|  |  |
| --- | --- |
|  | **Arquitectura de Computadores - Informe del Trabajo Final** |

Juan Manuel Díaz Moreno - A00394477

Valentina Gómez Rivera - A00398790

Alejandro Londoño Bermúdez - A00395978

Santiago Valencia García - A00395902

Danna Valentina López Muñoz – A00395625

**Análisis del Impacto de la Localidad de Caché en el Rendimiento de Algoritmos de Multiplicación de Matrices**

27 de noviembre de 2024

Magíster Carlos Andrés Díaz Andrade

TABLA DE CONTENIDO

[INTRODUCCIÓN 2](#_Toc1619390208)

[DEFINICIÓN DEL PROBLEMA Y OBJETIVOS 3](#_Toc2046431491)

[PROBLEMA 3](#_Toc104944309)

[OBJETIVOS 4](#_Toc1273109654)

[OBJETIVO GENERAL: 5](#_Toc278189387)

[OBJETIVOS ESPECÍFICOS: 5](#_Toc1761890362)

[MARCO TEÓRICO 5](#_Toc1084327192)

[FACTORES PRIMARIOS 12](#_Toc705253224)

[FACTORES SECUNDARIOS 16](#_Toc1929994316)

[UNIDAD EXPERIMENTAL 16](#_Toc1871604220)

[METODOLOGÍA DE ANÁLISIS 28](#_Toc1452353263)

[REGISTRO Y ORGANIZACION DE DATOS 29](#_Toc1953745236)

[LAPTOP-VK519CAP 29](#_Toc850454755)

[ANÁLISIS DE DATOS 30](#_Toc991095591)

[COMPARACIÓN GENERAL ENTRE ALGORITMOS 30](#_Toc1611153267)

[ANOVA 31](#_Toc1705150962)

[COMPARACIÓN DE SIMILITUD 31](#_Toc964229907)

[LAPTOP- OGEUFKBF 34](#_Toc351159390)

[ANÁLISIS DE DATOS 35](#_Toc1863185422)

[COMPARACIÓN GENERAL ENTRE ALGORITMOS 35](#_Toc1242083950)

[ANOVA 36](#_Toc1651232864)

[COMPARACIÓN DE SIMILITUD 36](#_Toc1168598675)

[COMPARACIÓN DE COMPUTADORES 39](#_Toc294917540)

[CONCLUSIONES 41](#_Toc785482107)

[REFERENCIAS 41](#_Toc2122627864)

# INTRODUCCIÓN

El rendimiento de los programas informáticos está intrínsecamente influenciado por la interacción entre la arquitectura de los computadores y la gestión de datos en memoria, siendo la memoria caché uno de los componentes más determinantes. Este experimento tiene como objetivo analizar cómo el principio de localidad afecta el desempeño de diferentes algoritmos de multiplicación de matrices, una operación fundamental en numerosos dominios de la computación. La importancia de este estudio radica en la posibilidad de identificar y optimizar patrones de acceso a memoria, minimizando cuellos de botella y mejorando el rendimiento general del sistema.

El propósito principal de este trabajo es evaluar el impacto de la localidad de caché en el rendimiento de seis versiones de algoritmos de multiplicación de matrices. Aunque comparten la misma complejidad algorítmica, presentan patrones de acceso a memoria diferenciados. Como complemento, se propondrá un esquema de multiplicación por bloques para mejorar el rendimiento del algoritmo menos eficiente, evaluando su efectividad en superar las limitaciones detectadas. Este enfoque está fundamentado en los principios teóricos de la jerarquía de memoria y el análisis de acceso eficiente a datos.

El experimento se desarrolla en dos fases principales. En la primera, se medirá el desempeño de seis algoritmos de multiplicación de matrices (a, b, c, d, e y f) considerando diversos tamaños de matrices y tipos de datos (Float y Double), en arquitecturas de tipo ISA x86 y x64. Esta fase permitirá identificar las variaciones en el rendimiento en función de factores como el tamaño de la matriz, la precisión de los datos y la arquitectura subyacente. En la segunda fase, se implementará un enfoque de multiplicación por bloques en el algoritmo con peor desempeño, verificando su capacidad para optimizar el rendimiento observado.

Este análisis es crucial para entender cómo las características del hardware y la estructura de los algoritmos interactúan para influir en el rendimiento computacional. Los resultados obtenidos no solo aportarán evidencia sobre la relevancia del principio de localidad en la arquitectura de computadores, sino que también subrayarán la importancia del diseño eficiente de algoritmos en entornos prácticos y de investigación.

# DEFINICIÓN DEL PROBLEMA Y OBJETIVOS

## PROBLEMA

En los sistemas computacionales modernos, el rendimiento de los programas no solo depende del número de operaciones realizadas, sino también de la manera en que los datos son accedidos y gestionados en memoria. Este aspecto se vuelve especialmente crítico en algoritmos intensivos en cálculos, como la multiplicación de matrices, donde la ineficiencia en el acceso a memoria puede generar cuellos de botella significativos.

El problema principal radica en que los algoritmos con patrones de acceso a memoria no optimizados, que no aprovechan la localidad espacial y temporal proporcionada por la memoria caché, sufren una degradación notable en el rendimiento. Esto se traduce en tiempos de ejecución más largos debido al constante intercambio de datos entre la memoria principal y el procesador. Dado que la jerarquía de memoria es una característica central de las arquitecturas modernas, la falta de optimización en los patrones de acceso limita el desempeño computacional general.

Este estudio aborda el problema al analizar el rendimiento de seis algoritmos de multiplicación de matrices con distintos patrones de acceso a memoria, identificando sus limitaciones, y proponiendo un enfoque basado en la multiplicación por bloques como solución para mejorar la eficiencia. La pregunta central es: ¿Cómo afectan los patrones de acceso a memoria al rendimiento de los algoritmos de multiplicación de matrices, y qué tan efectivo es el esquema de optimización por bloques para superar estas deficiencias?

## OBJETIVOS

### OBJETIVO GENERAL:

Evaluar el impacto de los patrones de acceso a memoria en el rendimiento de algoritmos de multiplicación de matrices y diseñar una estrategia de optimización basada en la multiplicación por bloques para superar las limitaciones identificadas.

### OBJETIVOS ESPECÍFICOS:

1. Analizar el rendimiento de seis versiones de algoritmos de multiplicación de matrices en términos de tiempo de ejecución, utilizando diferentes tamaños de matrices y tipos de datos (Float y Double).
2. Identificar las diferencias en el rendimiento de los algoritmos relacionadas con sus patrones de acceso a memoria.
3. Proponer un esquema de multiplicación por bloques como optimización para los algoritmos menos eficientes.
4. Implementar y evaluar el esquema de optimización en términos de su efectividad para mejorar el uso de memoria caché y el tiempo de ejecución.
5. Comparar los resultados de los algoritmos optimizados y no optimizados para demostrar la validez del enfoque basado en bloques.

# MARCO TEÓRICO

**Diseño de Experimentos (DoE)**

***Variables de Diseño***

En el diseño de experimentos, las variables de diseño son aquellos parámetros que el investigador puede controlar y manipular durante el experimento. Estas pueden ser continuas, como el tamaño de las matrices o el tiempo de ejecución, o discretas, como la selección de diferentes tipos de algoritmos o arquitecturas de procesador.

***Variables de Respuesta***

La variable de respuesta es la métrica que cuantifica el resultado del experimento. En el contexto del proyecto, corresponde al tiempo de ejecución, rendimiento del sistema, o métricas específicas como la tasa de aciertos en caché.

***Factores Experimentales***

Los factores experimentales se dividen en primarios y secundarios. Los factores primarios son aquellos cuyo efecto sobre la variable de respuesta se desea estudiar directamente, como el tipo de algoritmo o la arquitectura del procesador. Los factores secundarios son variables que pueden afectar los resultados, pero no son el foco principal del estudio, como la temperatura del sistema o la carga del sistema operativo.

***Replicación y Diseño de Bloques***

La replicación en experimentos consiste en la repetición de pruebas bajo condiciones idénticas para estimar la variabilidad experimental y aumentar la precisión de los resultados. El diseño de bloques, por su parte, es una técnica que agrupa unidades experimentales similares para reducir la variabilidad no deseada y mejorar la precisión de las comparaciones entre tratamientos.

***Análisis de Varianza (ANOVA)***

El Análisis de la Varianza, conocido como ANOVA, es un método estadístico empleado para identificar si existen diferencias significativas entre las medias de tres o más grupos. Esta técnica es particularmente valiosa cuando se estudian los efectos de varios factores en una variable específica. Su aplicación en el Diseño de Experimentos (DoE) permite evaluar cómo estos factores influyen en los resultados y si las variaciones observadas son atribuibles al azar o a las condiciones experimentales controladas.

El estadístico principal de ANOVA, *F*, se calcula mediante la relación entre la variabilidad entre grupos y la variabilidad dentro de los grupos:

Donde:

* *F*: Valor de prueba utilizado para tomar decisiones estadísticas.
* MSA: Media de cuadrados asociada con las diferencias entre grupos.
* MSE: Media de cuadrados asociada con las diferencias dentro de los grupos.

***Aplicaciones de ANOVA***

1. ***Análisis de medias:*** Permite determinar si las diferencias entre las medias de varios grupos son significativas.
2. ***Estudio de factores:*** Evalúa cómo uno o más factores independientes afectan una variable dependiente.
3. ***Diferenciación de grupos:*** Identifica entre qué grupos existen diferencias importantes en sus medias.

***Tiempo Normalizado***

El tiempo normalizado es una métrica utilizada para comparar la eficiencia de diferentes algoritmos o implementaciones al resolver problemas de tamaño o complejidad variable. Normalizar el tiempo de ejecución permite realizar comparaciones justas independientemente del tamaño del problema.

La fórmula para el tiempo normalizado es:

Donde:

* ​: Tiempo de ejecución real medido.
* : Función que describe la cantidad de operaciones en función del tamaño del problema N*.*

**Jerarquía de Memoria**

***Memoria Cache***

La memoria caché es un componente de almacenamiento rápido que actúa como intermediario entre el procesador y la memoria principal. Su diseño se basa en los principios de localidad y busca reducir el tiempo promedio de acceso a memoria. La caché se organiza en múltiples niveles, cada uno con diferentes características de capacidad y velocidad.

***Organización de la Memoria Caché***

La memoria caché está estructurada en bloques o líneas de caché, donde cada bloque almacena datos transferidos desde la memoria principal. La organización puede clasificarse en tres esquemas principales:

* Directamente Asociativa: Cada bloque de memoria principal se mapea en un único bloque de caché. Es simple y rápida, pero puede generar conflictos de reemplazo.
* Totalmente Asociativa: Cualquier bloque de memoria principal puede almacenarse en cualquier posición de la caché, maximizando la flexibilidad, pero incrementando la complejidad de búsqueda.
* Asociativa por Conjuntos: Divide la caché en un número fijo de conjuntos, y cada conjunto permite varios bloques. Este enfoque balancea flexibilidad y simplicidad.

***Impacto del Tamaño de la Caché y los Bloques***

El tamaño total de la caché y el tamaño de sus bloques afectan significativamente el rendimiento:

* Tamaño de la caché: Una caché más grande puede almacenar más datos y reducir los fallos de caché (cache misses). Sin embargo, incrementa el costo y la latencia de acceso.
* Tamaño de los bloques: Bloques más grandes aprovechan mejor la localidad espacial, pero si el tamaño es excesivo, pueden desperdiciarse datos innecesarios y aumentar la tasa de fallos por conflictos.
* Relación entre tamaño y jerarquía: En sistemas multinivel, las decisiones sobre el tamaño y los bloques deben equilibrar la capacidad, la velocidad de acceso y el costo energético.

***SRAM (Static Random Access Memory)***

La memoria SRAM utiliza flip-flops para almacenar información, manteniendo los datos mientras haya alimentación eléctrica. Se caracteriza por su alta velocidad de acceso y es utilizada principalmente en memorias caché. Su principal desventaja es el alto costo por bit y la menor densidad de almacenamiento en comparación con DRAM.

***DRAM (Dynamic Random Access Memory)***

La DRAM emplea condensadores para almacenar información, requiriendo refresco periódico para mantener los datos. Ofrece mayor densidad de almacenamiento y menor costo por bit que la SRAM, pero con mayores tiempos de acceso.

***Hits y Misses en la Memoria Caché***

1. ***Hit (Acierto):*** Ocurre cuando los datos solicitados por el procesador ya están almacenados en la caché. En este caso, el tiempo de acceso a la memoria es mínimo, ya que se evita la búsqueda en la memoria principal.
   * **Tiempo de acceso en caso de hit:**

Donde es el tiempo de acceso a la memoria caché.

1. ***Miss (Fallo):*** Sucede cuando los datos solicitados no se encuentran en la caché, obligando a buscarlos en la memoria principal. Esto introduce una penalización conocida como penalización por fallo (miss penalty), que incluye el tiempo necesario para transferir los datos desde la memoria principal hacia la caché.
   * **Tiempo de acceso en caso de miss:**

***Tasa de Hits y Misses***  
El rendimiento de la caché se mide en términos de las tasas de hit y miss:

* ***Tasa de hits (HH):*** Proporción de accesos a memoria que resultan en un hit. Se calcula como:
* ***Tasa de misses (MM):*** Proporción de accesos a memoria que resultan en un miss. Dado que H+M=1*H*+*M*=1, se calcula como:

***Tiempo Promedio de Acceso a Memoria (AMAT)***  
El tiempo medio de acceso a memoria es una métrica que combina el tiempo de acceso exitoso a caché (hit time) con el tiempo adicional requerido en caso de fallo (miss penalty), ponderados por la tasa de fallos. Esta métrica es fundamental para evaluar el rendimiento del sistema de memoria.

Se calcula combinando los tiempos de hit y miss:

Sustituyendo M=1−H*M*=1−*H*, también puede expresarse como:

**Principios de Localidad**

***Localidad Temporal***

La localidad temporal establece que las referencias a memoria recientemente utilizadas tienen alta probabilidad de ser accedidas nuevamente en el futuro cercano. Este principio fundamenta el almacenamiento de datos recientemente accedidos en la caché y afecta directamente al rendimiento del sistema.

**Localidad Espacial**

La localidad espacial indica que las referencias a memoria cercanas a una referencia actual tienen alta probabilidad de ser accedidas próximamente. Este principio justifica la transferencia de bloques contiguos de memoria a la caché y es crucial en el diseño de sistemas de memoria.

**Multiplicación de Matrices**

**Algoritmos de Multiplicación**

La multiplicación de matrices puede implementarse de diversas formas, cada una con diferentes patrones de acceso a memoria. El algoritmo básico tiene complejidad O(n³) y realiza accesos a memoria que pueden resultar subóptimos desde la perspectiva de la localidad de caché.

**Multiplicación por Bloques**

La multiplicación por bloques es una técnica de optimización que mejora la localidad de caché al dividir las matrices en sub-matrices más pequeñas. Esta técnica mantiene la complejidad algorítmica O(n³) pero reduce significativamente los fallos de caché, mejorando el rendimiento global.

**Arquitecturas x86 y x64**

***Arquitectura x86***

La arquitectura x86 de 32 bits proporciona un espacio de direcciones limitado y un conjunto más reducido de registros. Su diseño prioriza la compatibilidad con software legado y presenta ciertas limitaciones en términos de capacidad de direccionamiento y rendimiento.

***Arquitectura x64***

La arquitectura x64 extiende el espacio de direcciones a 64 bits y aumenta el número de registros disponibles. Estas mejoras permiten manejar mayores cantidades de memoria y optimizar el rendimiento en aplicaciones modernas, especialmente en operaciones con datos de punto flotante.

***Técnicas de Optimización***

Las técnicas de optimización incluyen la reorganización de bucles, el desenrollado de bucles, y la alineación de datos. Estas estrategias buscan maximizar la utilización de la caché y minimizar los accesos a memoria principal, mejorando el rendimiento global del sistema.

# FACTORES PRIMARIOS

En este experimento, los factores primarios o de tratamiento son aquellos directamente manipulados para analizar su efecto en el rendimiento de los algoritmos de multiplicación de matrices. Estos factores se seleccionaron con base en su relevancia para la interacción entre los algoritmos y la jerarquía de memoria. Los factores primarios identificados son:

***Tamaño de las matrices:***

*Descripción:* Define el número de elementos en las matrices cuadradas a multiplicar.

*Rango de variación:* Los tamaños evaluados son 185×185, 450×450, 550×550, 650×650, 750×750, 900×900, 1160×1160, 1480×1480, 1700×1700 y 2000×2000.

*Propósito:* Determinar cómo el tamaño de las matrices afecta el uso de la memoria caché y, por ende, el rendimiento de los algoritmos.

***Tipo de dato:***

*Descripción:* Representa la precisión de los elementos almacenados en las matrices, afectando la cantidad de memoria requerida.

*Niveles:* Se trabajará con los tipos de datos Float (32 bits) y Double (64 bits).

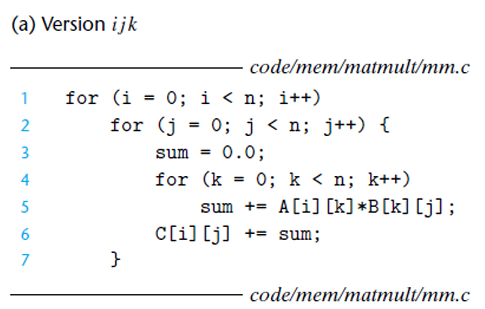
*Propósito:* Analizar el impacto del tamaño de los datos sobre el rendimiento, dado que los elementos de mayor precisión ocupan más memoria y afectan la capacidad de almacenamiento en caché.

***Patrón de acceso a memoria del algoritmo:***

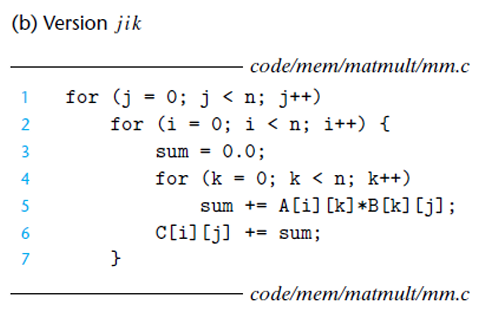
*Descripción:* Representa la secuencia en que los elementos de las matrices son accedidos durante la multiplicación.

*Niveles:* Se evaluarán seis algoritmos (a, b, c, d, e y f), cada uno con un patrón de acceso distinto. Los algoritmos por evaluar son los siguientes:

* Algoritmo (a) ijk:



* Algoritmo (b) jik:



* Algoritmo (c) jki:

Texto, Carta

Descripción generada automáticamente

* Algoritmo (d) kji:

Texto, Carta

Descripción generada automáticamente

* Algoritmo (e) kij:

Texto, Carta

Descripción generada automáticamente

* Algoritmo (f) ikj:

Texto, Carta

Descripción generada automáticamente

*Propósito:* Identificar cuál patrón aprovecha mejor las características de localidad espacial y temporal en la jerarquía de memoria.

# FACTORES SECUNDARIOS

Los factores secundarios son variables que podrían influir en los resultados, pero no forman parte de los aspectos directamente analizados. Para minimizar su impacto, se implementarán las siguientes estrategias de control:

**Arquitectura del hardware:**

*Descripción:* La configuración del hardware, como el tamaño de la caché y el conjunto de instrucciones, influye directamente en el rendimiento.

*Control:* Realizar pruebas en arquitecturas estándar y documentadas (ISA x86 y x64), manteniendo constante la configuración del hardware para asegurar resultados comparables.

**Cargas de trabajo adicionales en el sistema:**

*Descripción:* Procesos en ejecución en paralelo al experimento podrían interferir con los resultados.

*Control:* Las pruebas se realizarán en un sistema dedicado o con el mínimo de procesos activos, asegurando que los recursos del hardware estén disponibles exclusivamente para el experimento.

# UNIDAD EXPERIMENTAL

Para la realización del experimento, se seleccionaron como unidades experimentales los equipos LAPTOP-OGEUFKBF y LAPTOP-VK519CAP, los cuales fueron empleados para llevar a cabo las pruebas y recopilar los datos necesarios para el análisis. Esta elección se realizó con base en las características técnicas y funcionales de dichos equipos, incluyendo el tipo de procesador utilizado en cada uno, lo que garantizó su adecuación a los objetivos específicos planteados en el estudio.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración Procesador LAPTOP-OGEUFKBF

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración Procesador LAPTOP-VK519CAP

**4. Diseño Experimental**

4.1 **Elección del diseño experimental**

Para el análisis, se seleccionará un diseño experimental factorial completo. Este diseño es adecuado porque permite estudiar el efecto de múltiples factores y sus interacciones de forma simultánea. Los factores en este experimento son:

**Versión (Version): Niveles:** verA, verB, verC, verD, verE, verF.

**Tipo de dato (Data Type): Niveles:** float, double.

**Arquitectura del conjunto de instrucciones (ISA): Niveles:** X64, X86.

**Tamaño del dato (Size): Múltiples niveles:** {185, 450, 550, 650, 750, 900, 1160, 1480, 1700, 2000}.

Este diseño factorial completo considera todas las combinaciones posibles entre los niveles de los factores. Es importante porque permite evaluar tanto los efectos principales como las interacciones entre los factores.

**Justificación de la elección del diseño:**

**Hipótesis planteada:** Se busca determinar cómo las distintas configuraciones de versión, tipo de dato, arquitectura y tamaño afectan el desempeño. El diseño factorial completo es adecuado para identificar tanto los efectos individuales de cada factor como sus posibles interacciones.

**Factores identificados:** Ya que todos los factores son relevantes y no hay niveles anidados, un diseño factorial permite estudiar cada combinación sin pérdida de información.

**Restricciones existentes:** No hay restricciones explícitas sobre la aleatorización ni el número de pruebas; sin embargo, el número de tratamientos es grande, lo que sugiere que se puede dividir el análisis en bloques para facilitar la ejecución.

**Tamaño muestral:** La matriz original incluye repeticiones para ciertos tratamientos, asegurando robustez en los resultados.

**Organización de las pruebas:**

**Aleatorización:** Las combinaciones de los factores se ejecutarán en orden aleatorio para evitar sesgos causados por factores externos o sistemáticos (como variaciones en el hardware o el entorno durante las pruebas).

**Repeticiones:** Cada combinación se ejecutará al menos una vez. Para configuraciones clave (como verA-float-X64 y verB-double-X86), se incluirán réplicas adicionales para garantizar robustez en los datos y reducir la variabilidad experimental.

**Matriz de diseño:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Data Type** | **ISA** | **Size** |
| verA | float | X64 | 450 |
| verD | float | X64 | 185 |
| verA | float | X64 | 550 |
| verC | float | X64 | 185 |
| verF | float | X64 | 550 |
| verE | float | X64 | 550 |
| verD | float | X64 | 900 |
| verA | float | X64 | 1160 |
| verB | float | X64 | 1480 |
| verB | float | X64 | 900 |
| verF | float | X64 | 2000 |
| verE | float | X64 | 750 |
| verC | float | X64 | 1480 |
| verC | float | X64 | 550 |
| verB | float | X64 | 750 |
| verE | float | X64 | 650 |
| verA | float | X64 | 185 |
| verE | float | X64 | 1160 |
| verE | float | X64 | 1480 |
| verA | float | X64 | 750 |
| verE | float | X64 | 1700 |
| verA | float | X64 | 900 |
| verA | float | X64 | 650 |
| verD | float | X64 | 1700 |
| verD | float | X64 | 1480 |
| verF | float | X64 | 1480 |
| verD | float | X64 | 2000 |
| verE | float | X64 | 900 |
| verC | float | X64 | 750 |
| verF | float | X64 | 1700 |
| verC | float | X64 | 1700 |
| verB | float | X64 | 1160 |
| verE | float | X64 | 450 |
| verD | float | X64 | 1160 |
| verF | float | X64 | 1160 |
| verB | float | X64 | 450 |
| verA | float | X64 | 1700 |
| verE | float | X64 | 185 |
| verF | float | X64 | 900 |
| verC | float | X64 | 450 |
| verB | float | X64 | 2000 |
| verD | float | X64 | 750 |
| verF | float | X64 | 650 |
| verB | float | X64 | 1700 |
| verD | float | X64 | 550 |
| verB | float | X64 | 550 |
| verF | float | X64 | 185 |
| verA | float | X64 | 2000 |
| verD | float | X64 | 650 |
| verC | float | X64 | 650 |
| verB | float | X64 | 185 |
| verA | float | X64 | 1480 |
| verB | float | X64 | 650 |
| verC | float | X64 | 1160 |
| verE | float | X64 | 2000 |
| verF | float | X64 | 750 |
| verC | float | X64 | 900 |
| verF | float | X64 | 450 |
| verC | float | X64 | 2000 |
| verD | float | X64 | 450 |
| verE | float | X86 | 1160 |
| verB | float | X86 | 1160 |
| verC | float | X86 | 550 |
| verD | float | X86 | 750 |
| verE | float | X86 | 450 |
| verD | float | X86 | 1700 |
| verC | float | X86 | 900 |
| verB | float | X86 | 650 |
| verA | float | X86 | 550 |
| verF | float | X86 | 2000 |
| verB | float | X86 | 550 |
| verF | float | X86 | 185 |
| verB | float | X86 | 900 |
| verC | float | X86 | 1160 |
| verA | float | X86 | 185 |
| verA | float | X86 | 1480 |
| verA | float | X86 | 1700 |
| verE | float | X86 | 1480 |
| verA | float | X86 | 1160 |
| verD | float | X86 | 650 |
| verD | float | X86 | 1160 |
| verA | float | X86 | 900 |
| verD | float | X86 | 450 |
| verB | float | X86 | 1480 |
| verD | float | X86 | 550 |
| verC | float | X86 | 1700 |
| verE | float | X86 | 2000 |
| verB | float | X86 | 185 |
| verF | float | X86 | 1480 |
| verC | float | X86 | 1480 |
| verF | float | X86 | 450 |
| verF | float | X86 | 650 |
| verB | float | X86 | 2000 |
| verE | float | X86 | 185 |
| verB | float | X86 | 450 |
| verC | float | X86 | 750 |
| verC | float | X86 | 2000 |
| verF | float | X86 | 550 |
| verA | float | X86 | 2000 |
| verD | float | X86 | 1480 |
| verE | float | X86 | 1700 |
| verC | float | X86 | 450 |
| verA | float | X86 | 750 |
| verA | float | X86 | 650 |
| verB | float | X86 | 750 |
| verD | float | X86 | 185 |
| verB | float | X86 | 1700 |
| verD | float | X86 | 2000 |
| verE | float | X86 | 650 |
| verE | float | X86 | 550 |
| verD | float | X86 | 900 |
| verF | float | X86 | 1700 |
| verC | float | X86 | 185 |
| verF | float | X86 | 750 |
| verE | float | X86 | 900 |
| verE | float | X86 | 750 |
| verF | float | X86 | 1160 |
| verF | float | X86 | 900 |
| verC | float | X86 | 650 |
| verA | float | X86 | 450 |
| verE | double | X64 | 750 |
| verE | double | X64 | 1700 |
| verE | double | X64 | 450 |
| verA | double | X64 | 550 |
| verC | double | X64 | 2000 |
| verF | double | X64 | 900 |
| verB | double | X64 | 450 |
| verA | double | X64 | 900 |
| verD | double | X64 | 550 |
| verB | double | X64 | 750 |
| verC | double | X64 | 1160 |
| verD | double | X64 | 650 |
| verF | double | X64 | 1480 |
| verD | double | X64 | 1160 |
| verA | double | X64 | 1480 |
| verB | double | X64 | 1700 |
| verF | double | X64 | 2000 |
| verA | double | X64 | 650 |
| verC | double | X64 | 1700 |
| verC | double | X64 | 650 |
| verC | double | X64 | 900 |
| verB | double | X64 | 185 |
| verC | double | X64 | 550 |
| verA | double | X64 | 750 |
| verD | double | X64 | 450 |
| verE | double | X64 | 2000 |
| verB | double | X64 | 650 |
| verE | double | X64 | 1160 |
| verE | double | X64 | 650 |
| verF | double | X64 | 650 |
| verD | double | X64 | 1480 |
| verF | double | X64 | 1700 |
| verF | double | X64 | 550 |
| verD | double | X64 | 1700 |
| verB | double | X64 | 900 |
| verB | double | X64 | 1480 |
| verB | double | X64 | 2000 |
| verF | double | X64 | 185 |
| verE | double | X64 | 550 |
| verD | double | X64 | 900 |
| verB | double | X64 | 1160 |
| verA | double | X64 | 1700 |
| verE | double | X64 | 900 |
| verA | double | X64 | 185 |
| verC | double | X64 | 1480 |
| verA | double | X64 | 450 |
| verE | double | X64 | 1480 |
| verD | double | X64 | 2000 |
| verD | double | X64 | 185 |
| verE | double | X64 | 185 |
| verC | double | X64 | 185 |
| verC | double | X64 | 750 |
| verD | double | X64 | 750 |
| verF | double | X64 | 1160 |
| verB | double | X64 | 550 |
| verF | double | X64 | 750 |
| verC | double | X64 | 450 |
| verA | double | X64 | 2000 |
| verF | double | X64 | 450 |
| verA | double | X64 | 1160 |
| verF | double | X86 | 550 |
| verE | double | X86 | 550 |
| verD | double | X86 | 1480 |
| verF | double | X86 | 1480 |
| verE | double | X86 | 185 |
| verB | double | X86 | 450 |
| verC | double | X86 | 550 |
| verA | double | X86 | 900 |
| verC | double | X86 | 2000 |
| verB | double | X86 | 185 |
| verD | double | X86 | 1160 |
| verD | double | X86 | 2000 |
| verC | double | X86 | 450 |
| verE | double | X86 | 2000 |
| verF | double | X86 | 900 |
| verD | double | X86 | 900 |
| verE | double | X86 | 1160 |
| verD | double | X86 | 185 |
| verA | double | X86 | 1160 |
| verF | double | X86 | 185 |
| verF | double | X86 | 450 |
| verA | double | X86 | 1700 |
| verF | double | X86 | 650 |
| verE | double | X86 | 1700 |
| verB | double | X86 | 2000 |
| verC | double | X86 | 1160 |
| verF | double | X86 | 750 |
| verC | double | X86 | 750 |
| verA | double | X86 | 185 |
| verE | double | X86 | 900 |
| verF | double | X86 | 1700 |
| verA | double | X86 | 2000 |
| verD | double | X86 | 650 |
| verB | double | X86 | 900 |
| verD | double | X86 | 750 |
| verB | double | X86 | 650 |
| verD | double | X86 | 1700 |
| verE | double | X86 | 650 |
| verA | double | X86 | 450 |
| verF | double | X86 | 1160 |
| verA | double | X86 | 750 |
| verB | double | X86 | 750 |
| verA | double | X86 | 550 |
| verA | double | X86 | 650 |
| verB | double | X86 | 1700 |
| verA | double | X86 | 1480 |
| verC | double | X86 | 650 |
| verB | double | X86 | 550 |
| verB | double | X86 | 1160 |
| verC | double | X86 | 185 |
| verE | double | X86 | 750 |
| verC | double | X86 | 1700 |
| verD | double | X86 | 450 |
| verF | double | X86 | 2000 |
| verE | double | X86 | 1480 |
| verB | double | X86 | 1480 |
| verC | double | X86 | 1480 |
| verE | double | X86 | 450 |
| verC | double | X86 | 900 |
| verD | double | X86 | 550 |

**5. Experimentación Preliminar**

5.1 **Desarrollo de experimentos preliminares**

Durante el curso se llevaron a cabo análisis y experimentos que exploraron en profundidad cada uno de los factores a analizar en el experimento, especialmente el análisis de ANOVA durante la semana tres. Esto permitió adquirir experiencia con las variables experimentales, validar el correcto funcionamiento de los algoritmos y confirmar la precisión de los procedimientos empleados. Como resultado, se decidió no realizar experimentación preliminar adicional.

Se mantuvo un tamaño constante de 15 muestras para cada caso. Este número se determinó con base en los experimentos previos realizados en el curso, donde se demostró ser suficiente para analizar los 6 algoritmos seleccionados. Este tamaño de muestra no solo excede el mínimo requerido para obtener resultados estadísticamente significativos, sino que también garantiza la validez de los análisis.

En cuanto al entorno de ejecución, se tomó la precaución de evitar cualquier ruido externo que pudiera afectar los resultados. Esto implicó; deshabilitar aplicaciones y procesos en segundo plano en los equipos utilizados, y, asegurar que las condiciones de hardware y software fueran uniformes en ambos equipos de prueba.

Si bien no se realizaron experimentos preliminares específicos para este experimento, las conclusiones derivadas de los ejercicios del curso brindaron suficiente respaldo para proceder directamente con la experimentación principal, minimizando riesgos asociados a variables no identificadas o procedimientos imprecisos.

**6. Ejecución del Experimento**

6.1 **Procedimiento experimental**

El experimento se realizó siguiendo estrictamente el plan experimental previamente establecido para garantizar la validez de los resultados. Se utilizaron dos computadores configurados para minimizar cualquier posible interferencia externa, asegurando un entorno controlado. Ambos equipos estaban conectados a una fuente de energía estable desde el inicio del experimento para prevenir interrupciones por pérdida de energía.

Antes de iniciar las pruebas, se verificó que no hubiera aplicaciones en ejecución aparte del software necesario para procesar las matrices de diseño. Esto ayudó a evitar variaciones en el rendimiento causadas por la carga del sistema o la interferencia de programas en segundo plano. Además, se configuraron los computadores para evitar que entraran en estado de suspensión o ahorro de energía durante el experimento, garantizando así la continuidad del procesamiento.

Las pruebas se ejecutaron siguiendo un orden predefinido en la matriz de diseño, creado para minimizar posibles sesgos. Además, se monitorearon cuidadosamente los factores clave durante todo el proceso, incluyendo la estabilidad del hardware y el comportamiento del software. El monitoreo constante permitió identificar cualquier posible desviación en tiempo real.

# METODOLOGÍA DE ANÁLISIS

## REGISTRO Y ORGANIZACION DE DATOS

En el experimento, se recopilaron un total de 3600 datos por equipo. Estos datos se organizaron en el archivo Excel titulado "[Análisis.xlsx](https://icesiedu-my.sharepoint.com/:x:/g/personal/1107838593_u_icesi_edu_co/EQ-csLxRRLxCvSuQOxDD9FkBjJhbcKT954XNJToWj03rNg?e=jpkiAJ)", que se adjunta junto con este informe. Este documento incluye:

* Una hoja titulada "matriz\_diseño" que contiene la matriz de diseño y los factores experimentales.
* Una hoja titulada "datos\_1", que incluye los tratamientos correspondientes al equipo LAPTOP-VK519CAP.
* Una hoja titulada "datos\_2", que contiene los tratamientos del equipo LAPTOP-OGEUFKBF.

A cada conjunto de datos se les aplicó un análisis de regresión lineal, además de un ANOVA de cuatro vías, utilizando python en el cuaderno de trabajo "[Proyecto Final Arquicompu](https://colab.research.google.com/drive/1s-oNn17-OWUm1loEABkHONq7lAsDQh7M?usp=sharing)" en Google colab. Estos análisis permitieron evaluar la relación entre los factores y las respuestas del experimento, así como determinar la significancia estadística de las interacciones entre los factores.

A continuación, se presentan los resultados de la regresión lineal y análisis estadístico aplicada a los datos obtenidos del equipo LAPTOP-VK519CAP para posteriormente hacer lo mismo con los tratamientos obtenidos a partir del equipo LAPTOP-OGEUFKBF.

# LAPTOP-VK519CAP

## ANÁLISIS DE DATOS

Tabla

Descripción generada automáticamente

Gráfico, Gráfico de dispersión, Gráfico de cajas y bigotes

Descripción generada automáticamente

Gráfico, Gráfico de dispersión

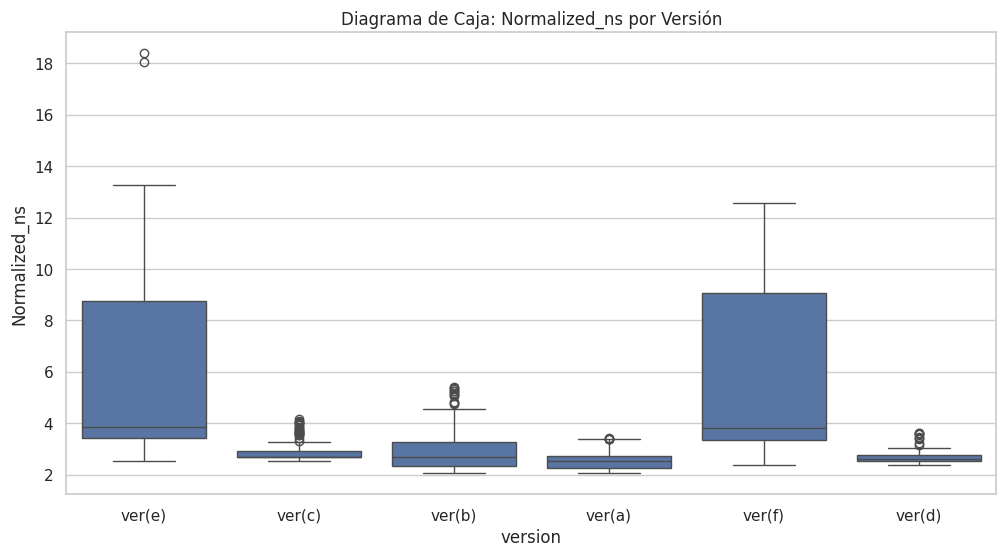
Descripción generada automáticamente

Gráfico

Descripción generada automáticamente

Se puede apreciar en la regresión que los algoritmos con mejor desempeño son el c y d, seguidos del a y b, mientras que los algoritmos con peor desempeño son el e y f, lo cuál sigue la teoría y nos permitirá agrupar los algoritmos entre los que tenga mayor similitud para su análisis.

## COMPARACIÓN GENERAL ENTRE ALGORITMOS



## ANOVA

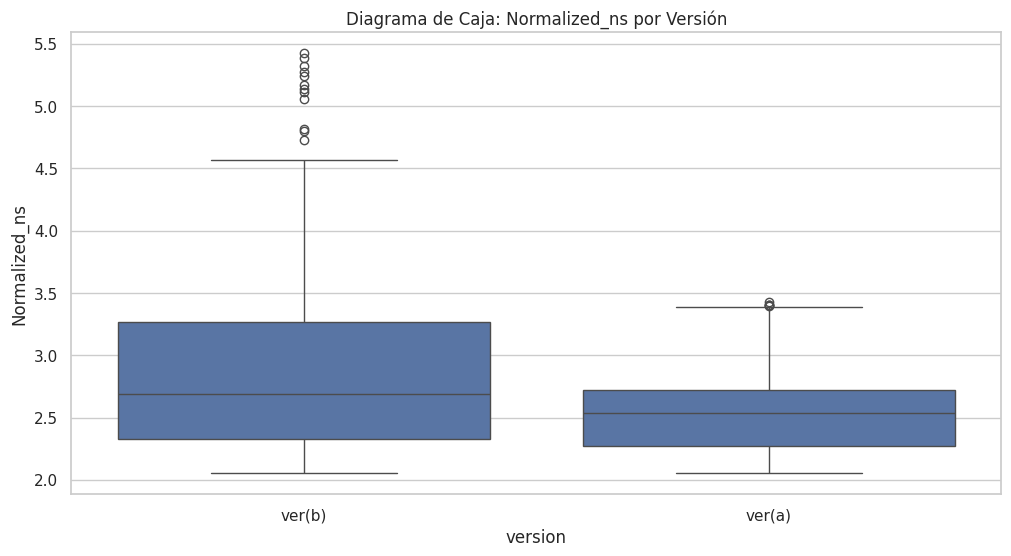
## 

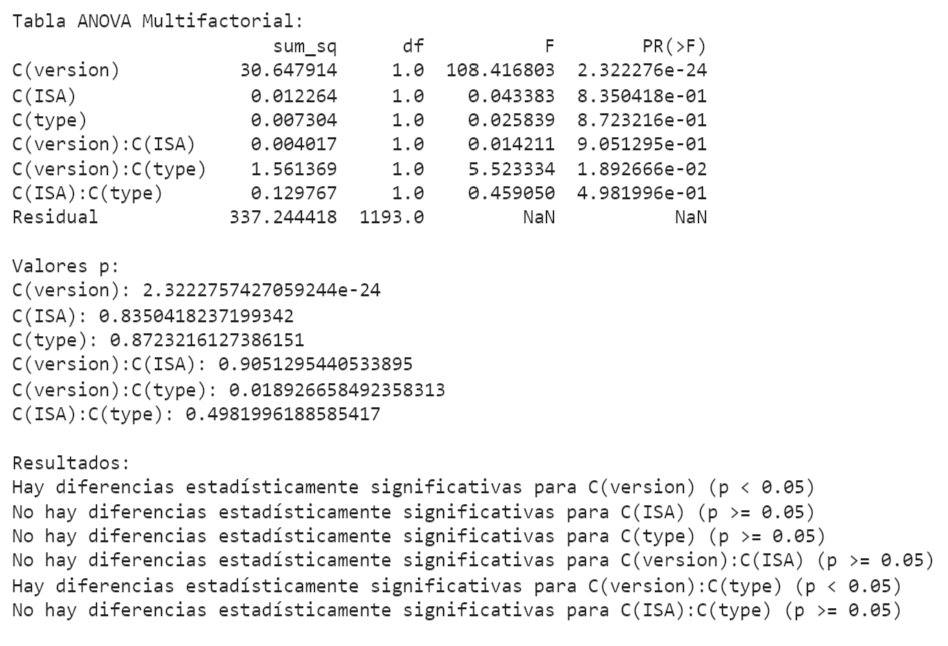
## COMPARACIÓN DE SIMILITUD

Comparando versiones ver(a) y ver(b):

*H0: No hay diferencias estadísticamente significativas entre los factores para las versiones de algoritmo a y b.*

*H1: Hay diferencias estadísticamente significativas entre los factores para las versiones de algoritmo a y b.*



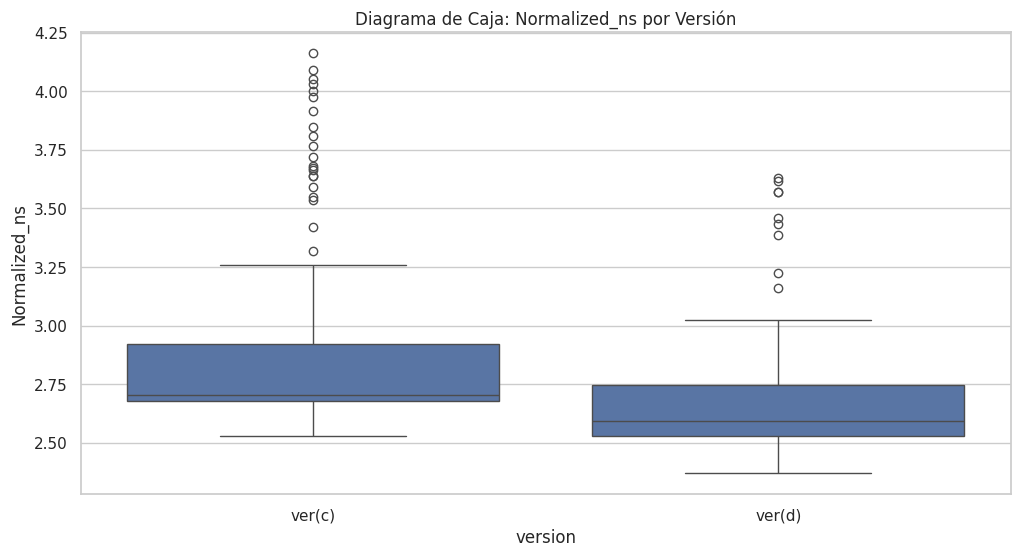


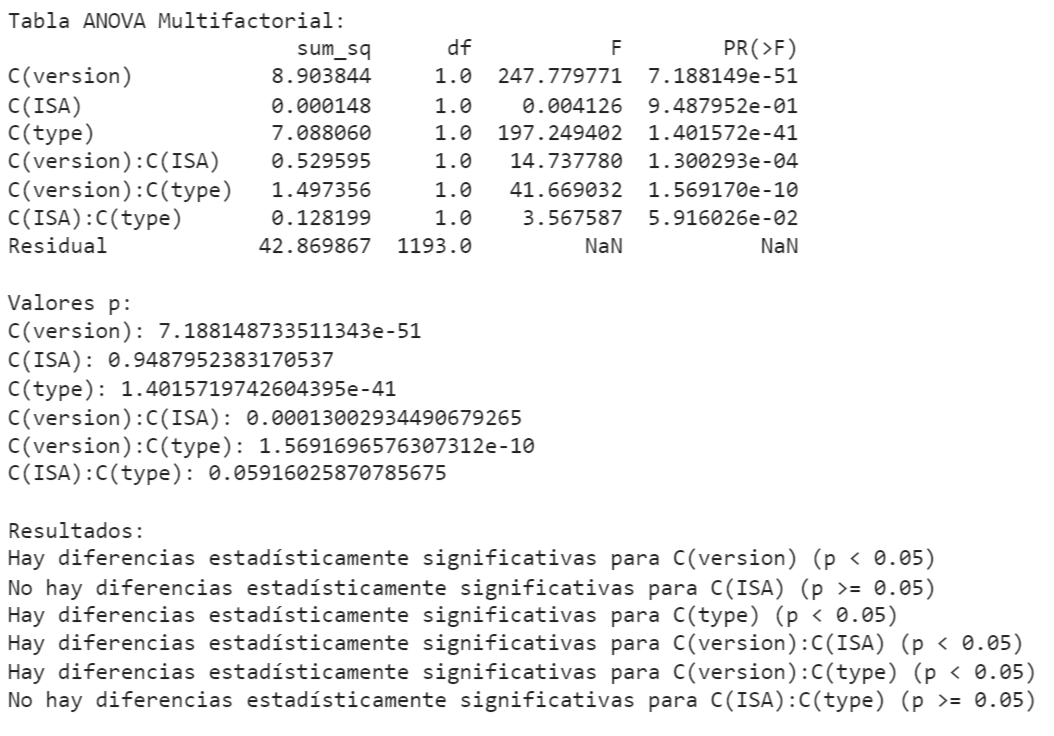
Se puede apreciar que el valor p para la *versión* es menor al alfa del 0,05 al considerar la interacción con el factor de tipo de dato, por lo que, se rechaza la hipótesis nula de que no hay diferencias estadísticamente significativas entre los factores para los algoritmos a y b.

Comparando versiones ver(c) y ver(d):

*H0: No hay diferencias estadísticamente significativas entre los factores para las versiones de algoritmo c y d.*

*H1: Hay diferencias estadísticamente significativas entre los factores para las versiones de algoritmo c y d.*

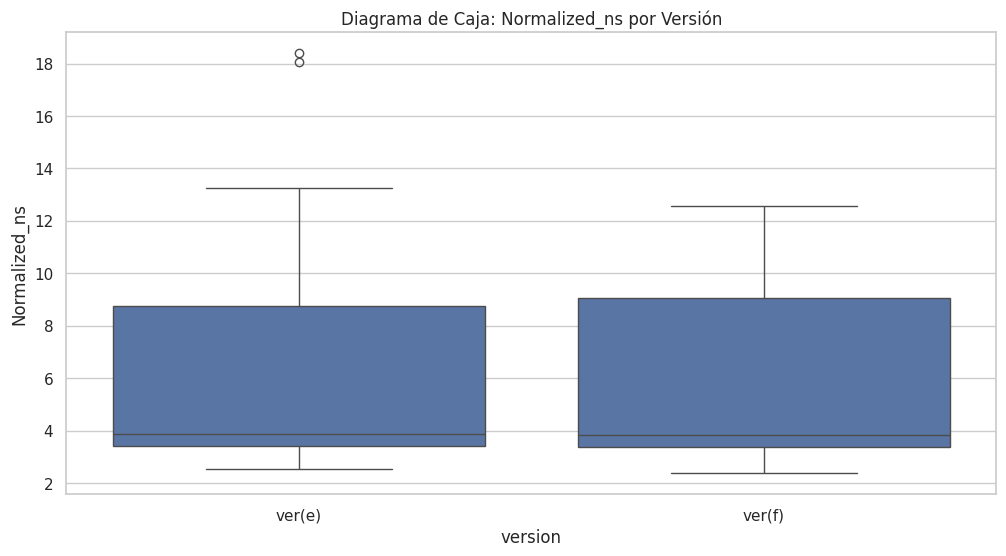


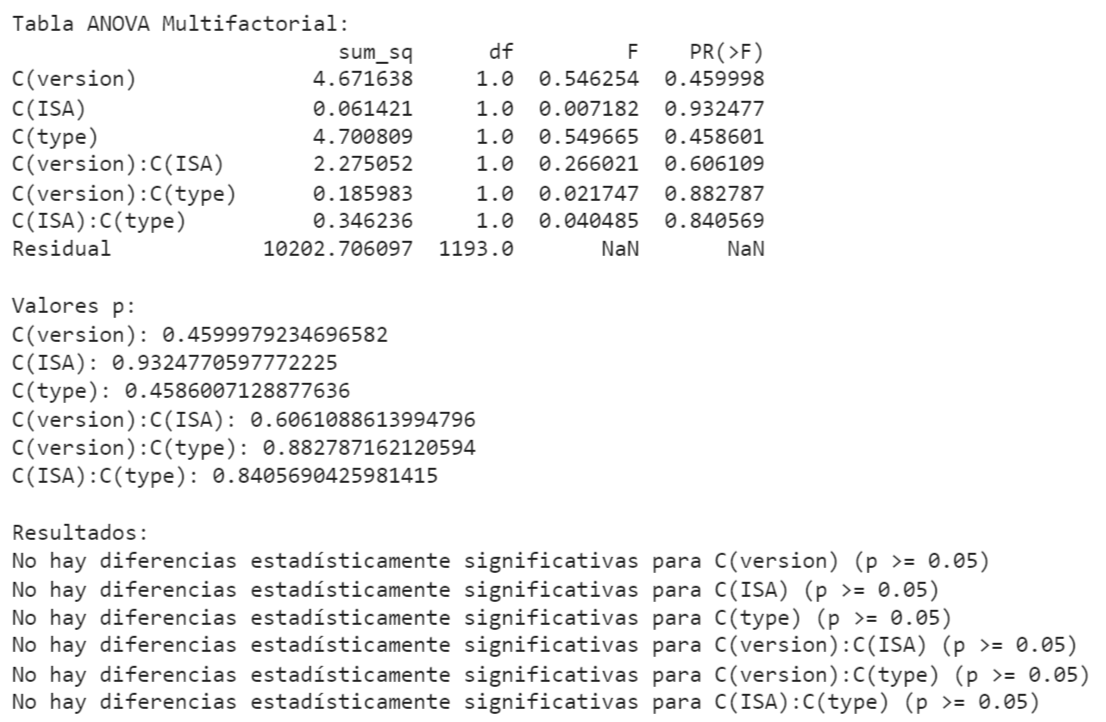


Se puede apreciar que el valor p para la *versión* es menor al alfa del 0,05 al considerar la interacción tanto con el tipo de dato como la arquitectura (ISA), por lo que, se rechaza la hipótesis nula de que no hay diferencias estadísticamente significativas entre los factores para los algoritmos c y d.

Comparando versiones ver(e) y ver(f)  
*H0: No hay diferencias estadísticamente significativas entre los factores para las versiones de algoritmo e y f.*

*H1: Hay diferencias estadísticamente significativas entre los factores para las versiones de algoritmo e y f.*

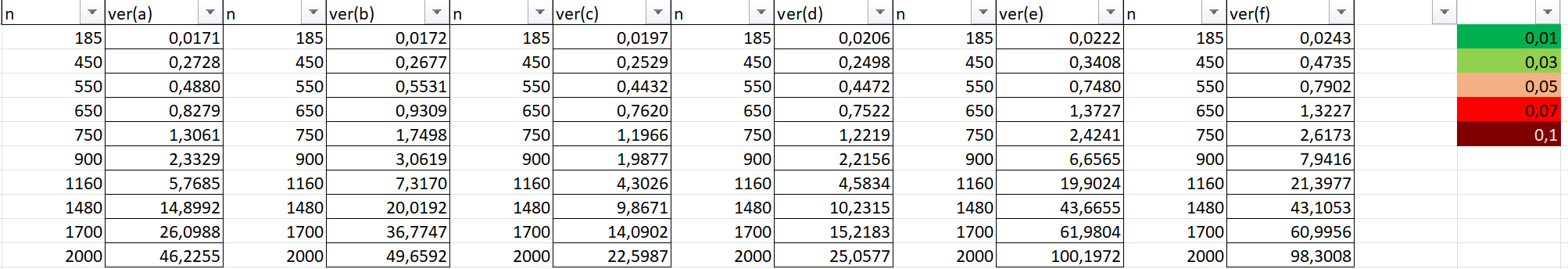


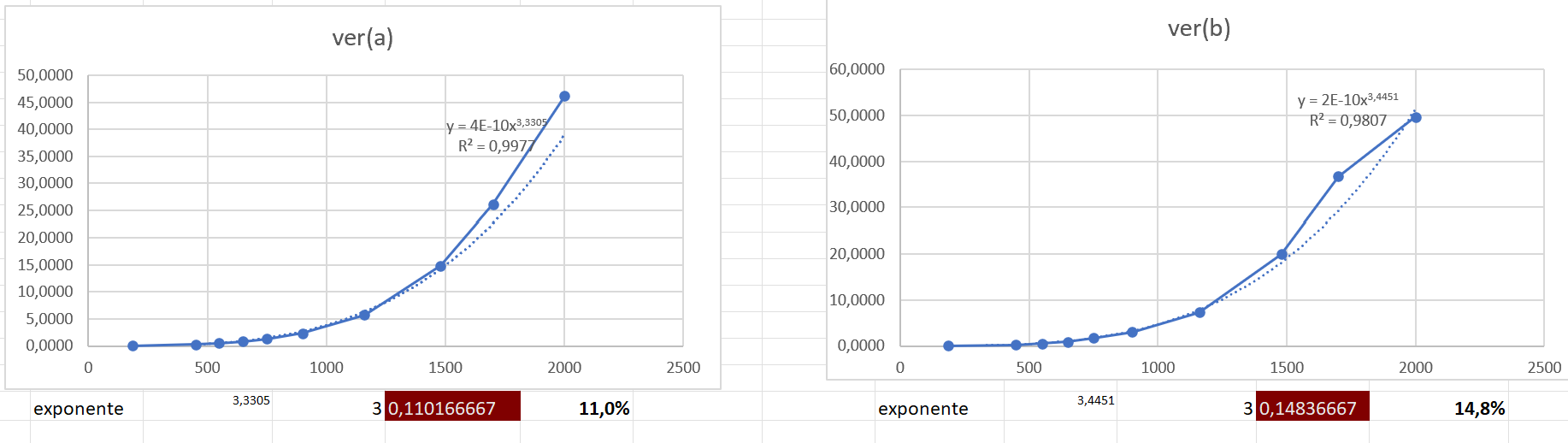


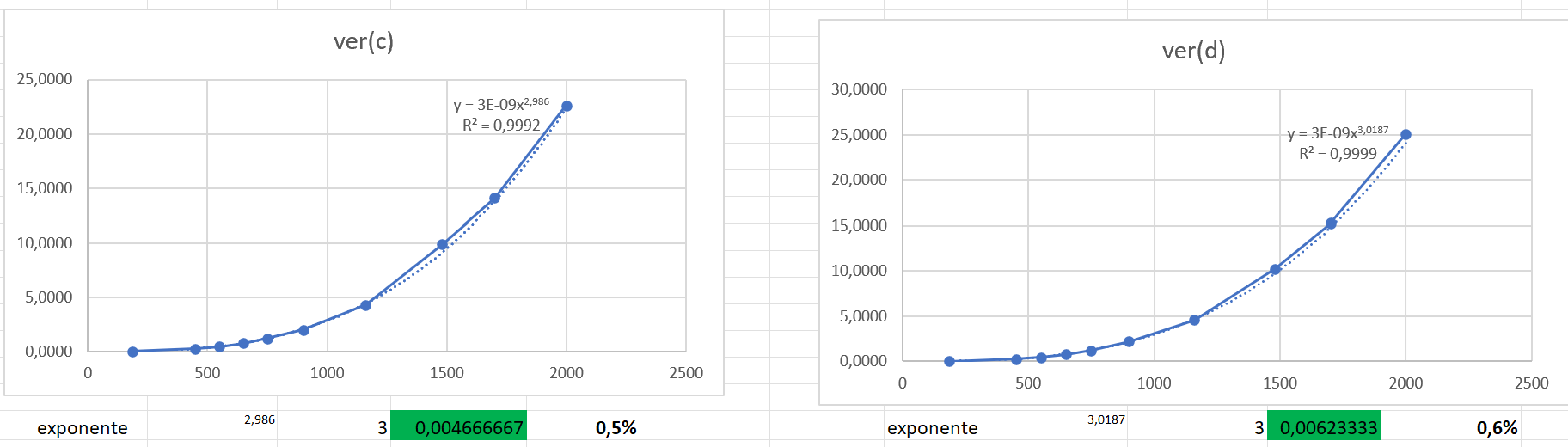
Se puede apreciar que el valor p de todos los factores es mayor al alfa del 0,05 y que al considerar la interacción entre factores, nunca se obtiene un alfa menor al 0,5 por lo que, se acepta la hipótesis nula de que no hay diferencias estadísticamente significativas entre los factores para los algoritmos e y f.

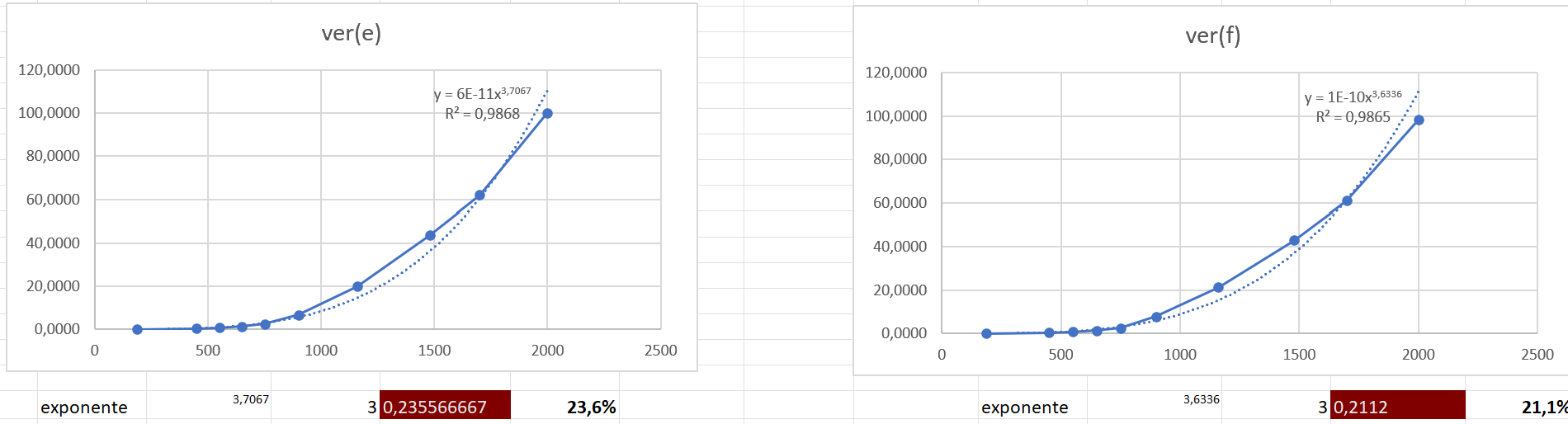
# LAPTOP- OGEUFKBF

## ANÁLISIS DE DATOS

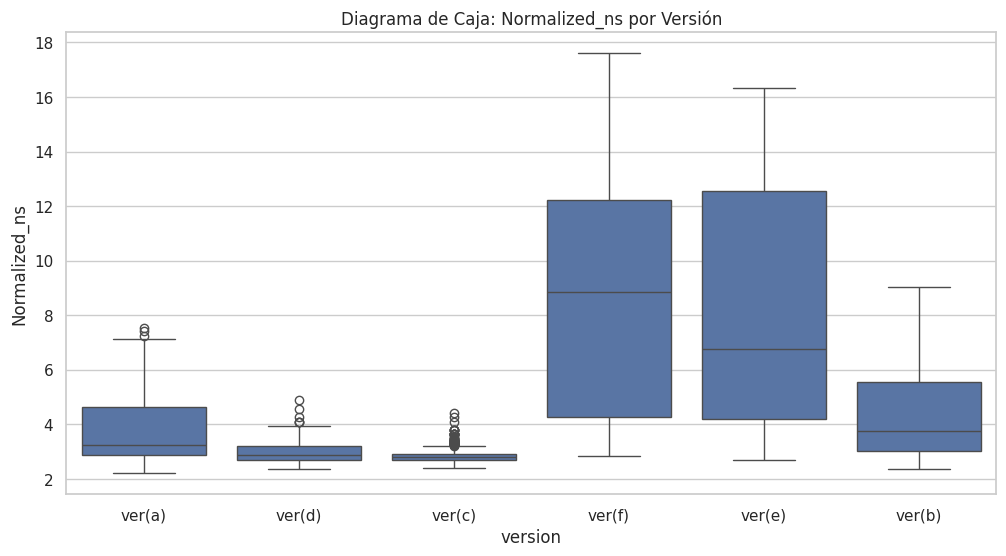




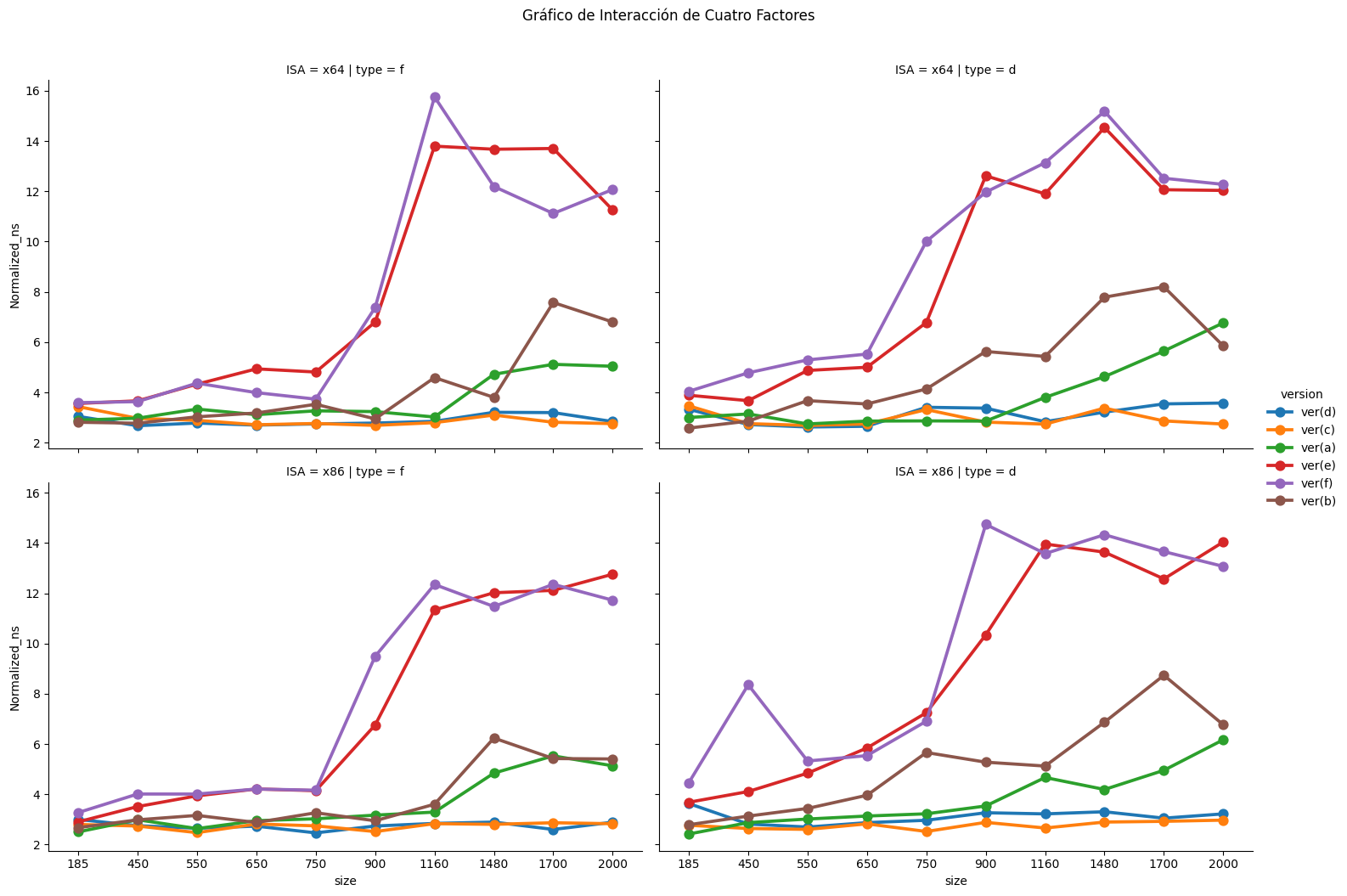


En las gráficas de regresión los algoritmos con mejor desempeño corresponden al c y d, los cuales presentan un ajuste más preciso. Por otro lado, los algoritmos a y b presentan un desempeño intermedio, por último, los algoritmos e y f muestran el peor desempeño con errores relativos significativamente más altos y menos precisión en sus ajustes.

## COMPARACIÓN GENERAL ENTRE ALGORITMOS



## ANOVA

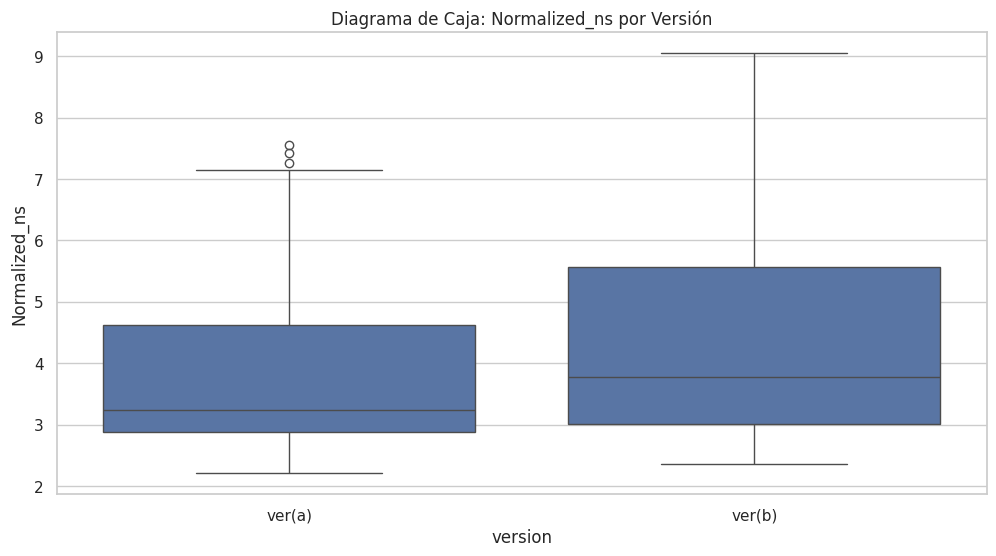


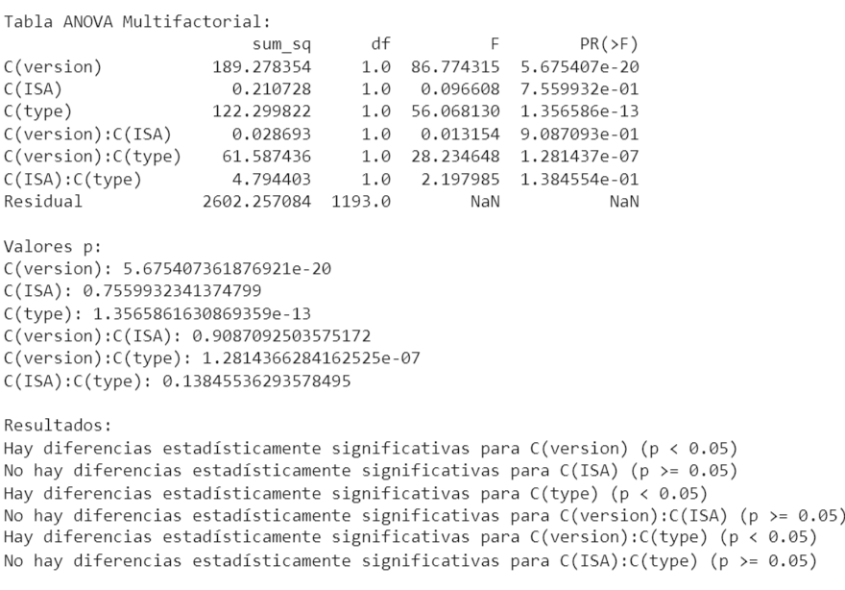
## COMPARACIÓN DE SIMILITUD

**Comparando versiones ver(a) y ver(b):**

*H0: No hay diferencias estadísticamente significativas entre los factores para las versiones de algoritmo a y b.*

*H1: Hay diferencias estadísticamente significativas entre los factores para las versiones de algoritmo a y b.*



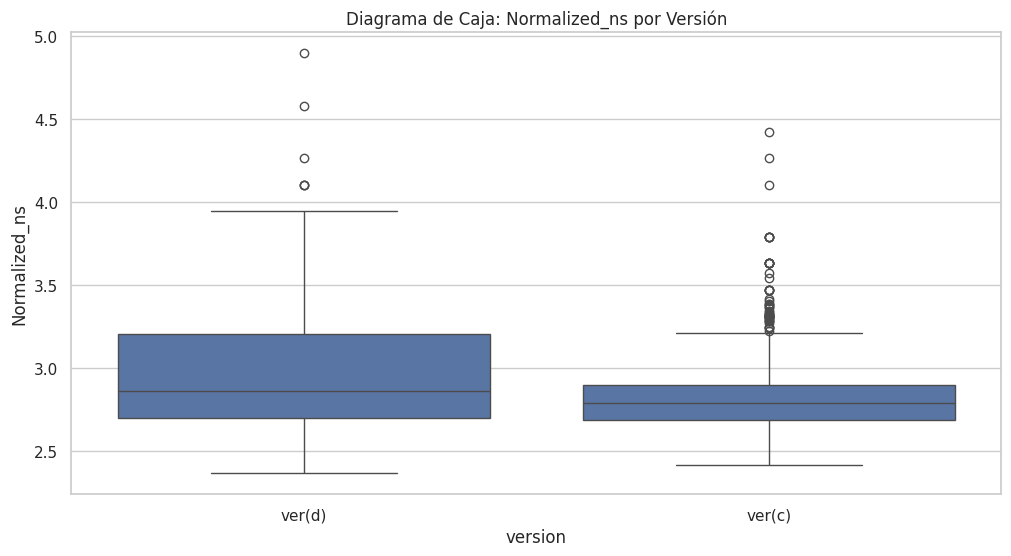


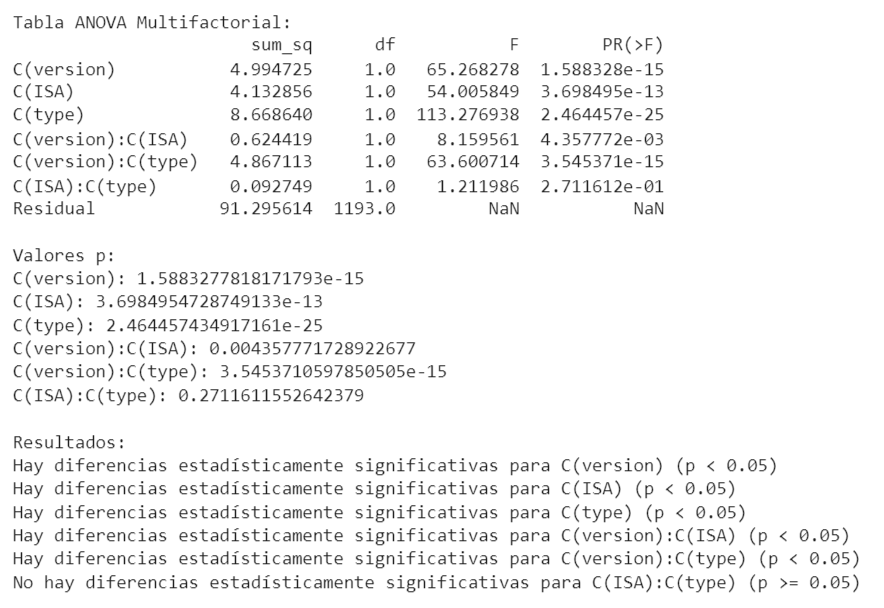
Como se puede ver, el resultado es p < 0.05, Esto indica que debemos rechazar la hipótesis nula, por lo tanto, hay diferencias estadísticamente significativas entre los factores para las versiones de los algoritmos a y b.

**Comparando versiones ver(c) y ver(d):**

*H0: No hay diferencias estadísticamente significativas entre los factores para las versiones de algoritmo c y d.*

*H1: Hay diferencias estadísticamente significativas entre los factores para las versiones de algoritmo c y d.*

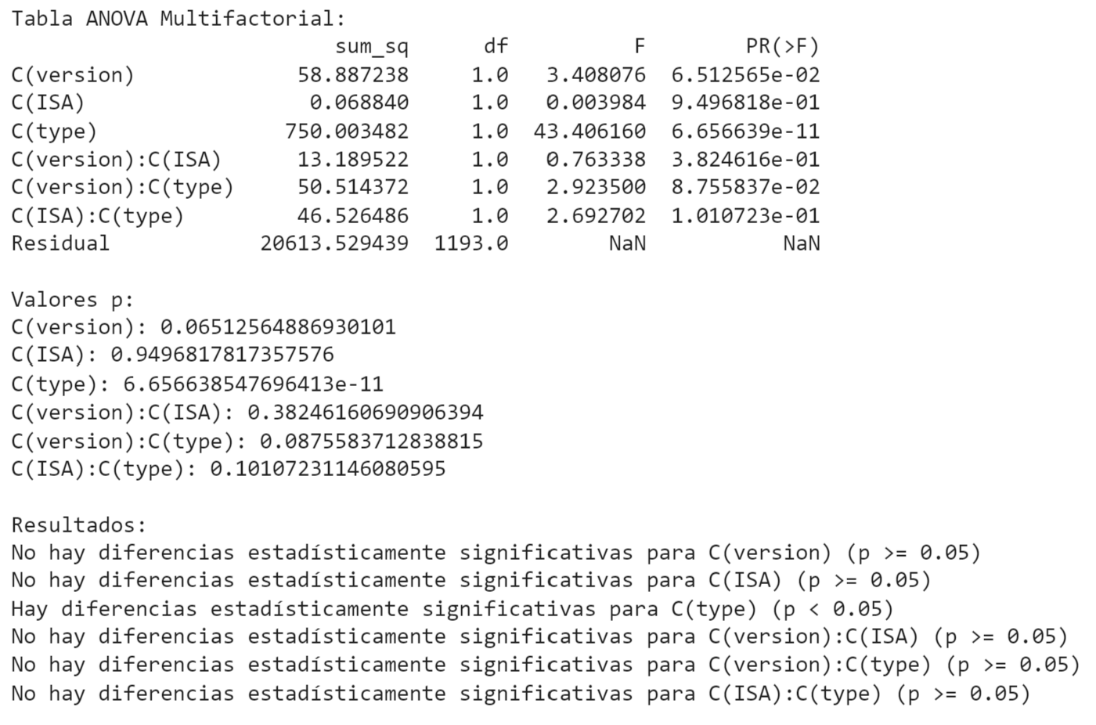
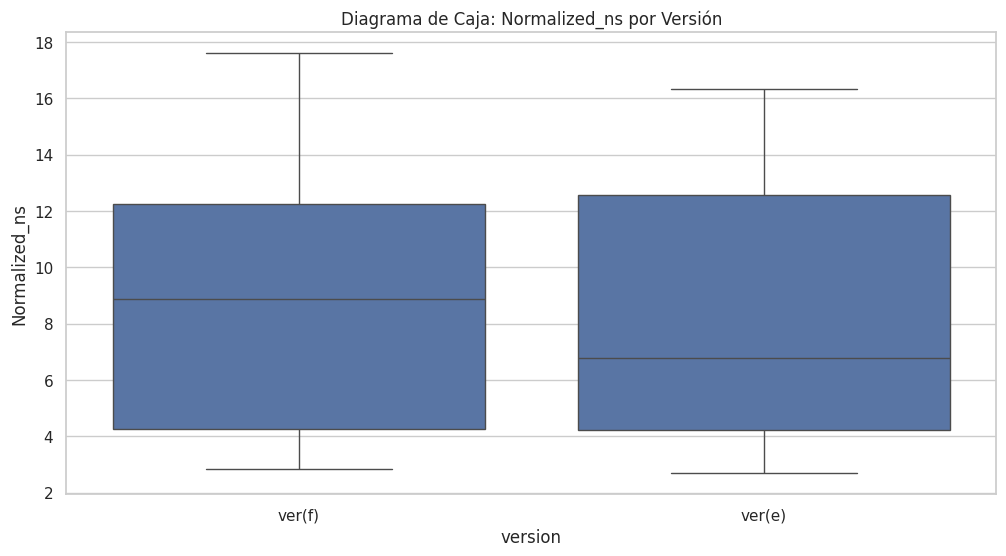




Como se puede ver, el resultado es p < 0.05, lo que indica que debemos rechazar la hipótesis nula, en especial en las interacciones entre la versión del algoritmo con la ISA y el tipo de dato, por lo tanto, hay diferencias estadísticamente significativas entre todos los factores para las versiones de los algoritmos d y c.

**Comparando versiones ver(e) y ver(f):**  
*H0: No hay diferencias estadísticamente significativas entre los factores para las versiones de algoritmo e y f.*

*H1: Hay diferencias estadísticamente significativas entre los factores para las versiones de algoritmo e y f.*



Se puede apreciar que el valor p < 0.05 para el tipo de dato, pero, al considerar la interacción entre factores, nunca se obtiene un alfa menor al 0,5 por lo que, se acepta la hipótesis nula de que no hay diferencias estadísticamente significativas entre los factores para los algoritmos e y f.

# COMPARACIÓN DE COMPUTADORES

A lo largo del análisis, se observó que los resultados de la regresión lineal y el ANOVA de cuatro vías fueron más favorables en el equipo **LAPTOP-VK519CAP** en comparación con **LAPTOP-OGEUFKBF**. A continuación, se presentan los principales hallazgos para cada conjunto de datos:

**Regresión lineal**

* **Desviaciones en las versiones de los algoritmos:**
  + Las versiones **e** y **f** (las de peor desempeño) tuvieron menores porcentajes de desviación en **LAPTOP-VK519CAP** con valores de 20.7% y 21.4%, respectivamente, en comparación con 23.6% y 21.1% en **LAPTOP-OGEUFKBF**.
  + Para las versiones **c** y **d** (las de mejor desempeño), **LAPTOP-OGEUFKBF** obtuvo porcentajes de desviación de 0.5% y 0.6%, ligeramente mejores que los 0.8% y 0.9% de **LAPTOP-VK519CAP**; sin embargo, esta diferencia no es estadísticamente significativa.
  + En las versiones **a** y **b** (intermedias), **LAPTOP-VK519CAP** mostró un mejor comportamiento, especialmente en la versión **a**, donde la desviación en **LAPTOP-OGEUFKBF** fue más del doble.

En general, **LAPTOP-VK519CAP** presentó porcentajes de desviación más consistentes y mejores para casi todas las versiones del algoritmo, salvo para la versión f, en la que ambos equipos tuvieron un desempeño similar.

**ANOVA de cuatro vías**

* **LAPTOP-VK519CAP:**
  + El análisis ANOVA mostró menores valores residuales en la suma de cuadrados entre versiones, lo que indica menor variación:
    - **Versiones a y b:** Residual de 337.24.
    - **Versiones c y d (mejor comportamiento):** Residual de 42.86.
    - **Versiones e y f (peor comportamiento):** Residual de 10,202.70.
  + El valor residual total de la suma de cuadrados fue de 229.99.
  + Los factores significativos (valor p < 0.05) fueron: versión, size (tamaño de la matriz) y type (tipo de algoritmo). El factor ISA no tuvo un efecto relevante de manera aislada, pero las interacciones entre ISA y los demás factores sí fueron significativas.
* **LAPTOP-OGEUFKBF:**
  + Se encontró mayor variación entre las versiones:
    - **Versiones a y b:** Residual de 2,602.25.
    - **Versiones c y d (mejor comportamiento):** Residual de 91.29.
    - **Versiones e y f (peor comportamiento):** Residual de 20,613.52.
  + El valor residual total de la suma de cuadrados fue de 1,012.60.
  + Todos los factores y sus interacciones resultaron significativos (valor p < 0.05), siendo los factores más influyentes la **versión del algoritmo** y el **tamaño de la matriz**, como lo indican sus valores F.

Por lo anterior, se puede destacar que el equipo **LAPTOP-VK519CAP** presentó menor variación y desviación en la mayoría de las versiones del algoritmo, lo que sugiere un mejor rendimiento general, especialmente en las versiones con peor comportamiento. Por otro lado, aunque **LAPTOP-OGEUFKBF** tuvo un desempeño ligeramente superior en las versiones óptimas, esta diferencia no es significativa en términos estadísticos.

En términos de ANOVA, ambos equipos demostraron que la versión del algoritmo y el tamaño de la matriz son factores críticos para el desempeño. Sin embargo, el equipo **LAPTOP-VK519CAP** destacó por tener valores residuales menores y mayor estabilidad entre los niveles de los factores.

**Juicio sobre los resultados:**

Los resultados del LAPTOP-VK519CAP reflejan menor desviación en las versiones con peor desempeño (e y f) y mayor estabilidad general. Esto sugiere que la arquitectura mejorada y la mayor eficiencia del 11300H permiten manejar mejor las tareas computacionalmente intensivas, reduciendo los errores acumulados o las variaciones introducidas por diferencias en la carga de trabajo o la gestión de recursos y aunque el equipo LAPTOP-OGEUFKBF mostró un desempeño ligeramente superior en versiones óptimas (c y d), esta ventaja no fue estadísticamente significativa, lo que refuerza la hipótesis de que el rendimiento general y la estabilidad del LAPTOP-VK519CAP están influenciados por las capacidades superiores del 11300H.

El Intel Core i5-11300H pertenece a la 11ª generación de procesadores Intel (Tiger Lake), mientras que el i5-10300H pertenece a la 10ª generación (Comet Lake). Los procesadores más recientes, como el 11300H, incorporan mejoras en la arquitectura, incluyendo optimizaciones en el manejo de tareas paralelas y un mejor uso de los recursos disponibles, lo que puede explicar la mayor estabilidad y menor desviación observada en los resultados del LAPTOP-VK519CAP. Además, el 11300H tiene una velocidad base de 3.10 GHz, superior a los 2.50 GHz del 10300H. Aunque ambos procesadores pueden alcanzar velocidades más altas, la frecuencia base más alta del 11300H lo hace más adecuado para manejar cargas de trabajo sostenidas, lo que puede haber beneficiado la consistencia en los datos analizados. A pesar de que la temperatura es un factor secundario que no se puede manejar durante el transcurso de este experimento, el i5-11300H tiene un TDP máximo de 35.0 W, en comparación con los 45.0 W del 10300H, lo que indica que el 11300H está diseñado para operar de manera más eficiente con menor generación de calor esto puede haber ayudado a mantener un rendimiento consistente bajo cargas intensivas y explicar de manera adicional por qué el LAPTOP-VK519CAP mostró menor variación entre las versiones de los algoritmos.

El experimentador emitirá un juicio basado en los resultados obtenidos. Se evaluarán las limitaciones del experimento y se propondrán escenarios alternativos o nuevas hipótesis para futuros experimentos, por ejemplo, mejorar un algoritmo determinando un parámetro de acuerdo con la interpretación de los resultados, en esta sección se puede realizar un experimento o una prueba para validar esa mejora.

# CONCLUSIONES

9.1 **Conclusión final y recomendaciones**  
El informe culminará con una conclusión clara y bien justificada, basada en el análisis de los resultados. Se deberán incluir recomendaciones sobre el curso de acción más adecuado, así como sugerencias para experimentos de seguimiento o para la mejora de los procedimientos empleados.

**Impacto del patrón de acceso en el rendimiento:**

El análisis de los datos confirmó que los patrones de acceso a memoria influyen significativamente en el rendimiento de los algoritmos de multiplicación de matrices. Los algoritmos con patrones más alineados a los principios de localidad espacial y temporal, como las versiones c y d, mostraron un desempeño considerablemente superior en comparación con las versiones e y f, las cuales sufrieron mayores penalizaciones por fallos de caché.

**Importancia del tamaño de las matrices:**

El tamaño de las matrices evaluadas tuvo un impacto directo en el tiempo de ejecución de los algoritmos. A medida que el tamaño aumentó, se observaron mayores penalizaciones asociadas al uso de la jerarquía de memoria, especialmente en algoritmos con acceso menos eficiente. Esto destaca la relevancia de optimizar patrones de acceso para manejar matrices de mayor escala.

**Influencia del tipo de dato y la arquitectura:**

Los datos demostraron que el tipo de dato utilizado (float o double) y la arquitectura del procesador (x86 o x64) también afectan el rendimiento, aunque en menor medida que el tamaño de las matrices o el patrón de acceso. Las configuraciones en x64 ofrecieron un mejor manejo de memoria para tamaños grandes, mientras que las operaciones en float fueron más rápidas pero menos precisas en general.

**Diferencias entre equipos experimentales:**

Los resultados indicaron diferencias en la estabilidad y rendimiento entre los equipos LAPTOP-VK519CAP y LAPTOP-OGEUFKBF. El equipo LAPTOP-VK519CAP mostró una menor desviación en las versiones con peor desempeño (e y f) y una mayor consistencia general. Esto sugiere que las diferencias en la arquitectura de los procesadores (11ª generación vs. 10ª generación) y en la frecuencia base influyeron en la capacidad de manejar cargas computacionales intensivas.

# REFERENCIAS

En esta sección, se incluirá una lista de todas las fuentes y referencias bibliográficas utilizadas a lo largo del experimento. Todas las citas deben estar en formato APA.