Información General

Valor:	25 % de la calificación final del curso
Modalidad:	Grupos de 1 a 3 estudiantes
Lenguaje:	C (estándar C11 o C99)

1 Descripción del proyecto

En este proyecto, los estudiantes implementarán y analizarán algoritmos fundamentales de aprendizaje automático (Machine Learning) en el lenguaje C, partiendo de un código base con estructuras de datos fundamentales. El objetivo es comprender profundamente los fundamentos matemáticos y algorítmicos detrás de estas técnicas, desarrollando implementaciones eficientes sin depender de bibliotecas especializadas. Los estudiantes aplicarán estos algoritmos a problemas prácticos de clasificación, regresión y agrupamiento, evaluando su rendimiento y proponiendo optimizaciones.

2 Objetivos de aprendizaje

- Implementar correctamente algoritmos fundamentales de aprendizaje automático desde cero
- Comprender los principios matemáticos que sustentan los algoritmos de ML
- Desarrollar estructuras de datos eficientes para procesamiento de datos
- Analizar empíricamente la complejidad y rendimiento de los algoritmos
- Aplicar técnicas de optimización para mejorar la eficiencia computacional
- Diseñar un sistema modular y extensible de análisis predictivo
- Documentar adecuadamente el código y los resultados del análisis

3 Requisitos técnicos

Se proporcionará un código base con estructuras de datos fundamentales (matrices, conjuntos de datos, árboles) y funciones para carga y procesamiento básico de datos. A partir de este código, los estudiantes deberán:

3.1 Algoritmos de clasificación

Implementar un algoritmo de clasificación:

1. **k-vecinos más cercanos (k-NN):**
    - Implementación con diferentes métricas de distancia (Euclidiana, Manhattan, Coseno)
    - Optimización para búsqueda eficiente de vecinos (árboles k-d, hashing localidad-sensitivo)
    - Soporte para ponderación por distancia y técnicas de normalización

3.2 Algoritmos de regresión o agrupamiento

Implementar el siguiente algoritmo de regresión o agrupamiento:

1. **Regresión lineal:**
    - Implementación con ecuaciones normales y descenso de gradiente
    - Soporte para regularización (Ridge, Lasso)
    - Análisis de la calidad del ajuste (R^2, MSE, MAE)
2. **K-means:**
    - Implementación con inicialización aleatoria y k-means++
    - Optimización de criterios de convergencia
    - Análisis de sensibilidad al número de clusters

3.3 Técnicas de evaluación y validación

Métricas de evaluación:

- Para clasificación: precisión, recall, F1-score, matriz de confusión
- Para regresión: MSE, MAE, R^2
- Para agrupamiento: índice de silueta, inercia

3.4 Optimizaciones

Implementar una (1) de las siguientes optimizaciones para los algoritmos seleccionados:

1. **Optimización algorítmica:**
    - Implementación de variantes mejoradas de los algoritmos básicos
    - Análisis comparativo de eficiencia y precisión
2. **Optimización de memoria:**
    - Técnicas para reducir el uso de memoria (matrices dispersas, etc.)
    - Análisis del impacto en rendimiento vs. uso de memoria
3. **Optimización de caché:**
    - Técnicas para mejorar la localidad de datos y reducir fallos de caché
    - Medición del impacto en rendimiento
4. **Técnicas de aproximación:**
    - Implementación de métodos aproximados para acelerar algoritmos
    - Análisis del balance entre precisión y velocidad

4 Aplicación práctica: Sistema de análisis predictivo

Desarrollar un sistema completo de análisis predictivo que funcione desde la línea de comandos, con las siguientes funcionalidades:

1. **Carga y preprocesamiento de datos:**
    - Lectura de archivos CSV y similares
    - Normalización, manejo de valores faltantes
    - División en conjuntos de entrenamiento/validación/prueba
2. **Entrenamiento de modelos:**
    - Interfaz unificada para todos los algoritmos implementados
    - Configuración de hiperparámetros mediante argumentos
    - Serialización de modelos entrenados para uso posterior
3. **Evaluación y predicción:**
    - Evaluación de modelos con múltiples métricas
    - Predicción sobre nuevos datos
    - Generación de informes de rendimiento
4. **Visualización de resultados:**
    - Generación de archivos CSV para visualización externa
    - Estadísticas descriptivas en formato tabular
    - Matrices de confusión y otras métricas visuales en texto
5. **Interfaz de línea de comandos:**
    - Programa ejecutable con argumentos bien definidos (sin menú interactivo)
    - Soporte para pipelines de procesamiento completos
    - Documentación accesible mediante flag `--help` o `-h`

5 Entregables

1. **Código fuente:**
    - Archivos .c y .h bien organizados y comentados
    - Estructura modular con separación clara de responsabilidades
    - Makefile para compilación del proyecto
2. **Repositorio GitHub:**
    - Repositorio con historial de commits de todos los integrantes
    - Archivo README.md con instrucciones detalladas
3. **Datos de prueba:**

- Conjunto de datos para demostrar la funcionalidad de los algoritmos
- Scripts para generar datos sintéticos (opcional)

4. **Informe técnico:** Documento PDF (máximo 30 páginas) que incluya:
   - Fundamentos teóricos de los algoritmos implementados
   - Detalles de implementación y optimizaciones realizadas
   - Análisis de complejidad (tiempo y espacio)
   - Resultados experimentales con gráficos comparativos
   - Análisis de rendimiento de las optimizaciones implementadas
5. **Presentación:** Diapositivas para una presentación oral de 10-15 minutos

6 Presentación Oral Obligatoria

Como parte integral de la evaluación, cada grupo deberá realizar una presentación oral obligatoria de su proyecto, con las siguientes características:

- Duración: 10-15 minutos máximo
- Participación: Todos los integrantes del grupo deben participar activamente
- Contenido: Algoritmos implementados, optimizaciones, resultados experimentales
- Preguntas: Capacidad de responder a consultas técnicas del profesor y compañeros

7 Sistema de Evaluación

El proyecto será evaluado considerando tres componentes principales: el código (55 %), el informe técnico (35 %) y la evaluación de pares (10 %). La nota del informe técnico estará afectada por un factor multiplicador dependiente de la presentación oral.

7.1 Rúbrica para la evaluación del código

7.2 Rúbrica para la evaluación del informe técnico

7.3 Factor multiplicador de la presentación oral

La nota del informe técnico será modificada por un factor multiplicador basado en la calidad de la presentación oral:

$$\textbf{Nota final del informe} = \textbf{Nota original del informe} \times \textbf{Factor de presentación}$$

7.4 Valoración de originalidad y material extra

Se premiarán especialmente las ideas originales y el material extra que enriquezca el proyecto:

- **Ideas originales:** Implementaciones novedosas de los algoritmos, optimizaciones creativas o enfoques alternativos que demuestren un pensamiento innovador
- **Algoritmos adicionales:** Implementación de algoritmos no requeridos o extensiones a los algoritmos básicos

- **Visualizaciones:** Herramientas para visualizar el comportamiento de los algoritmos o los resultados del análisis
- **Conjuntos de datos:** Creación de conjuntos de datos interesantes o aplicación a problemas prácticos reales

7.5 Evaluación de pares

Como parte del proceso de evaluación, se implementará un sistema de evaluación de pares donde cada integrante del grupo evaluará la contribución y desempeño de sus compañeros de equipo:

- Cada integrante evaluará individualmente a los demás miembros de su propio grupo
- La evaluación se realizará con una escala de 0 a 5
- La evaluación incluirá aspectos como contribución al código, participación en reuniones, cumplimiento de responsabilidades asignadas y calidad del trabajo
- Las evaluaciones serán confidenciales y entregadas directamente al profesor el día de la presentación oral
- Esta evaluación constituirá un 10 % de la nota final individual de cada estudiante

Este proceso promueve la responsabilidad individual, el trabajo equitativo y la transparencia en las contribuciones de cada miembro al proyecto grupal.

7.6 Cálculo de la Nota Final

Nota final = [(Nota del código $\times$ 0.55) + (Nota del informe $\times$ Factor de presentación $\times$ 0.35)] $\times$ 0.9 + (Evaluación de pares $\times$ 0.1)

8 Consejos y Recomendaciones

- Comience estudiando detenidamente el código base proporcionado para comprender las estructuras disponibles.
- Implemente primero versiones simples de los algoritmos y luego agregue optimizaciones.
- Utilice conjuntos de datos pequeños para validar la correctitud antes de probar con datos grandes.
- Considere la eficiencia desde el inicio, especialmente en la gestión de memoria.
- Documente su código a medida que lo desarrolla, no lo deje para el final.
- Utilice herramientas como Valgrind para detectar fugas de memoria y otros problemas.
- Realice pruebas incrementales de cada componente del sistema.

8.1 Descripción de las categorías principales

- **Aprendizaje supervisado:** Algoritmos que aprenden a partir de datos etiquetados para predecir resultados en nuevos datos.
  - **Clasificación:** Predice categorías discretas. Ejemplos: filtrado de spam, diagnóstico médico.
  - **Regresión:** Predice valores continuos. Ejemplos: precio de viviendas, predicción de ventas.
- **Aprendizaje no supervisado:** Algoritmos que descubren patrones en datos no etiquetados.
  - **Agrupamiento:** Agrupa datos similares. Ejemplos: segmentación de clientes, agrupación de documentos.
  - **Reducción de dimensionalidad:** Reduce la complejidad de los datos preservando sus características principales.
  - **Detección de anomalías:** Identifica observaciones atípicas. Ejemplos: detección de fraude, monitoreo de sistemas.
- **Deep learning:** Algoritmos basados en redes neuronales artificiales con múltiples capas.
  - **Redes convolucionales (CNN):** Especializadas en procesar datos con estructura de cuadrícula (imágenes).
  - **Redes recurrentes (RNN):** Diseñadas para datos secuenciales (texto, series temporales).
  - **Modelos generativos:** Capaces de generar nuevos datos similares a los datos de entrenamiento.
- **Aprendizaje por refuerzo:** Algoritmos que aprenden a tomar decisiones secuenciales mediante prueba y error.

8.2 Relación con el proyecto actual

En este proyecto, nos centramos en algoritmos clásicos de machine learning que son adecuados para implementar en C. Aunque el esquema anterior muestra la amplia variedad de algoritmos existentes, para este proyecto hemos seleccionado:

- **Clasificación:** k-NN
- **Regresión:** Regresión Lineal
- **Agrupamiento:** K-Means

Estos algoritmos ofrecen un buen equilibrio entre valor educativo, complejidad de implementación en C, y aplicabilidad en problemas reales.

9 Recursos recomendados

9.1 Libros y referencias teóricas

- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.
- Theobald, O. (2017). *Machine Learning for Absolute Beginners* (2nd ed.). Scatterplot Press.
- Burkov, A. (2019). *The Hundred-Page Machine Learning Book*. Andriy Burkov.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Grus, J. (2019). *Data Science from Scratch: First Principles with Python* (2nd ed.). O'Reilly Media.
- Howard, J., & Gugger, S. (2020). *Deep Learning for Coders with fastai and PyTorch: AI Applications Without a PhD*. O'Reilly Media.

9.2 Recursos para implementación en C

- Repositorio Github: tiny-dnn - https://github.com/tiny-dnn/tiny-dnn (Implementación minimalista de redes neuronales en C++, buena referencia para técnicas de optimización)
- Repositorio Github: Cranium - https://github.com/100/Cranium (Biblioteca de redes neuronales en C puro)
- Repositorio Github: LIBLINEAR - https://github.com/cjlin1/liblinear (Biblioteca de referencia para regresión logística en C/C++)
- ViennaCL - http://viennacl.sourceforge.net/ (Biblioteca para álgebra lineal de alto rendimiento, útil para algoritmos de ML)

9.3 Recursos de referencia para algoritmos

- Scikit-learn - https://scikit-learn.org/1.5/index.html (Biblioteca de Python para aprendizaje automático con excelente documentación sobre algoritmos, parámetros y ejemplos)
- Papers With Code - https://paperswithcode.com/ (Implementaciones de algoritmos de ML con sus papers asociados)
- Machine Learning Mastery - https://machinelearningmastery.com/ (Tutoriales detallados sobre algoritmos de ML)

9.4 Conjuntos de datos

- Repositorio UCI Machine Learning: https://archive.ics.uci.edu/ml/index.php (Colección extensa de conjuntos de datos para ML)
- Kaggle Datasets: https://www.kaggle.com/datasets (Plataforma con miles de conjuntos de datos para práctica)
- CIFAR-10 y CIFAR-100: https://www.cs.toronto.edu/~kriz/cifar.html (Conjuntos de imágenes para clasificación)

- Boston Housing Dataset: Para regresión (disponible en UCI)
- Wine Dataset: Para clasificación y clustering (disponible en UCI)
- Iris Dataset: Conjunto clásico para clasificación multi-clase

9.5 Herramientas y utilidades

- Valgrind: https://valgrind.org/ (Herramienta para detección de fugas de memoria y perfilado)
- GProf: Herramienta de perfilado incluida con GCC (útil para encontrar cuellos de botella)
- GDB: GNU Debugger para C (esencial para depuración)
- Doxygen: https://www.doxygen.nl/ (Generador de documentación para C)
- BLAS (Basic Linear Algebra Subprograms): Fundamental para implementaciones eficientes
- CBLAS: Interfaz C para BLAS
- OpenBLAS: https://www.openblas.net/ (Implementación optimizada de BLAS)
- LAPACK: Biblioteca para álgebra lineal avanzada

9.6 Comunidades y foros

Criterio	Excelente (90-100 %)	Bueno (75-89 %)	Satisfactorio (60-74 %)	Insuficiente (0-59 %)
Correctitud (25 %)	Implementación precisa de algoritmos con manejo completo de casos borde.	Implementación correcta con buen manejo de casos comunes.	Implementación funcional con errores menores.	Implementación con errores significativos.
Eficiencia (20 %)	Código altamente optimizado con excelente rendimiento computacional.	Buen rendimiento y uso eficiente de recursos.	Rendimiento aceptable para casos comunes.	Código ineficiente con problemas de rendimiento.
Calidad (20 %)	Estructura modular excelente, alta legibilidad y manejo robusto de errores.	Buena estructura, código legible y manejo adecuado de errores.	Estructura básica funcional con legibilidad aceptable.	Código desorganizado o difícil de mantener.
Funcionalidad (10 %)	Implementación completa de todas las funcionalidades con optimizaciones adicionales.	Implementación de la mayoría de funcionalidades con buenas optimizaciones.	Implementación básica con optimizaciones simples.	Funcionalidades incompletas o sin optimizaciones.
Innovación y creatividad (10 %)	Soluciones altamente creativas con enfoques originales y extensiones innovadoras.	Buenas ideas originales que mejoran la implementación estándar.	Algunas ideas interesantes que complementan el enfoque básico.	Implementación estándar sin elementos creativos o innovadores.
Documentación y pruebas (15 %)	Documentación exhaustiva y conjunto completo de pruebas para todos los casos.	Buena documentación y pruebas adecuadas para casos principales.	Documentación básica y pruebas limitadas.	Documentación insuficiente o ausencia de pruebas.

Cuadro 1: Rúbrica para la evaluación del código

Criterio	Excelente (90-100 %)	Bueno (75-89 %)	Satisfactorio (60-74 %)	Insuficiente (0-59 %)
Originalidad (10 %)	Texto completamente original con citas adecuadas.	Texto mayormente original con referencias correctas.	Texto original con algunas citas imprecisas.	Plagio parcial o total (descalificación).
Análisis teórico (25 %)	Fundamentación matemática rigurosa y completa.	Buen análisis teórico con base matemática sólida.	Análisis teórico básico pero correcto.	Análisis superficial o con errores conceptuales.
Experimentación (20 %)	Metodología rigurosa con análisis estadístico detallado de optimizaciones.	Buena metodología con análisis adecuado de resultados.	Experimentos básicos con análisis simple.	Experimentos insuficientes o mal analizados.
Estructura y organización (25 %)	Estructura lógica impecable con secciones perfectamente integradas.	Buena organización con secuencia coherente de contenidos.	Estructura adecuada con algunas inconsistencias menores.	Organización deficiente o estructura ilógica.
Claridad y precisión técnica (20 %)	Explicaciones precisas con terminología técnica perfecta.	Buena claridad conceptual y uso adecuado de términos técnicos.	Explicaciones comprensibles con algunos errores menores.	Explicaciones confusas o uso incorrecto de terminología.

Cuadro 2: Rúbrica para la evaluación del informe técnico

Situación	Factor	Descripción
No presentación	0.0	La no realización de la presentación implica un factor 0, anulando completamente la nota del informe técnico.
Presentación deficiente	0.5	Presentación desorganizada, explicaciones confusas, incapacidad para responder preguntas básicas.
Presentación regular	0.7	Presentación estructurada pero con explicaciones superficiales y respuestas parciales a las preguntas.
Presentación adecuada	1.0	Presentación clara con buena estructura, explicaciones adecuadas y respuestas satisfactorias.
Presentación excelente	1.2	Presentación sobresaliente con dominio excepcional del tema.

Cuadro 3: Factores multiplicadores para la evaluación del informe técnico