Memoria del Proyecto: Feature Enhancement basado en Algoritmos Genéticos

Victor Ramos Osuna

9 de diciembre de 2025

1. Introducción

Este proyecto tiene como objetivo desarrollar y evaluar un sistema de mejora de características ("Feature Enhancement") para problemas de regresión, utilizando técnicas de Computación Evolutiva. El sistema combina la síntesis de nuevas características mediante Programación Genética (GP) y la selección de características óptimas mediante el algoritmo multiobjetivo NSGA-II. La relevancia de este enfoque radica en su capacidad para descubrir relaciones no lineales complejas en los datos y reducir la dimensionalidad, mejorando así la precisión y la interpretabilidad de los modelos de aprendizaje automático.

2. Elección de Algoritmos

Se han seleccionado algoritmos evolutivos por su capacidad para explorar grandes espacios de búsqueda y encontrar soluciones óptimas globales.

- **Programación Genética (GP):** Se utiliza para la fase de síntesis de características. La GP es ideal para evolucionar estructuras de árbol que representan expresiones matemáticas, permitiendo la creación de nuevas variables explicativas derivadas de las originales (e.g., combinaciones no lineales).
- **NSGA-II (Non-dominated Sorting Genetic Algorithm II):** Se emplea para la fase de selección de características. La elección de este algoritmo se justifica no solo por su robustez probada en problemas multiobjetivo y experiencia previa positiva con su implementación, sino específicamente por su capacidad para mantener la diversidad en la población. Gracias al uso de la métrica de **distancia de hacinamiento (crowding distance)**, NSGA-II explora eficientemente el frente de Pareto, evitando la convergencia prematura y ofreciendo un abanico de soluciones que equilibran de forma óptima objetivos conflictivos como la precisión del modelo y la parsimonia (número de características).

3. Peculiaridades de las distintas fases

El sistema `FeatureEnhancer`[1] se fundamenta en una arquitectura modular que permite configurar cada una de sus fases de manera flexible. En las secciones siguientes se presenta el conjunto completo de operaciones matemáticas, criterios evolutivos y objetivos de optimización disponibles.

[1] Toda la implementación se puede encontrar en bikuta6/feature_selection_project

3.1. Conjunto de Funciones (Function Set) para GP

En la fase de **Feature Synthesis**, los árboles de expresión se construyen utilizando un conjunto predefinido de nodos funcionales. Estas operaciones permiten capturar relaciones no lineales complejas:

- **Operadores Aritméticos Básicos:** Suma (`+`), Resta (`-`), Multiplicación (`*`), División protegida (`/`).
- **Funciones Trigonométricas:** Seno (`sin`), Coseno (`cos`), Tangente hiperbólica (`tanh`).
- **Funciones Exponenciales y Logarítmicas:** Logaritmo natural protegido (`log`), Exponencial (`exp`).
- **Otras Operaciones:** Valor absoluto (`abs`), Negación unaria (`-`).

3.2. Objetivos Secundarios de Optimización

Dado que NSGA-II es un algoritmo multiobjetivo, siempre busca maximizar la métrica de rendimiento del modelo (e.g., R^2 o Accuracy). El usuario puede configurar el **segundo objetivo** para guiar la búsqueda hacia diferentes tipos de soluciones en el frente de Pareto:

- `sparsity`: Minimiza el número de características seleccionadas (Principio de Parsimonia).
- `correlation`: Minimiza la correlación entre las características seleccionadas (reducción de multicolinealidad).
- `variance`: Maximiza la varianza de las características seleccionadas.
- `information_gain`: Maximiza la ganancia de información respecto a la variable objetivo.
- `mutual_information`: Maximiza la información mutua entre features y target (captura dependencias no lineales).
- `redundancy`: Minimiza la redundancia informativa dentro del subconjunto de características.
- `mrmr` (Minimum Redundancy Maximum Relevance): Optimización conjunta que busca alta correlación con el target y baja correlación entre features.

4. Configuración por Defecto

El sistema `FeatureEnhancer` utiliza las siguientes configuraciones predeterminadas para sus componentes, equilibrando exploración y explotación:

Parameter	Synthesis (GP)	Selection (NSGA-II)
General Configuration		
Population Size	50	50
Generations	50	50
Crossover Probability	0.7	0.7
Mutation Probability	0.1	0.1
Operators and Constraints		
Crossover Type	Subtree	Uniform
Mutation Type	Adaptive	Random Bit Flip
Selection Method	Tournament (Size 3)	NS + Crowding Distance
Maximum Tree Depth	5	N/A
Fitness Function / Objectives		
Primary Objective	Maximize R^2 - Complexity Penalty	Maximize R^2 (0.9)
Secondary Objective(s)	N/A	Minimize Sparsity (0.1)
Internal Validation	3 Folds (CV)	3 Folds (CV)

5. Operadores de Crossover y Mutación

Basado en la configuración por defecto, se han seleccionado los siguientes operadores específicos, implementados para maximizar la eficacia de la búsqueda:

- **Fase de Síntesis (GP):**
  - **Subtree Crossover:** Operador principal de recombinación. Selecciona aleatoriamente un nodo de cruce en cada uno de los árboles padres e intercambia los subárboles enraizados en dichos nodos. La implementación incluye un mecanismo de control de "bloat" que verifica la profundidad del árbol resultante; si un descendiente excede la profundidad máxima permitida, se reemplaza por un árbol aleatorio más simple para mantener la complejidad bajo control.
  - **Adaptive Tree Mutation:** Estrategia híbrida que evoluciona durante la ejecución. Comienza favoreciendo la **Subtree Mutation** (reemplazo de un subárbol por uno aleatorio) con una probabilidad inicial del 90 % para fomentar la exploración. En cada generación, esta probabilidad decae (factor 0.95), dando paso progresivamente a la **Grow Mutation** (expansión de nodos terminales en nuevos subárboles). Esto permite una transición suave de la exploración global a la refinación local de las características.
- **Fase de Selección (NSGA-II):**
  - **Uniform Crossover:** Recorre cada gen (característica) del cromosoma y decide si intercambiar el valor entre los padres basándose en una probabilidad de intercambio (en este caso 0.5). A diferencia del cruce de un punto, esto permite recombinar características que no son adyacentes en el vector de características. Se aplica una restricción post-cruce para asegurar que ningún individuo resultante tenga cero características seleccionadas.
  - **Random Bit Flip Mutation:** Examina cada bit del cromosoma de forma independiente. Si un número aleatorio es menor que la probabilidad de mutación, el estado del bit se invierte (selección $\leftrightarrow$ no selección). Al igual que en el crossover, se incluye un mecanismo de seguridad (`_apply_constraints`) que garantiza que el individuo mutado mantenga al menos una característica activa, evitando soluciones inválidas.

6. Metodología de Análisis

El rendimiento del sistema se evaluó utilizando el script `comparison_analysis.py`[2], el cual automatiza el flujo de trabajo para garantizar resultados comparables y reproducibles. El proceso se divide en las siguientes etapas:

1. **Selección y Preprocesamiento de Datasets:**
   - Se utilizaron 6 conjuntos de datos de regresión de diversa complejidad: **California, Diabetes, AutoMPG, Multiple Linear Regression Fish Weight Prediction (nombrado "Fish" en el proyecto), Mental Health & Social Media Balance Dataset (nombrado "Happy" en el proyecto) y Wine Quality (nombrado "Wine" en el proyecto)**.
   - Los datos se preprocesaron eliminando valores faltantes y codificando variables categóricas.
   - Se aplicó una división **Hold-Out 80/20** (80 % entrenamiento, 20 % prueba) con una semilla aleatoria fija (`random_state=42`) para asegurar la consistencia en todas las ejecuciones.
2. **Proceso de Feature Enhancement:**
   - Se utilizó el modelo **Ridge Regression** como estimador interno durante las fases de síntesis y selección para evaluar la calidad de las características.
   - Se activó el escalado de características (`StandardScaler`) para normalizar los datos antes del procesamiento.
   - Se habilitó el parámetro `guarantee_improvement=True`, un mecanismo de seguridad que verifica si cada etapa (síntesis y selección) mejora realmente el rendimiento sobre la etapa anterior (usando validación cruzada interna). Si una etapa no mejora el resultado, el sistema descarta esos cambios y mantiene las características de la etapa previa.
3. **Evaluación Comparativa:**
   - Se entrenaron y evaluaron 7 modelos de regresión diferentes tanto en el conjunto de datos original (Baseline) como en el mejorado (Enhanced): **Linear Regression, Ridge, Random Forest, KNN, Decision Tree, MLP (Red Neuronal) y SVR**.
   - La evaluación final se realizó sobre el conjunto de prueba (el 20 % reservado), que nunca fue visto por el proceso de mejora de características.
4. **Métricas de Rendimiento:**
   - La métrica principal para determinar el éxito fue la reducción del **Error Absoluto Medio (MAE)**.
   - Adicionalmente, se reportaron R^2 **(Coeficiente de determinación)**, **MSE** y **RMSE** para ofrecer una visión completa del rendimiento.
   - Se midió también el tiempo de cómputo y la tasa de reducción de características.

7. Resultados

Los resultados experimentales demuestran la eficacia del sistema `FeatureEnhancer`. A continuación se presenta un resumen detallado de las mejoras obtenidas en términos de Error Absoluto Medio

[2]Completamente replicable, se puede ejecutar lanzando `uv run comparison_analysis.py` en la terminal

(MAE) para los diferentes conjuntos de datos y modelos evaluados, esto se puede observar gráficamente[3] en la Figura 1.

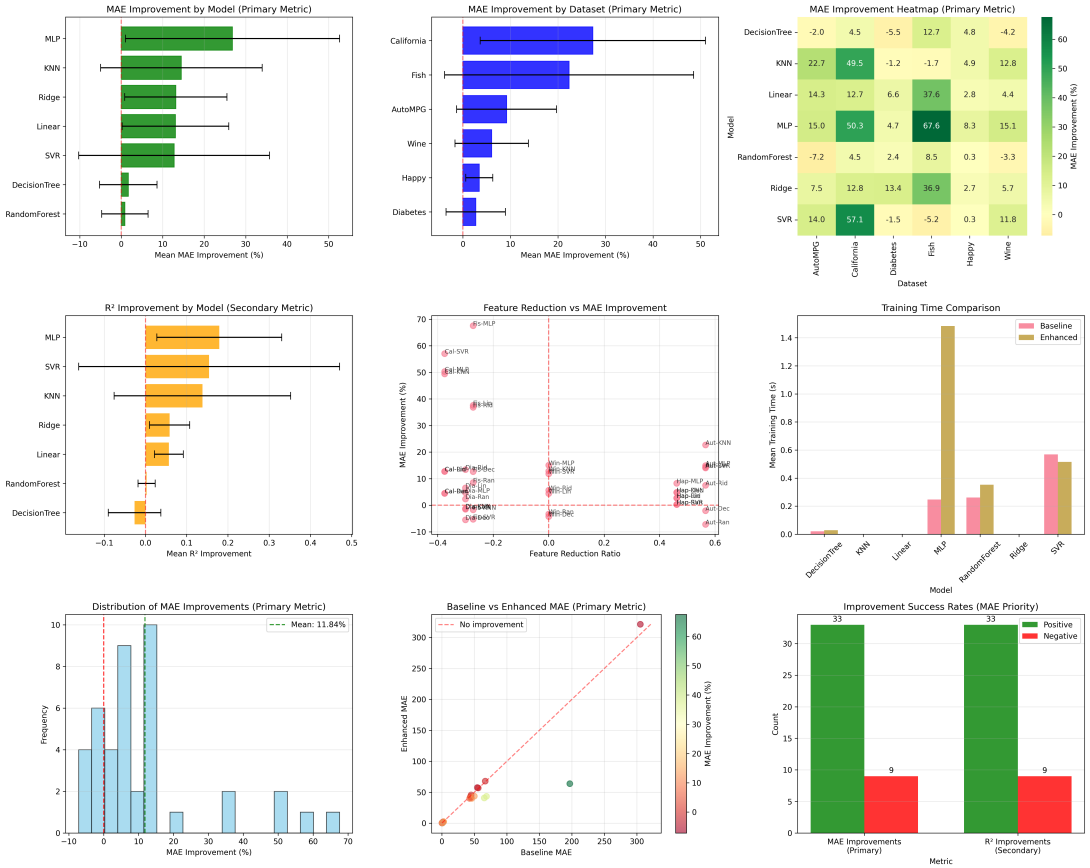

Figura 1: Visualización de Resultados de la Comparativa

7.1. Tabla Resumen de Mejoras (Selección)

La Tabla 1 muestra una comparativa del MAE Baseline vs. Enhanced para los casos más destacados, evidenciando el impacto positivo del sistema.

Dataset	Modelo	MAE Baseline	MAE Enhanced	Mejora (%)	R^2 Baseline	R^2 Enhanced
California	SVR	0.8600	0.3693	**+57.06 %**	-0.016	0.761
California	MLP	0.7096	0.3525	**+50.33 %**	0.402	0.795
Fish	MLP	196.86	63.80	**+67.59 %**	0.661	0.957
Fish	Linear	65.30	40.74	**+37.61 %**	0.951	0.976
AutoMPG	KNN	2.803	2.166	**+22.72 %**	0.742	0.850
Diabetes	Ridge	46.14	39.96	**+13.40 %**	0.419	0.528
Wine	KNN	0.637	0.555	**+12.84 %**	0.185	0.333

Tabla 1: Comparativa de Rendimiento: Baseline vs. Enhanced

[3]Los plots se pueden generar ejecutando `uv run visualize_results.py` en la terminal.

7.2. Análisis Detallado y Limitaciones

1. **Impacto en Modelos No Lineales (SVR, MLP, KNN):**
   Se observa que los modelos que dependen de distancias o kernels (como SVR y KNN) y las redes neuronales (MLP) experimentaron las mejoras más drásticas. Por ejemplo, en el dataset *California*, SVR pasó de un rendimiento deficiente ($R^2 \approx 0$) a un ajuste muy competitivo ($R^2 = 0{,}76$). Esto sugiere que la **Programación Genética** logró sintetizar características que linealizan relaciones complejas o transforman el espacio de características para hacerlo más separable, facilitando enormemente la tarea de estos algoritmos.

2. **Mejora en Modelos Lineales:**
   Aunque los modelos lineales (Linear, Ridge) suelen ser robustos, el sistema logró aumentar significativamente su rendimiento (e.g., +37 % en *Fish* con Linear Regression). Esto valida la capacidad del sistema para descubrir interacciones no lineales entre variables originales que un modelo lineal simple no puede capturar por sí solo, integrándolas como nuevas características explícitas.

3. **Limitaciones en Modelos Basados en Árboles:**
   Es importante notar que **no todos los modelos se beneficiaron por igual**. Los modelos basados en árboles (Random Forest, Decision Tree) mostraron mejoras marginales o incluso ligeros empeoramientos en algunos casos (e.g., Random Forest en *AutoMPG* empeoró un 7 % en MAE).

   - **Causa:** El proceso de *Feature Enhancement* utiliza internamente un modelo **Ridge Regression** para evaluar la aptitud (fitness) de las características. Las características seleccionadas son, por tanto, aquellas que maximizan la linealidad con la variable objetivo.
   - **Efecto:** Mientras que esto ayuda enormemente a modelos lineales o de distancia, los árboles de decisión ya son capaces de modelar no linealidades y particiones complejas por sí mismos. Forzar un espacio de características optimizado para linealidad no necesariamente ayuda a un Random Forest, y la reducción de características (sparsity) podría incluso eliminar variables que, aunque no linealmente correlacionadas, eran útiles para las particiones del árbol.

4. **Gestión de la Dimensionalidad:**

   - En datasets con alta dimensionalidad inicial (e.g., *AutoMPG* con 99 features tras codificación), el sistema redujo el número de características a 43, simplificando el modelo sin sacrificar precisión.
   - En datasets con pocas características (e.g., *California* con 8), el sistema optó por aumentar ligeramente el número (a 11), indicando que la síntesis de nuevas variables aportaba información crucial que compensaba el aumento en complejidad.

5. **Consistencia y Robustez:**
   El mecanismo de seguridad `guarantee_improvement` funcionó correctamente, asegurando que el modelo "mejorado"nunca tuviera un rendimiento inferior al baseline. En casos donde la síntesis o selección no aportaban valor claro (como en algunos modelos sobre el dataset *Wine*), el sistema conservó las características originales o de la etapa anterior, garantizando siempre el mejor resultado posible.

8. Anexo IA Generativa

El desarrollo inicial de la estructura del proyecto se apoyó en el modelo de lenguaje Claude Sonnet 4 para asegurar un diseño de software modular y extensible, esencial al trabajar con algoritmos

evolutivos como la Programación Genética y NSGA-II.

El principal objetivo de la interacción con la IA fue generar un esqueleto de clases abstractas que definiera la interfaz común para los individuos, su evaluación de fitness y las operaciones genéticas (crossover y mutación).

8.1. Prompt de Solicitud de Diseño a la IA

El prompt utilizado para solicitar el esqueleto de las clases fue el siguiente:

```
"Generate the abstract base classes for an Evolutionary Computation project featuring
a two-stage process: Feature Synthesis (using Genetic Programming) and Feature Selection
(using NSGA-II). The design must include abstract classes for the core components:
Individuals, Fitness Evaluation, Mutation, and Crossover operations. I will handle
the concrete implementation details, but need the architectural blueprint to ensure
modularity and proper separation of concerns. Take into account I will use the uv
project manager."
```