto está bajo la licencia MIT. Ver LICENSE para más detalles.

Autor

Tu Nombre - GitHub: @usuario - LinkedIn: perfil

~~~
---

##  2. Template de Model Card

```markdown
# Model Card: [Nombre del Modelo]

## Model Details

| Aspecto                    | Detalle                         |
|----------------------------|---------------------------------|
| **Nombre**                 | [Nombre descriptivo]            |
| **Versión**                | 1.0.0                           |
| **Tipo**                   | [Classification/Regression/etc] |
| **Framework**              | scikit-learn 1.3.0              |
| **Fecha de Entrenamiento** | YYYY-MM-DD                      |
| **Autor**                  | [Nombre]                        |

### Descripción
[1-2 párrafos describiendo qué hace el modelo y cómo funciona]

### Arquitectura
~~~

Pipeline: ├── Preprocessor (ColumnTransformer) │ ├── Numerical: Imputer + StandardScaler │ └── Categorical: Imputer + OneHotEncoder └── Classifier VotingClassifier ├── LogisticRegression (weight=0.4) └── RandomForestClassifier (weight=0.6)

```
---

## Intended Use

### Uso Principal
- [Describir el caso de uso principal]

### Usuarios Objetivo
- [Quién debería usar este modelo]

### Fuera de Alcance
- x [Usos para los que NO está diseñado]
- x [Limitaciones explícitas]

---

## Training Data

### Fuente
- **Dataset**: [Nombre/Fuente]
- **Tamaño**: [N] muestras
- **Período**: [Fechas si aplica]

### Características
| Feature   | Tipo        | Descripción   |
|-----------|-------------|---------------|
| feature1  | Numerical   | [Descripción] |
| feature2  | Categorical | [Descripción] |
| ...       | ...         | ...           |

### Target
- **Variable**: [nombre]
- **Distribución**: [X]% clase 0, [Y]% clase 1

### Preprocesamiento
- [Paso 1]
- [Paso 2]
- ...

---

## Evaluation Data

- **Split**: [X]% train, [Y]% validation, [Z]% test
- **Estratificación**: Sí/No
- **Seed**: 42

---

## Performance Metrics

### Métricas Principales

| Métrica   | Train   | Validation   | Test   |
|-----------|---------|--------------|--------|
| Accuracy  | 0.XX    | 0.XX         | 0.XX   |
| Precision | 0.XX    | 0.XX         | 0.XX   |
| Recall    | 0.XX    | 0.XX         | 0.XX   |
| F1 Score  | 0.XX    | 0.XX         | 0.XX   |
| AUC-ROC   | 0.XX    | 0.XX         | 0.XX   |

### Matriz de Confusión (Test)
```

```
          Predicted
          0     1
```

Actual 0 [TN] [FP] 1 [FN] [TP]

```
### Análisis por Segmento
| Segmento   | N   | Accuracy   | F1    |
|-----------|-----|-----------|-------|
| [Grupo A]  | X   | 0.XX       | 0.XX  |
| [Grupo B]  | Y   | 0.XX       | 0.XX  |

---

## Limitations

### Limitaciones Conocidas
- [Limitación 1]
- [Limitación 2]

### Casos de Fallo
- [Situación donde el modelo puede fallar]

### Sesgo Potencial
- [Descripción de posibles sesgos]

---

## Ethical Considerations

### Riesgos Identificados
- [Riesgo 1]
- [Riesgo 2]

### Mitigaciones Implementadas
- [Mitigación 1]
- [Mitigación 2]

### Recomendaciones de Uso
-  [Buena práctica]
- x [Práctica a evitar]

---

## Caveats and Recommendations

### Caveats
- [Advertencia 1]
- [Advertencia 2]

### Recomendaciones para Producción
- Reentrenar cada [período]
- Monitorear [métricas específicas]
- Umbral recomendado: [valor]

---

## References

- [Paper/Doc relevante 1]
- [Paper/Doc relevante 2]

---

## Contact

- **Autor**: [Nombre]
- **Email**: [email]
- **Issues**: [link a issues del repo]
```

3. Template de config.yaml

```yaml
# Configuration for ML Project
# configs/config.yaml

# Metadata
project:
  name: "mi-proyecto-ml"
  version: "1.0.0"
  description: "Descripción del proyecto"

# Data configuration
data:
  target_column: "target"

  categorical_features:
    - "feature_cat_1"
    - "feature_cat_2"

  numerical_features:
    - "feature_num_1"
    - "feature_num_2"
    - "feature_num_3"

  drop_columns:
    - "id"
    - "timestamp"

# Training configuration
training:
  test_size: 0.2
  validation_size: 0.1
  random_state: 42
  stratify: true

  cv_folds: 5
  cv_shuffle: true

  primary_metric: "f1"
  secondary_metrics:
    - "roc_auc"
    - "precision"
    - "recall"
    - "accuracy"

# Model configuration
model:
  type: "ensemble"  # ensemble, random_forest, logistic_regression

  ensemble:
    voting: "soft"
    weights: [0.4, 0.6]

  logistic_regression:
    C: 0.1
    class_weight: "balanced"
    solver: "liblinear"
    max_iter: 1000

  random_forest:
    n_estimators: 100
    max_depth: 10
    min_samples_split: 10
    min_samples_leaf: 5
    class_weight: "balanced_subsample"
    n_jobs: -1

# Preprocessing
preprocessing:
  numerical:
    imputer_strategy: "median"
    scaler: "standard"  # standard, minmax, robust

  categorical:
    imputer_strategy: "constant"
    imputer_fill_value: "missing"
    encoder: "onehot"
    encoder_drop: "first"
    handle_unknown: "ignore"

# Paths
paths:
  data:
    raw: "data/raw/"
    processed: "data/processed/"
  models: "models/"
  results: "results/"
  logs: "logs/"

# API configuration
api:
  host: "0.0.0.0"
  port: 8000
  workers: 1
  reload: false

  cors:
    allow_origins: ["*"]
    allow_methods: ["GET", "POST"]

# MLflow configuration
mlflow:
  enabled: true
  tracking_uri: "file:./mlruns"
  experiment_name: "mi-experimento"

# Logging
logging:
  level: "INFO"
  format: "%(asctime)s - %(name)s - %(levelname)s - %(message)s"
```

4. Template de Dockerfile

```dockerfile
# ============================================================================
# Multi-stage Dockerfile for ML Project
# ============================================================================

# ----------------------------------------------------------------------------
# Stage 1: Builder - Compile dependencies
# ----------------------------------------------------------------------------
FROM python:3.11-slim AS builder

# Build arguments
ARG PIP_NO_CACHE_DIR=1
ARG PIP_DISABLE_PIP_VERSION_CHECK=1

WORKDIR /build

# Install build dependencies
RUN apt-get update && apt-get install -y --no-install-recommends \
    gcc \
    g++ \
    && rm -rf /var/lib/apt/lists/*

# Copy and install requirements
COPY requirements.txt .
RUN pip wheel --no-cache-dir --wheel-dir /wheels -r requirements.txt

# ----------------------------------------------------------------------------
# Stage 2: Runtime - Final lightweight image
# ----------------------------------------------------------------------------
FROM python:3.11-slim AS runtime

# Labels
LABEL maintainer="tu@email.com"
LABEL version="1.0.0"
LABEL description="ML Project API"

# Environment variables
ENV PYTHONUNBUFFERED=1
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONPATH=/app
ENV PATH="/opt/venv/bin:$PATH"

WORKDIR /app

# Install runtime dependencies only
RUN apt-get update && apt-get install -y --no-install-recommends \
    curl \
    && rm -rf /var/lib/apt/lists/*

# Create non-root user
RUN groupadd -r appuser && useradd -r -g appuser appuser

# Copy wheels from builder and install
COPY --from=builder /wheels /wheels
RUN pip install --no-cache /wheels/* && rm -rf /wheels

# Copy application code
COPY --chown=appuser:appuser . .

# Create necessary directories
RUN mkdir -p logs models data && chown -R appuser:appuser /app

# Switch to non-root user
USER appuser

# Expose port
EXPOSE 8000

# Health check
HEALTHCHECK --interval=30s --timeout=10s --start-period=15s --retries=3 \
    CMD curl -f http://localhost:8000/health || exit 1

# Default command
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

5. Template de GitHub Actions Workflow

```yaml
# .github/workflows/ci.yml
name: CI Pipeline

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]

env:
  PYTHON_VERSION: '3.11'

jobs:
  # ═══════════════════════════════════════════════
  # Job 1: Lint and Format Check
  # ═══════════════════════════════════════════════
  lint:
    name: Lint & Format
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: ${{ env.PYTHON_VERSION }}

      - name: Install linters
        run: pip install black flake8 isort mypy

      - name: Check Black formatting
        run: black --check src/ tests/

      - name: Check isort
        run: isort --check-only src/ tests/

      - name: Flake8
        run: flake8 src/ --select=E9,F63,F7,F82

  # ═══════════════════════════════════════════════
  # Job 2: Tests
  # ═══════════════════════════════════════════════
  test:
    name: Tests
    runs-on: ubuntu-latest
    needs: lint

    steps:
      - uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: ${{ env.PYTHON_VERSION }}
          cache: 'pip'

      - name: Install dependencies
        run: |
          pip install -e ".[dev]"

      - name: Run tests with coverage
        run: |
          pytest tests/ -v \
            --cov=src \
            --cov-report=xml \
            --cov-report=term-missing \
            --cov-fail-under=70

      - name: Upload coverage
        uses: codecov/codecov-action@v4
        with:
          files: coverage.xml

  # ═══════════════════════════════════════════════
  # Job 3: Docker Build
  # ═══════════════════════════════════════════════
  docker:
    name: Docker Build
    runs-on: ubuntu-latest
    needs: test

    steps:
      - uses: actions/checkout@v4

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Build image
        run: |
          docker build -t proyecto:${{ github.sha }} .

      - name: Test image
        run: |
          docker run --rm proyecto:${{ github.sha }} python -c "import src; print('OK')"
```

6. Template de pyproject.toml

```toml
[build-system]
requires = ["setuptools>=65.0", "wheel"]
build-backend = "setuptools.build_meta"

[project]
name = "mi-proyecto-ml"
version = "1.0.0"
description = "Descripción del proyecto ML"
readme = "README.md"
requires-python = ">=3.10"
license = {file = "LICENSE"}
authors = [
    {name = "Tu Nombre", email = "tu@email.com"}
]
keywords = ["machine-learning", "classification", "mlops"]
classifiers = [
    "Development Status :: 4 - Beta",
    "Intended Audience :: Science/Research",
    "License :: OSI Approved :: MIT License",
    "Programming Language :: Python :: 3",
    "Programming Language :: Python :: 3.10",
    "Programming Language :: Python :: 3.11",
    "Programming Language :: Python :: 3.12",
]

dependencies = [
    "pandas>=1.5.0",
    "numpy>=1.23.0",
    "scikit-learn>=1.2.0",
    "joblib>=1.2.0",
    "pyyaml>=6.0",
    "pydantic>=2.0.0",
    "fastapi>=0.100.0",
    "uvicorn>=0.22.0",
]

[project.optional-dependencies]
dev = [
    "pytest>=7.0.0",
    "pytest-cov>=4.0.0",
    "black>=23.0.0",
    "isort>=5.12.0",
    "flake8>=6.0.0",
    "mypy>=1.0.0",
    "pre-commit>=3.0.0",
]
ml = [
    "mlflow>=2.0.0",
    "optuna>=3.0.0",
]

[project.urls]
Homepage = "https://github.com/usuario/repo"
Repository = "https://github.com/usuario/repo"

[tool.setuptools.packages.find]
where = ["."]
include = ["src*"]

[tool.black]
line-length = 120
target-version = ['py310', 'py311']

[tool.isort]
profile = "black"
line_length = 120

[tool.pytest.ini_options]
minversion = "7.0"
testpaths = ["tests"]
addopts = ["-v", "--cov=src", "--cov-report=term-missing"]

[tool.mypy]
python_version = "3.10"
ignore_missing_imports = true

[tool.coverage.run]
source = ["src"]
omit = ["tests/*"]

[tool.coverage.report]
exclude_lines = [
    "pragma: no cover",
    "if __name__ == .__main__.:",
]
```

Navegación

◀ Anterior	Índice	▶ Siguiente
20_PLAN_ESTUDIOS.md	Índice	22_CHECKLIST.md

Módulo 21 Completado