UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

APLICACIÓN DE LOS ALGORITMOS GENÉTICOS EN EL APRENDIZAJE DE MÁQUINA

CURSO:

APRENDIZAJE DE MÁQUINA

DOCENTE:

FERNANDEZ CHAMBI MAYENKA

SEMESTRE: NOVENO

INTEGRANTES:

AQUINO SANDOVAL KENYA XIOMARA

ARACAYO MAMANI JHON MARCO

CANAZA PAUCARA JUAN DIEGO

CASTRO CCALLO VANESSA YHOSELIN

PUNO - PERÚ

2025 - II

Algoritmos Genéticos aplicados a ML

Feature Selection · Hyperparameter Optimization · Neuroevolución

Objetivo. Demostrar, con ejemplos funcionales y reproducibles, cómo los **Algoritmos Genéticos (AG)** resuelven tres tareas clave de *Machine Learning* (ML): selección de características (FS), optimización de hiperparámetros (HPO) y búsqueda de arquitecturas de redes neuronales (NE).

Estructura del trabajo. El proyecto se organiza en tres carpetas: **feature selection/**, **Hyperparameter optimization/** y **Neuroevolution/**.

1) ¿Qué es un Algoritmo Genético?

Los AG son meta-heurísticas inspiradas en la evolución biológica. Mantienen una **población** de soluciones (**cromosomas**) y, en cada generación, aplican:

- **Representación:** cómo se codifica una solución (binaria, discreta/continua o mixta).
- **Inicialización:** población aleatoria con semilla fija para reproducibilidad.
- **Fitness:** medida de desempeño (accuracy en CV/validación) con posibles penalizaciones por complejidad.
- **Selección:** torneo o ruleta, favoreciendo a los mejores.
- **Cruce:** recombinación (1 punto o uniforme) para explorar regiones prometedoras.
- **Mutación:** cambios pequeños (1–2 genes) para mantener diversidad.
- **Elitismo:** preserva a los mejores individuos.
- **Terminación:** número fijo de generaciones.

Este ciclo se aplicó **sin cambios conceptuales** en los tres casos, variando solo la **codificación** y la **función de aptitud**.

2) Experimentos y aportes

2.1 Feature Selection (FS)

- **Representación:** cromosoma **binario** (1=usa la variable, 0=no).
- **Fitness:** Accuracy_CV(LogisticRegression) − α·k (k = número de *features*; penalización leve para favorecer simplicidad). Se garantiza al menos 1 *feature*.
- **GA:** torneo, cruce a 1 punto, mutación *flip bit*, elitismo.
- **Implementación y evidencias:**
  - Preprocesamiento robusto (normalización, *one-hot* si aplica).
  - Curva de mejor fitness por generación (**fitness_curve.png**).
  - Subconjunto resultante en **selected_features.csv** y máscara/metrics en **bloque1_selection.joblib**.

  Conclusión: reduce dimensionalidad manteniendo o mejorando el rendimiento frente a usar todas las variables; modelos más simples y explicables.

2.2 Hyperparameter Optimization (HPO) — RandomForest

- **Representación:** cromosoma **mixto** por **índices** a dominios discretos: n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features (incluye "sqrt", "log2" y fracciones).
- **Fitness:** Accuracy_CV con 5 folds (semilla 42).
- **GA:** torneo k=3, cruce **uniforme**, mutación de 1–2 genes, elitismo (2).
- **Flujo y evidencias:**
    - Puede usar el **subset de FS** (máscara desde bloque1_selection.joblib).
    - Reporte del **mejor set de hiperparámetros**, curva del mejor fitness y rendimiento en **test** tras reentrenar.

  Conclusión: el GA evita búsquedas exhaustivas y encuentra configuraciones robustas que equilibran sesgo-varianza.

2.3 Neuroevolución (NE) — MLP (Keras/TensorFlow)

- **Representación:** vector **entero/real** [n_layers, units_1, units_2, dropout, log10(lr)] con n_layers ∈ {1,2}, units ∈ {32,64,128}, dropout ∈ [0,0.5], lr ∈ [1e−4,1e−2].
- **Fitness: val_accuracy** máxima en entrenamiento corto (p. ej., 5 épocas) menos penalización leve por complejidad.
- **GA:** ruleta (con *clipping*), cruce 1 punto ajustando tamaño según n_layers, mutación 1–2 genes, elitismo ≈20%.
- **Evidencias:**
    - Gráfico **evolution_progress.png** (Mejor vs Promedio por generación).
    - Arquitectura ganadora con capas/unidades/*dropout*/lr.
- **Conclusión:** con presupuestos modestos aparecen arquitecturas de 1–2 capas y 64–128 neuronas con *dropout* moderado y lr ~ 1e−3, balanceando calidad/costo.

3) Conclusiones generales

- **Marco unificador.** Un mismo esquema evolutivo resuelve problemas combinatorios (FS), mixtos (HPO) y estructurales (NE).
- **Fortalezas.** Búsqueda **global**, adaptable a espacios **mixtos** y fácil de incluir **penalizaciones** por complejidad.
- **Límites.** Costo computacional (población × generaciones × CV) y necesidad de calibrar operadores; para NE conviene **GPU/Colab**.
- **Buenas prácticas empleadas.** Semilla 42, CV estratificada cuando corresponde, registro de curvas y preservación de artefactos (joblib, CSV, PNG); requirements.txt unificado; notebooks/scripts listos para **Restart & Run All**.

Reproducibilidad. El repositorio incluye dependencias, scripts y notebooks autoejecutables. Los resultados (curvas, CSV, Joblib, PNG) se generan automáticamente al correr cada bloque.

GITHUB: https://github.com/tamoil-0/genetic-ml.git