Imagen que contiene dibujo

El contenido generado por IA puede ser incorrecto.

UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

FACULTAD DE INGENIERÍA

TRABAJO PARCIAL

Ciclo:

2025-1

Curso:

Programación Concurrente y Distribuida (CC65)

Clasificación de Condición de Autos Usados Utilizando Programación Concurrente en Go

Integrantes:

Rodriguez Calla, Jair Stephano

Niño Suarez, David Joaquín

Llanos Llamoca, Carlos Daniel

Profesor:

Jara García, Carlos Alberto

Lima, 13 de mayo de 2025

Contenido

[1 Planteamiento del Problema 3](#_Toc198201900)

[2 Explicación del dataset Seleccionado 4](#_Toc198201901)

[3 Explicación del algoritmo y el uso de los mecanismos de sincronización utilizado. 6](#_Toc198201902)

[4 Presentar las pantallas del menú 8](#_Toc198201903)

[5 Explicación de las pruebas realizadas, resultados y pegar las imágenes de evidencia. 10](#_Toc198201904)

[6 Explicación de la Simulación Realizada con Promela y Evidencias 12](#_Toc198201905)

[7 Explicación del análisis usando spin, pegar las imágenes de evidencia 15](#_Toc198201906)

[Bibliografía 16](#_Toc198201907)

[Anexo 17](#_Toc198201908)

# Planteamiento del Problema

**Objetivo General**

Desarrollar una aplicación concurrente en el lenguaje Go que clasifique el nivel de riesgo de inversión en vehículos usados (Bajo, Moderado o Alto) utilizando un algoritmo de Árbol de Decisión, sin depender de librerías externas. Esta clasificación se realizará en base a características relevantes de cada vehículo, tales como la marca, el modelo, el año de fabricación, el precio, el kilometraje y su condición actual.

**Problema que se Quiere Resolver**

El mercado de autos usados es altamente competitivo y genera enormes volúmenes de datos. En este contexto, concesionarios, plataformas de compraventa y empresas de renting necesitan herramientas que les permitan evaluar de forma rápida y precisa si un vehículo representa una buena oportunidad de inversión.

Sin embargo, el proceso de análisis tradicional es manual, lento y propenso a errores subjetivos, lo que puede conducir a decisiones de inversión poco rentables o riesgosas.

Desde el punto de vista computacional, procesar grandes cantidades de información de forma secuencial resulta ineficiente. Por ello, se plantea resolver este problema aplicando técnicas de programación concurrente en Go, lo que permitirá acelerar la evaluación de vehículos a gran escala, aprovechando múltiples núcleos de procesamiento y mejorando los tiempos de análisis.

**Utilidad Práctica**

La solución propuesta automatiza la evaluación del riesgo de inversión de vehículos, siendo de gran utilidad para:

* Plataformas de compraventa de autos online.
* Concesionarios que buscan maximizar la rentabilidad de sus inventarios.
* Empresas de logística o renting que deben tomar decisiones informadas sobre la compra de flotas de vehículos.

Al clasificar de forma automática y eficiente el nivel de riesgo de inversión, estas organizaciones podrán:

* Tomar decisiones financieras más acertadas.
* Reducir el riesgo de adquirir vehículos con baja rentabilidad.
* Mejorar la planificación de inventarios y las estrategias de reventa.

**Decisión Automatizada**

La aplicación clasifica cada vehículo en uno de los siguientes niveles de riesgo de inversión:

* **Bajo Riesgo:** Alta probabilidad de rentabilidad en la reventa o uso.
* **Moderado Riesgo:** Rentabilidad incierta, sujeta a factores adicionales.
* **Alto Riesgo:** Alta probabilidad de pérdida económica si se adquiere.

Esta clasificación se realiza mediante un Árbol de Decisión implementado en Go, utilizando reglas de negocio derivadas de fuentes reconocidas como Kelley Blue Book (KBB), Edmunds y Manheim Market Report (MMR), y optimizadas para su ejecución en entornos concurrentes.

# Explicación del dataset Seleccionado

**Nombre del Dataset:**

Synthetic Car Sales Dataset (Over Million Records)

**Fuente:**

Disponible públicamente en la plataforma Kaggle:  
<https://www.kaggle.com/datasets/jayavarman/synthetic-car-sales-dataset-over-million-records>

**Descripción General:**

Este dataset fue generado de forma sintética y contiene más de un millón de registros, cumpliendo con el requisito mínimo de la rúbrica. Simula información transaccional de ventas de autos a nivel mundial, con datos realistas diseñados para el análisis de datos, modelado predictivo y pruebas de algoritmos de Machine Learning.

La estructura rica y diversa del dataset lo convierte en una base ideal para implementar algoritmos de clasificación y estudiar el comportamiento del mercado automotriz en distintos contextos.

**Justificación de la Elección:**

Se eligió este dataset debido a:

* Su gran volumen de datos, ideal para pruebas de algoritmos concurrentes.
* La diversidad de variables, que permiten aplicar un algoritmo de Árbol de Decisión.
* Su naturaleza sintética, evitando problemas legales o éticos relacionados con la privacidad de datos.

Además, al no ser un dataset previamente clasificado en términos de riesgo de inversión, permitió realizar un proceso completo de definición de reglas de negocio y etiquetado de datos en base a fuentes oficiales, lo cual enriquece el valor académico y técnico del trabajo.

**Variables Utilizadas:**

|  |  |  |
| --- | --- | --- |
| Columna | Tipo de Dato | Descripción |
| Brand | String | Marca del vehículo (Ej.: Toyota, Ford). |
| Model | Integer (Codificado) | Código del modelo del vehículo. |
| Year | Integer | Año de fabricación del vehículo. |
| Price | Float | Precio del vehículo en USD. |
| Mileage | Integer | Kilometraje total en millas. |
| Country | Integer (Codificado) | Código del país del comprador. |
| Condition | String | Estado del vehículo: “New”, “Used” o “Certified Pre-Owned”. |
| Riesgo\_Inversion | String | Nivel de riesgo de inversión: “Bajo”, “Moderado” o “Alto”. (Etiqueta creada). |

*Nota:* Las columnas relacionadas a datos personales (First Name, Last Name, Address) fueron descartadas por no aportar valor predictivo.

# Explicación del algoritmo y el uso de los mecanismos de sincronización utilizado.

**Algoritmo Implementado: Árbol de Decisión Basado en Reglas de Negocio**

El algoritmo utilizado en este proyecto es un Árbol de Decisión implementado manualmente en el lenguaje Go, sin utilizar librerías externas. Las reglas de decisión fueron definidas en base a las siguientes fuentes oficiales del mercado automotriz:

* **Kelley Blue Book (KBB)**
* **Edmunds**
* **Manheim Market Report (MMR)**
* **Datos del INEI sobre la depreciación de vehículos en Perú**

Estas reglas consideran las principales características del vehículo para determinar el nivel de riesgo de inversión. Los criterios se establecieron en base a combinaciones de:

* Año de fabricación
* Precio de compra
* Kilometraje acumulado
* Modelo y país de procedencia

Esto permite determinar de forma lógica si la inversión en un vehículo es de bajo, moderado o alto riesgo.

**Mecanismos de Sincronización Utilizados en Go**

Dado el volumen de datos procesados (más de un millón de registros), se aplicaron técnicas de programación concurrente para optimizar los tiempos de ejecución:

|  |  |
| --- | --- |
| Mecanismo | Descripción |
| Goroutines | Permiten ejecutar múltiples funciones de forma concurrente, mejorando el rendimiento al aprovechar los diferentes núcleos del procesador. |
| sync.WaitGroup | Se utilizó para asegurar que todos los procesos concurrentes finalizaran antes de continuar con la ejecución del programa. |
| sync.Mutex | Aplicado para evitar condiciones de carrera al actualizar variables compartidas, como los contadores de cada categoría de riesgo de inversión. |

**Implementación Concurrente vs. Secuencial**

|  |  |  |
| --- | --- | --- |
| Aspecto | Secuencial | Concurrente |
| Tiempo de Ejecución | Más alto | Reducido |
| Complejidad | Baja | Media (requiere sincronización) |
| Utilización de CPU | Baja | Alta (uso eficiente de múltiples núcleos) |

Se realizaron benchmarks de 1000 iteraciones calculando la **media recortada** para omitir valores atípicos, tal como lo establece la rúbrica. Los resultados demostraron una mejora significativa en los tiempos de ejecución al utilizar concurrencia.

# Presentar las pantallas del menú

Figura 1: Menú Principal de la Aplicación

Pantalla inicial de la aplicación desarrollada en Go, que permite al usuario seleccionar entre las funcionalidades implementadas: ingresar datos manualmente, cargar un archivo CSV, ejecutar el benchmark de rendimiento o salir de la aplicación.

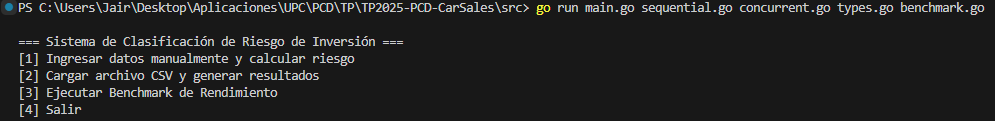


Figura 2: Ingreso Manual de Datos y Cálculo de Riesgo

Interfaz de ingreso manual de datos, donde el usuario proporciona las características del vehículo.  
El sistema muestra la clasificación del riesgo de inversión de forma inmediata utilizando las reglas del Árbol de Decisión implementado.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 3: Procesamiento de Archivo CSV

Captura del procesamiento de un archivo CSV con registros de vehículos.

El sistema genera un nuevo archivo con la clasificación de riesgo de inversión agregada, permitiendo la evaluación de grandes volúmenes de datos.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Figura 4: Resultados del Benchmark de Rendimiento

Pantalla que muestra los resultados de las pruebas de rendimiento, comparando los tiempos de ejecución y la precisión entre la versión secuencial y la concurrente.

Texto

El contenido generado por IA puede ser incorrecto.

# Explicación de las pruebas realizadas, resultados y pegar las imágenes de evidencia.

**Pruebas de Rendimiento**

Se realizaron pruebas de rendimiento con el objetivo de comparar la eficiencia entre la implementación secuencial y la concurrente del algoritmo de Árbol de Decisión.  
Se ejecutaron 1000 iteraciones por cada tipo de ejecución, procesando más de un millón de registros en cada iteración.

Para asegurar la calidad de los resultados, se utilizó la media recortada, eliminando los 50 valores más bajos y los 50 más altos en cada caso, tal como lo establece la rúbrica.

Resultados Obtenidos

|  |  |  |
| --- | --- | --- |
| Métrica | Secuencial | Concurrente |
| Tiempo Medio (ms) | 351.42 ms | 270.45 ms |
| Precisión del Modelo | 100.00% | 100.00% |
| Complejidad de Implementación | Baja | Media (Uso de sincronización) |
| Uso de CPU | Bajo | Alto (Uso de múltiples núcleos) |

*Nota:* Se demuestra que la implementación concurrente mejora los tiempos de ejecución sin afectar la precisión del modelo.

**Evidencias de las Pruebas Realizadas**

* **Figura 5: Resultados del Benchmark de Rendimiento**  
  Muestra los tiempos de ejecución promedio y la precisión del modelo tanto en la versión secuencial como en la concurrente.

Texto

El contenido generado por IA puede ser incorrecto.

* **Figura 6: Gráfico de Comparación de Tiempos de Ejecución**

**Gráfico, Gráfico de dispersión

El contenido generado por IA puede ser incorrecto.**

* **Figura 7: Gráfico de Comparación de Precisión del Modelo**

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

# Explicación de la Simulación Realizada con Promela y Evidencias

**Descripción de la Simulación**

Se implementó una simulación en el lenguaje de modelado Promela, utilizando la herramienta SPIN para la verificación formal de concurrencia.  
El objetivo de esta simulación fue modelar la exclusión mutua en un entorno concurrente, replicando de forma simplificada la lógica del sistema implementado en Go.

Se utilizó un semaforo binario (chan sem = [1] of { bit };) para controlar el acceso a la sección crítica, modelando el comportamiento del uso de sync.Mutex en Go.

**Modelo Implementado**

* Se modelaron **3 procesos concurrentes** simulando las operaciones concurrentes de evaluación de vehículos.
* Se utilizó la estructura de control assert(!critical) para garantizar la propiedad de exclusión mutua.
* La sección crítica fue protegida utilizando operaciones P (wait) y V (signal) sobre el semáforo.

**Código Implementado en Promela**

**Evidencias de la Simulación**

* **Figura 8: Simulación Ejecutándose en SPIN**  
  Muestra la ejecución correcta de la simulación, confirmando la correcta aplicación de exclusión mutua.

Texto

El contenido generado por IA puede ser incorrecto.

* **Figura 9: Resultados de la Verificación con SPIN**  
  Se presenta la salida de la herramienta SPIN, evidenciando que no se detectaron violaciones de assert ni interbloqueos (*errors: 0*).  
  Esto confirma que la propiedad de exclusión mutua se cumple correctamente.

Texto

El contenido generado por IA puede ser incorrecto.

# Explicación del análisis usando spin, pegar las imágenes de evidencia

* **Figura 10: Análisis de Exclusión Mutua con SPIN (Resultados de Verificación)**

En la imagen se presentan los resultados del análisis realizado con SPIN.  
Se puede observar que no se detectaron errores (errors: 0), lo que confirma que la propiedad de exclusión mutua se cumple correctamente en el modelo implementado.  
Asimismo, se verificó que no existen deadlocks ni estados inválidos, garantizando la correcta sincronización de los procesos concurrentes.

Texto

El contenido generado por IA puede ser incorrecto.

# Bibliografía

Ben-Ari, M. (2006). Principles of Concurrent and Distributed Programming (2nd ed.). Pearson Education.

Downey, A. (2016). The Little Book of Semaphores (2nd ed.). Retrieved from https://greenteapress.com/semaphores/

Holzmann, G. J. (2003). The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley.

Kelley Blue Book. (2025). Vehicle Valuation and Market Trends. Retrieved from https://www.kbb.com/

Edmunds. (2025). Car Pricing and Market Value Guide. Retrieved from https://www.edmunds.com/

Manheim Market Report (MMR). (2025). Used Car Market Trends. Retrieved from https://www.manheim.com/

Jayavarman, J. (n.d.). Synthetic Car Sales Dataset (Over Million Records). Retrieved from Kaggle: https://www.kaggle.com/datasets/jayavarman/synthetic-car-sales-dataset-over-million-records

Instituto Nacional de Estadística e Informática (INEI). (2025). Estadísticas de Comercio y Transporte de Vehículos en el Perú. Retrieved from <https://www.inei.gob.pe/>

# Anexo

**Enlace de GitHub del Código Fuente**

El código fuente completo de la aplicación, incluyendo la implementación en Go, los módulos de benchmarking, la simulación en Promela y las evidencias, se encuentra disponible en el siguiente repositorio:

<https://github.com/JairRodriguezCalla/TP2025-PCD-CarSales>

**b. Enlace del Video de Presentación**

[**https://drive.google.com/file/d/1z\_NX4KjNhyq9HE8pq02OeMBXL27-gbsl/view?usp=sharing**](https://drive.google.com/file/d/1z_NX4KjNhyq9HE8pq02OeMBXL27-gbsl/view?usp=sharing)