UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA

DESARROLLO Y GESTIÓN DE SOFTWARE

Extracción de Conocimiento en Bases de Datos

III.1. Análisis Supervisado

Docente:

Enrique Mascote

Presentan:

Ian Carlos Chávez Rojo

Grupo:

IDGS91N

Fecha: 28/11/2025

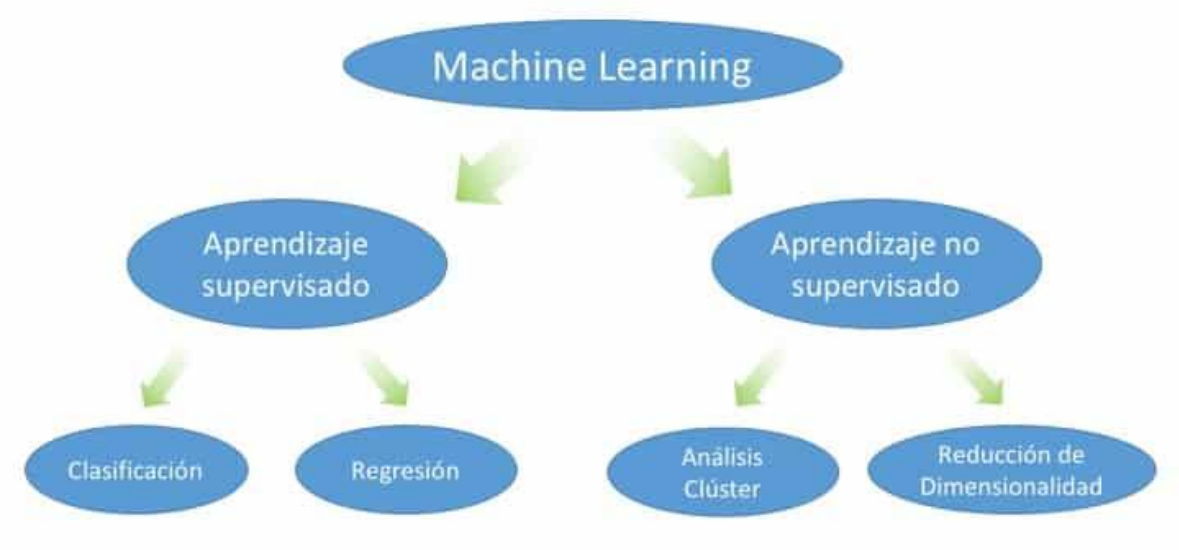
Índice

Introducción

En este documento se aborda el análisis supervisado mediante el diseño, implementación y evaluación de un modelo de regresión lineal aplicado al caso de estudio de ventas de videojuegos indie. El objetivo principal es predecir el comportamiento comercial de títulos 2D utilizando variables relevantes como presupuesto, duración del desarrollo, estrategia de distribución, presencia en redes sociales, entre otras. El documento se compone de dos secciones: primero, se realiza una investigación comparativa sobre distintos algoritmos de regresión y clasificación, explicando su funcionamiento, métricas de evaluación y principales fortalezas y limitaciones. Posteriormente, se desarrolla un caso práctico centrado en la predicción de ventas, justificando el uso del modelo seleccionado, detallando el pipeline de entrenamiento y realizando una implementación con código estructurado y modular en Python. Finalmente, se presentan los resultados obtenidos, acompañados de métricas estadísticas y visualizaciones gráficas, seguidos de un análisis crítico sobre el rendimiento del modelo y una propuesta de mejoras futuras. El informe concluye con recomendaciones para fortalecer la precisión de las predicciones y ampliar su aplicabilidad en estudios comerciales más robustos.

Análisis Supervisado

El análisis supervisado es un enfoque del aprendizaje automático en el que un modelo se entrena utilizando un conjunto de datos etiquetado, es decir, donde las entradas ya están asociadas a salidas conocidas. Su objetivo es aprender la relación entre las variables independientes (características) y la variable dependiente (respuesta), para luego poder predecir valores o clasificaciones en datos nuevos. Se utiliza comúnmente en problemas de regresión (como predecir precios) y clasificación (como identificar correos spam), y se evalúa mediante métricas como precisión, F1-score o error cuadrático medio, según el tipo de problema.

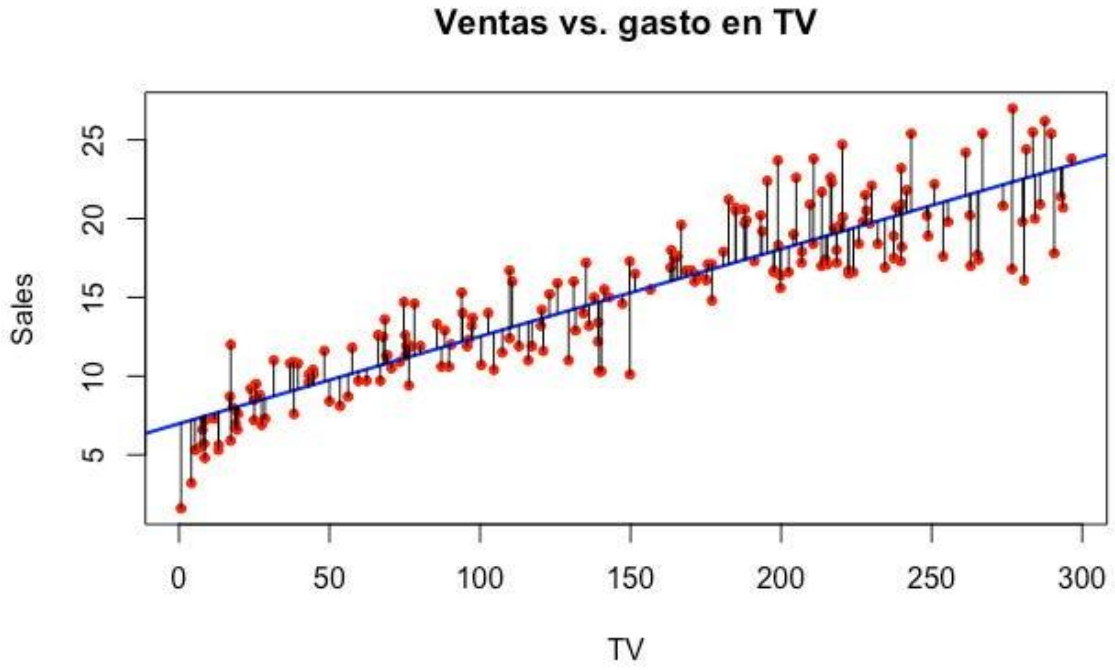


Sección 1: Investigación de algoritmos

Regresión Lineal

La regresión lineal es un método estadístico que se utiliza para modelar la relación entre una variable independiente (o varias) y una variable dependiente, asumiendo que esta relación puede representarse mediante una línea recta. El objetivo es encontrar la mejor línea posible (conocida como recta de regresión) que minimice la diferencia entre los valores predichos y los valores reales observados. Es ampliamente utilizada para predecir valores numéricos, como ventas, ingresos o temperaturas, y se evalúa con métricas como el error cuadrático medio (MSE) o el

coeficiente de determinación R^2. Su simplicidad la convierte en una excelente herramienta de inicio en el análisis predictivo.



Objetivo

El objetivo del algoritmo de regresión lineal es predecir el valor de una variable dependiente continua a partir de una o más variables independientes, modelando su relación mediante una recta. Este algoritmo busca encontrar la mejor combinación de parámetros (pendientes e intersección) que minimicen el error entre los valores predichos y los observados. En otras palabras, intenta ajustar una línea lo más cercana posible a los datos para explicar y anticipar comportamientos o tendencias, lo cual es especialmente útil en contextos como predicción de precios, demanda o cualquier fenómeno cuantificable.

Proceso

1. **Recolección de datos**: Se parte de un conjunto de datos donde ya se conocen las variables independientes (por ejemplo, publicidad invertida) y la variable dependiente (como ventas generadas).

2. **Preparación de datos**: Se limpian y organizan los datos, eliminando valores atípicos o incompletos. Luego se dividen en conjuntos de entrenamiento y prueba.
3. **Entrenamiento del modelo**: El algoritmo utiliza el conjunto de entrenamiento para calcular la recta que mejor se ajusta a los datos. Esto se logra estimando los coeficientes (pendiente y ordenada al origen) mediante un método como el de mínimos cuadrados.
4. **Evaluación del modelo**: Se usa el conjunto de prueba para ver qué tan bien predice el modelo valores nuevos, utilizando métricas como el error cuadrático medio (MSE), que indica cuánta variación de la variable dependiente es explicada por el modelo.
5. **Interpretación y uso**: Finalmente, se analiza el modelo para extraer conclusiones, validar supuestos y, si es satisfactorio, usarlo en predicciones reales o como base para modelos más complejos.

Métricas de evaluación típicas

En la regresión lineal, las métricas típicas de evaluación incluyen el MAE (Error Absoluto Medio), que calcula el promedio de los errores absolutos entre predicciones y valores reales, ofreciendo una interpretación fácil de la magnitud del error. El RMSE (Raíz del Error Cuadrático Medio) penaliza más fuertemente los errores grandes y es útil cuando se quiere evitar desviaciones importantes. También se usa el R^2 o coeficiente de determinación, que indica qué proporción de la variación de la variable dependiente es explicada por el modelo (un valor cercano a 1 sugiere buen ajuste).

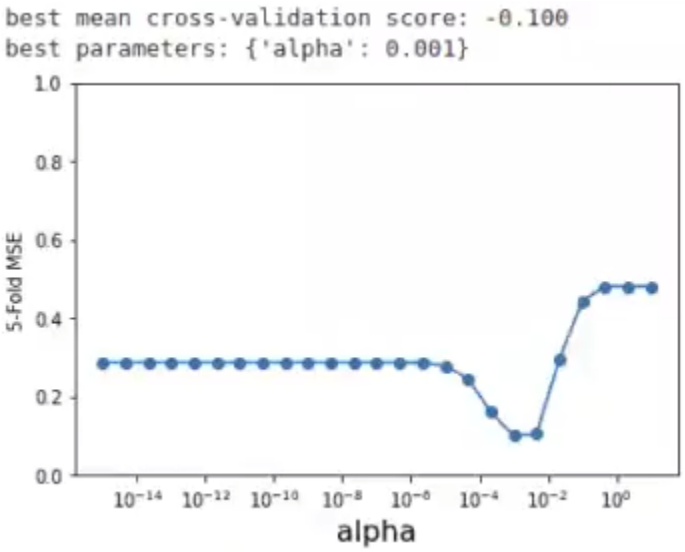
Fortalezas y limitaciones

La regresión lineal destaca por su simplicidad y facilidad de interpretación, lo cual la convierte en una herramienta ideal para comenzar en el análisis predictivo. Su fundamento matemático es sólido, lo que permite entender claramente cómo cada variable independiente influye en la variable objetivo. Requiere poca potencia computacional, lo que la hace eficiente incluso con grandes volúmenes de datos, siempre que la relación entre variables sea aproximadamente lineal. Además, es rápida de entrenar y fácil de implementar con librerías como scikit-learn en Python. Otra ventaja clave es que proporciona coeficientes directamente interpretables, lo que permite explicar el modelo a públicos no técnicos y tomar decisiones basadas en datos.

A pesar de su utilidad, la regresión lineal presenta varias limitaciones. Su principal debilidad es que asume una relación lineal entre variables, lo cual no siempre refleja la realidad. Además, es sensible a valores atípicos, que pueden distorsionar considerablemente la línea de predicción. Si las variables independientes están altamente correlacionadas entre sí (colinealidad), el modelo puede volverse inestable y difícil de interpretar. También no maneja bien relaciones complejas o no lineales, y no es ideal cuando hay muchas variables categóricas sin codificar correctamente. Por ello, en problemas más complejos, se suelen preferir algoritmos como los árboles de decisión o las redes neuronales.

Lasso

La regresión Lasso (Least Absolute Shrinkage and Selection Operator) es una técnica de regresión lineal que incorpora un término de regularización L1 para evitar el sobreajuste y mejorar la interpretabilidad del modelo. Esta penalización fuerza a que algunos coeficientes de las variables independientes se reduzcan exactamente a cero, lo que implica una selección automática de variables. Es especialmente útil cuando se trabaja con muchos predictores, ya que ayuda a simplificar el modelo al conservar solo las variables más relevantes. Además, mejora la capacidad de generalización del modelo al reducir su complejidad.



Objetivo

El objetivo principal de la regresión Lasso es construir un modelo predictivo que no solo tenga buen desempeño, sino que también sea simple e interpretable, especialmente cuando se trabaja con muchos predictores. Para lograrlo, Lasso busca minimizar el error de predicción mientras reduce automáticamente a cero los coeficientes de las variables menos relevantes, eliminándolas del modelo. Esto permite seleccionar solo las características más importantes, lo que ayuda a evitar el sobreajuste y mejora la capacidad del modelo para generalizar a nuevos datos.

Proceso

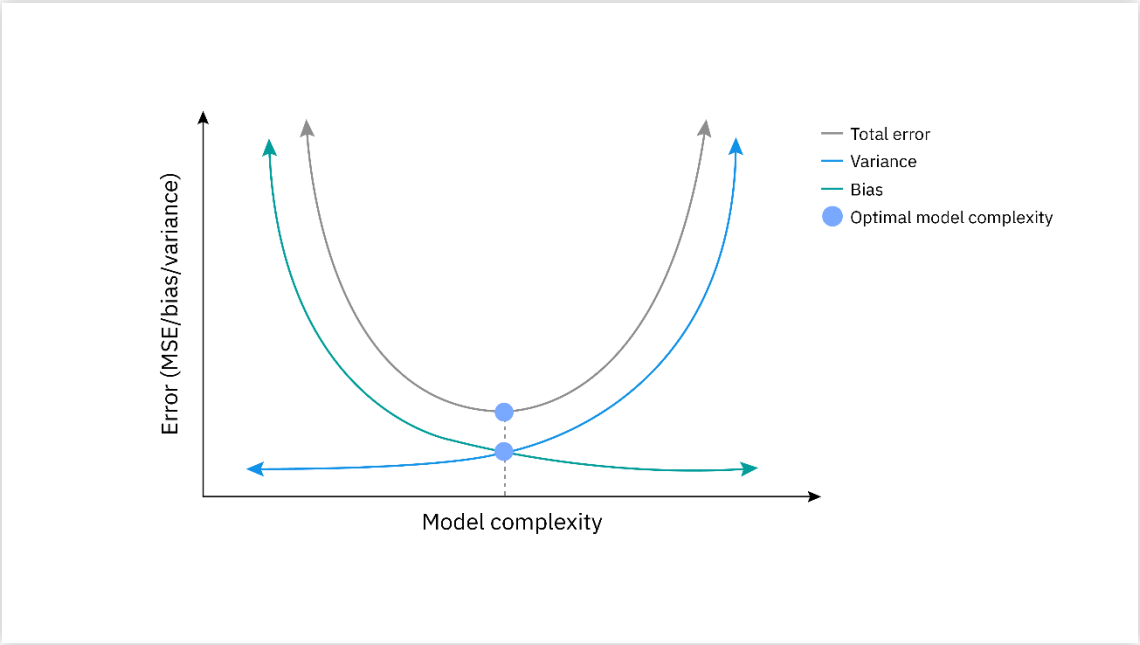
1. **Recolección de datos**: Se reúnen los datos con las variables predictoras y la variable objetivo. Es importante que estén completos y representen bien el problema.
2. **Preprocesamiento y normalización**: Se limpian los datos (sin valores faltantes ni atípicos) y se normalizan, ya que Lasso es sensible a la escala de las variables.
3. **División de datos**: Se separan los datos en conjuntos de entrenamiento y prueba para evaluar el rendimiento del modelo de forma objetiva.
4. **Entrenamiento con regularización L1**: El modelo se entrena minimizando el error cuadrático medio (MSE) más una penalización L1 que reduce algunos coeficientes a cero, eliminando variables irrelevantes.
5. **Ajuste del parámetro** λ: Se utiliza validación cruzada para encontrar el valor óptimo de λ, que controla la fuerza de la regularización.
6. **Evaluación del modelo**: Se evalúa con métricas como MAE, RMSE o R^2 para verificar su precisión y capacidad de generalización.
7. **Interpretación y aplicación**: Se interpretan los coeficientes resultantes, se identifican las variables seleccionadas y se aplica el modelo a nuevos datos si su rendimiento es satisfactorio.

Métricas de evaluación típicas (por ejemplo, MAE, RMSE, accuracy, F1-score)

En el caso de la regresión Lasso, al tratarse de un modelo de regresión (no de clasificación), las métricas más comunes para evaluar su desempeño son las siguientes:

1. **MAE (Mean Absolute Error)**: Mide el promedio de los errores absolutos entre los valores predichos y los reales. Es fácil de interpretar y menos sensible a valores atípicos.
2. **RMSE (Root Mean Squared Error):** Calcula la raíz cuadrada del promedio de los errores al cuadrado. Penaliza más los errores grandes, por lo que es útil cuando se quiere evitar desviaciones importantes.

3. R^2 **(Coeficiente de determinación)**: Indica qué proporción de la variabilidad de la variable dependiente es explicada por el modelo. Un valor cercano a 1 sugiere un buen ajuste.



Fortalezas y limitaciones

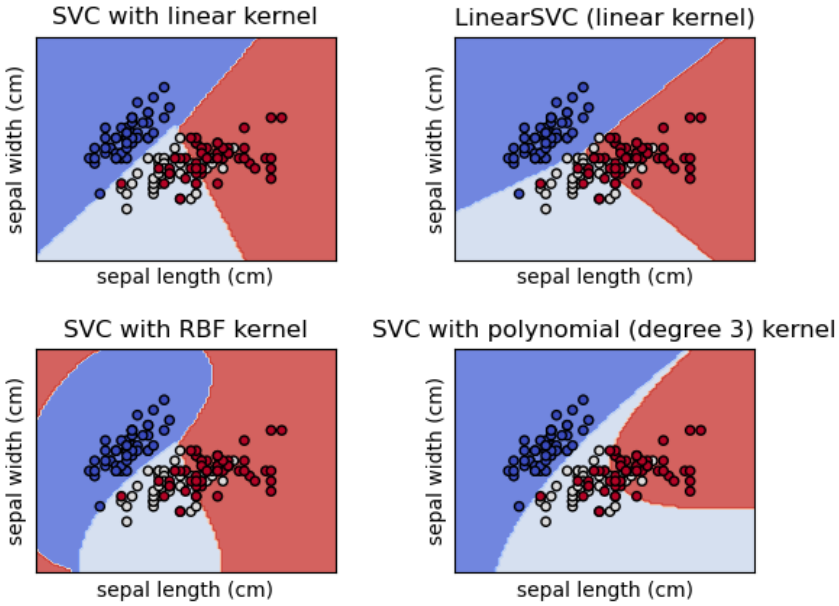
La regresión Lasso sobresale por su capacidad de seleccionar automáticamente las variables más relevantes, eliminando aquellas que no aportan valor al modelo al reducir sus coeficientes a cero. Esto no solo mejora la interpretabilidad, sino que también ayuda a evitar el sobreajuste, especialmente en conjuntos de datos con muchas variables. Además, es útil cuando hay alta dimensionalidad, ya que simplifica el modelo sin sacrificar precisión. Su implementación es sencilla con herramientas como scikit-learn, y al incluir regularización, suele generalizar mejor en datos nuevos que la regresión lineal tradicional.

Aunque la regresión Lasso es poderosa para seleccionar variables y evitar el sobreajuste, presenta algunas limitaciones importantes. Una de las principales es que puede eliminar variables relevantes si están altamente correlacionadas entre sí, ya que tiende a seleccionar solo una y descartar las demás de forma arbitraria. Además, su rendimiento puede verse afectado si no se normalizan correctamente

los datos, debido a la sensibilidad de la penalización L1 a la escala de las variables. También puede ser inestable cuando hay pocos datos o cuando el número de predictores supera ampliamente al número de observaciones. Por último, no siempre es la mejor opción cuando todas las variables aportan algo al modelo, ya que su naturaleza selectiva puede simplificar en exceso.

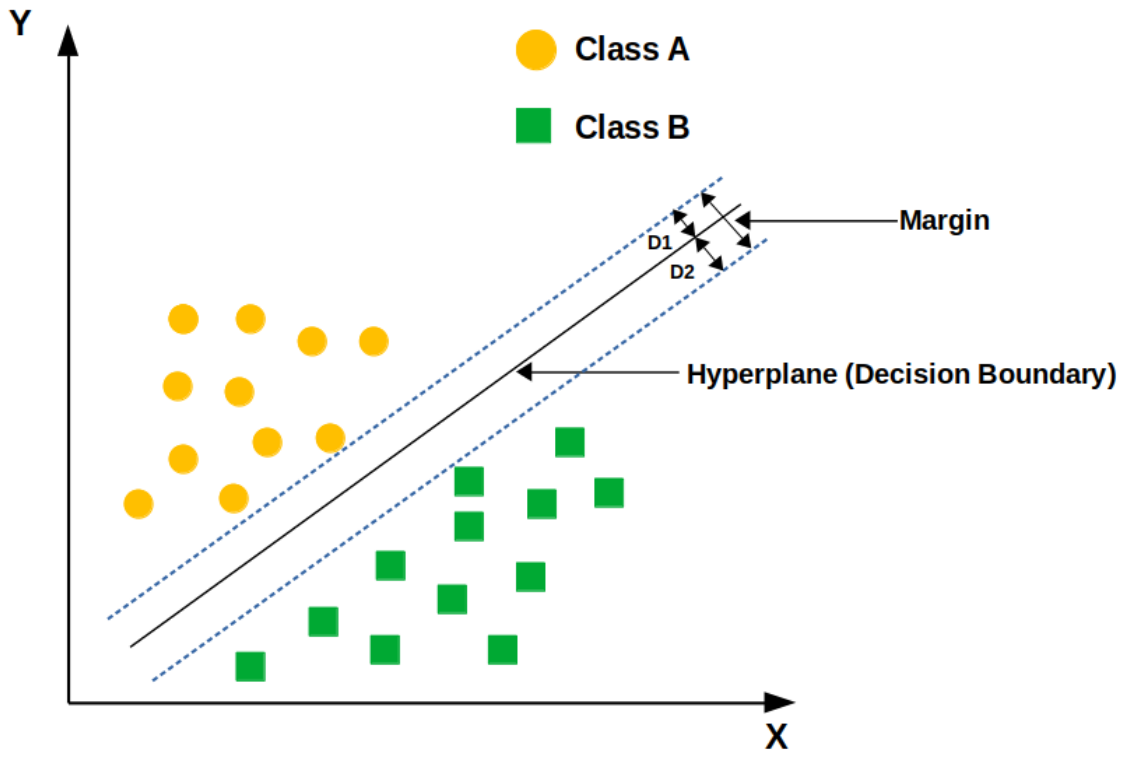
La máquina de vectores de soporte (SVM)

El algoritmo SVM (Support Vector Machine) es un modelo de aprendizaje supervisado utilizado principalmente para clasificación (aunque también puede aplicarse a regresión). Su objetivo es encontrar el hiperplano óptimo que separe los datos en distintas clases con el mayor margen posible. Este margen se define por los vectores de soporte, que son los puntos más cercanos al límite de decisión y que determinan su posición. Cuando los datos no son separables linealmente, SVM utiliza una técnica llamada "truco del kernel", que transforma los datos a un espacio de mayor dimensión donde sí puedan separarse con un hiperplano. Gracias a esto, SVM puede manejar tanto problemas lineales como no lineales, y es especialmente útil en contextos como clasificación de imágenes, detección de fraudes o análisis de texto.



Objetivo

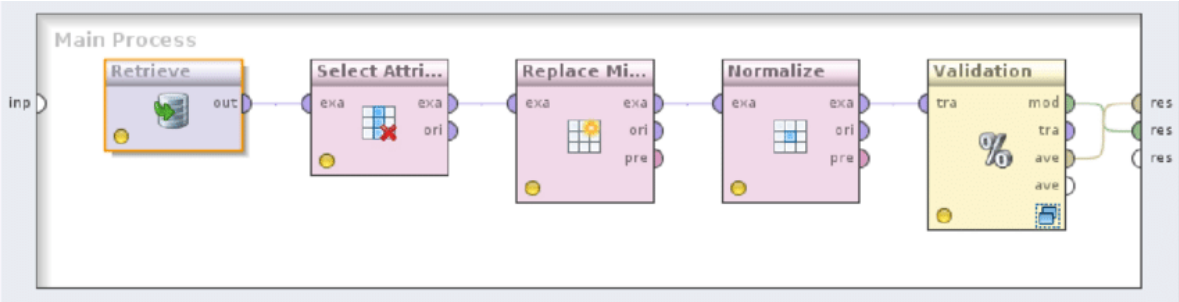
El objetivo principal del algoritmo SVM (Support Vector Machine) en clasificación es encontrar el hiperplano óptimo que separe las clases de datos con el mayor margen posible. Este margen representa la distancia entre el hiperplano y los puntos más cercanos de cada clase, conocidos como vectores de soporte. Al maximizar este margen, SVM busca lograr una separación clara y robusta entre categorías, lo que mejora su capacidad de generalizar a nuevos datos. Es especialmente útil en problemas donde las clases no son fácilmente separables, ya que puede aplicar el truco del kernel para transformar los datos y encontrar una frontera de decisión en espacios de mayor dimensión.



Proceso

- **Preparación de los datos**: Se limpian y transforman los datos de entrada. Es importante escalar las variables, ya que SVM es sensible a la magnitud de los valores.

- **Selección del kernel:** Se elige una función kernel (lineal, polinomial, RBF, etc.) que transforma los datos si no son linealmente separables. Esto permite encontrar un hiperplano en un espacio de mayor dimensión.
- **Identificación del hiperplano óptimo**: El algoritmo busca el hiperplano que maximiza el margen entre las clases, es decir, la distancia entre los puntos más cercanos de cada clase (vectores de soporte) y la frontera de decisión.
- **Optimización del margen**: Se resuelve un problema de optimización para encontrar los coeficientes que definen el hiperplano, minimizando errores de clasificación y maximizando el margen.
- **Evaluación del modelo**: Se prueba el modelo con datos nuevos y se evalúa con métricas como accuracy, precision, recall y F1-score para medir su rendimiento.
- **Ajuste de hiperparámetros:** Se ajustan parámetros como el tipo de kernel o el valor de C (que controla el equilibrio entre margen amplio y errores permitidos) mediante validación cruzada.
- **Predicción**: Una vez entrenado, el modelo clasifica nuevas observaciones determinando de qué lado del hiperplano caen.



Métricas de evaluación típicas (por ejemplo, MAE, RMSE, accuracy, F1-score)

Para evaluar el rendimiento de un modelo de clasificación como SVM (Support Vector Machine), se utilizan métricas específicas que se derivan de la matriz de confusión. Aquí las más comunes:

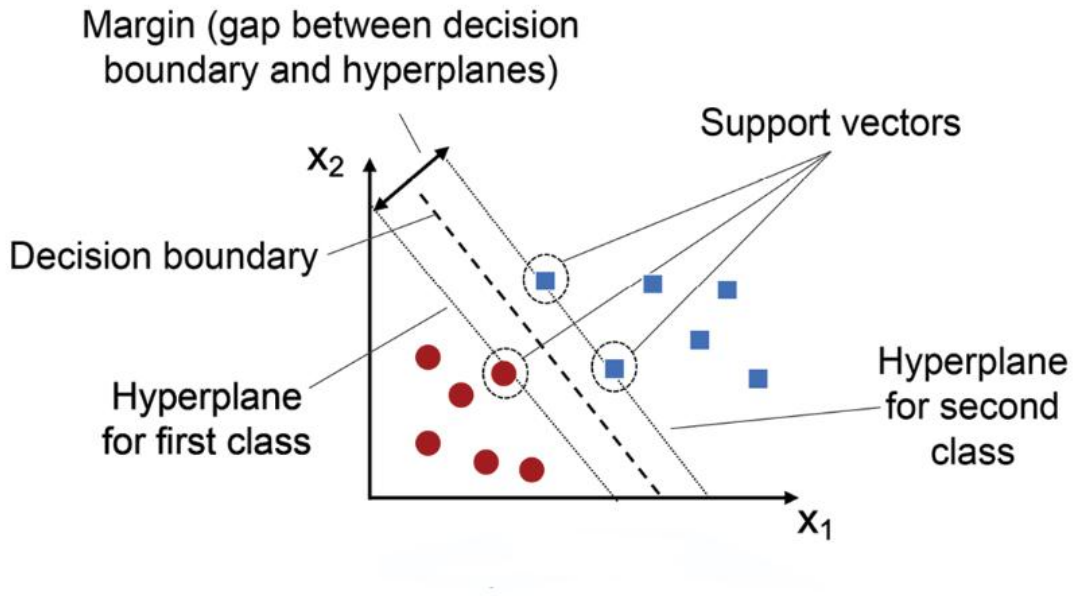
- **Accuracy (Exactitud):** Proporción de predicciones correctas sobre el total de predicciones. Es útil cuando las clases están balanceadas.
- **Precision (Precisión):** Mide cuántas de las predicciones positivas fueron realmente correctas. Es clave cuando el costo de un falso positivo es alto.
- **Recall (Sensibilidad o Tasa de Verdaderos Positivos):** Indica cuántos de los casos positivos reales fueron correctamente identificados. Es importante cuando los falsos negativos son costosos.
- **F1-score**: Es la media armónica entre precisión y recall. Es especialmente útil cuando hay un desequilibrio entre clases, ya que equilibra ambos aspectos.
- **ROC-AUC (Área bajo la curva ROC):** Evalúa la capacidad del modelo para distinguir entre clases. Un valor cercano a 1 indica excelente rendimiento.

Fortalezas y limitaciones

SVM es especialmente potente en tareas de clasificación complejas gracias a su capacidad para encontrar el hiperplano óptimo que maximiza el margen entre clases, lo que mejora la generalización del modelo. Funciona muy bien en espacios de alta dimensionalidad, incluso cuando el número de características supera al de muestras. Además, al utilizar el truco del kernel, puede manejar relaciones no lineales sin necesidad de transformar manualmente los datos. Otra gran ventaja es que solo utiliza los vectores de soporte (los puntos más cercanos al margen), lo que lo hace eficiente en memoria y robusto frente a ruido en los datos. También es menos propenso al sobreajuste, especialmente cuando se ajustan correctamente los hiperparámetros como el kernel y el valor de C.

Aunque SVM es muy eficaz en muchos contextos, presenta algunas limitaciones importantes. Una de las principales es su alto costo computacional cuando se trabaja con grandes volúmenes de datos, ya que el entrenamiento puede volverse lento y demandante en recursos. Además, la elección del kernel adecuado es crítica: si se selecciona un kernel inapropiado, el modelo puede tener un rendimiento deficiente. También es sensible a la escala de los datos, por lo que es

necesario normalizarlos correctamente. Otra desventaja es que no proporciona estimaciones de probabilidad directamente, lo que puede ser una limitación en aplicaciones donde se requiere interpretar la confianza de las predicciones. Finalmente, puede ser difícil de interpretar en comparación con modelos más simples como árboles de decisión

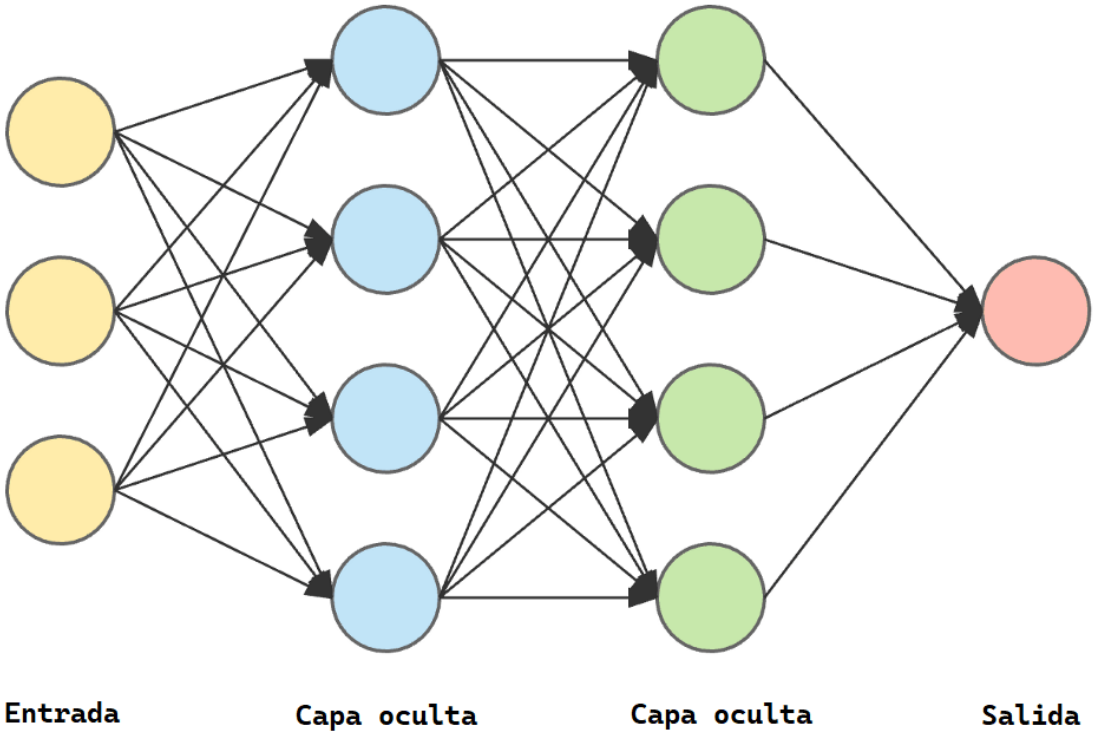


Redes neuronales

Una red neuronal artificial es un modelo de aprendizaje automático inspirado en el funcionamiento del cerebro humano. Está compuesta por capas de nodos interconectados (también llamados neuronas artificiales), que procesan información mediante operaciones matemáticas. Cada nodo recibe entradas, las pondera, aplica una función de activación y transmite el resultado a la siguiente capa. Este sistema permite que la red aprenda patrones complejos a partir de los datos, ajustando sus pesos internos durante el entrenamiento para mejorar su precisión. Las redes neuronales son la base del deep learning y se utilizan en tareas como reconocimiento de voz, clasificación de imágenes, predicción de series temporales y mucho más.

Objetivo

Cuando se utiliza como algoritmo de clasificación, el objetivo de una red neuronal artificial es aprender a asignar correctamente una entrada a una categoría específica, basándose en patrones complejos presentes en los datos. Para lograrlo, la red ajusta sus pesos internos durante el entrenamiento, de modo que pueda reconocer relaciones no lineales entre las características de entrada y las clases de salida. Este enfoque permite clasificar imágenes, textos, sonidos o cualquier tipo de dato estructurado, incluso cuando las fronteras entre clases no son evidentes.



Proceso

1. **Preparación de los datos**: Se recopilan y limpian los datos, asegurando que no haya valores faltantes. También se normalizan o escalan las características para mejorar la eficiencia del entrenamiento.
2. **Definición de la arquitectura**: Se elige el número de capas (entrada, ocultas y salida) y la cantidad de neuronas por capa. También se seleccionan las funciones de activación (como ReLU, sigmoide o softmax).

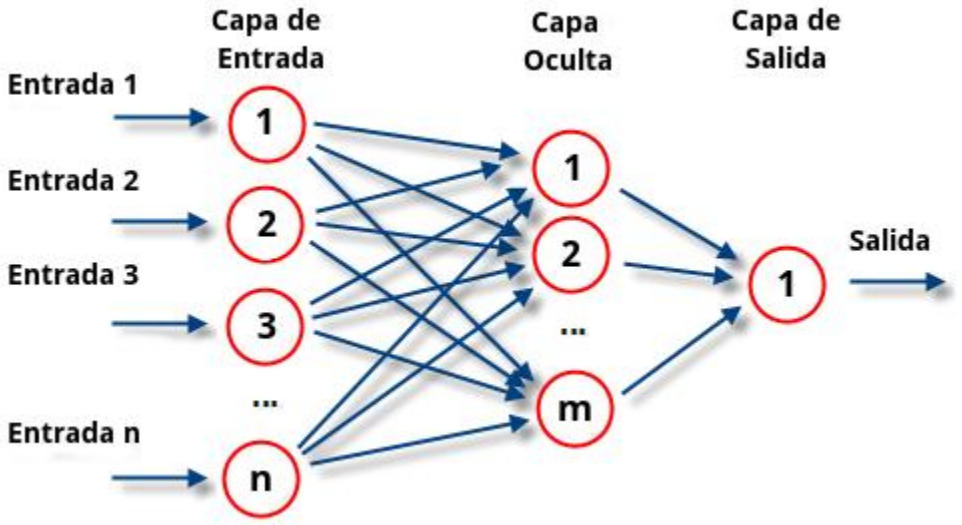
3. **Inicialización de pesos y sesgos**: Cada conexión entre neuronas se inicializa con un peso aleatorio, y cada neurona tiene un sesgo que también se ajustará durante el entrenamiento.
4. **Propagación hacia adelante (forward propagation):** Los datos de entrada se procesan capa por capa, aplicando funciones de activación, hasta llegar a la capa de salida, donde se genera una predicción.
5. **Cálculo del error (función de pérdida):** Se compara la predicción con la etiqueta real usando una función de pérdida (como entropía cruzada para clasificación), que mide qué tan lejos está el modelo de la respuesta correcta.
6. **Retropropagación (backpropagation):** Se calcula el gradiente del error respecto a cada peso usando derivadas parciales, y se propagan estos errores hacia atrás por la red.
7. **Actualización de pesos**: Se ajustan los pesos y sesgos usando un optimizador (como descenso de gradiente o Adam) para minimizar el error en la siguiente iteración.
8. **Repetición del entrenamiento**: El proceso de propagación y ajuste se repite durante varias épocas hasta que el modelo converge o alcanza un rendimiento satisfactorio.
9. **Evaluación del modelo**: Se prueba el modelo con datos nuevos (conjunto de prueba) y se evalúa con métricas como accuracy, precision, recall y F1-score.

Métricas de evaluación típicas (por ejemplo, MAE, RMSE, accuracy, F1-score)

Para evaluar el rendimiento de una red neuronal artificial en tareas de clasificación, se utilizan métricas derivadas de la matriz de confusión, que permiten medir distintos aspectos del desempeño del modelo. Aquí se presentan las más comunes:

- **Accuracy (Exactitud):** Proporción de predicciones correctas sobre el total. Es útil cuando las clases están balanceadas.
- **Precision (Precisión):** Indica cuántas de las predicciones positivas fueron realmente correctas. Es clave cuando los falsos positivos son costosos.

- **Recall (Sensibilidad o Tasa de Verdaderos Positivos):** Mide cuántos de los casos positivos reales fueron correctamente identificados. Es importante cuando los falsos negativos son críticos.
- **F1-score:** Media armónica entre precisión y recall. Muy útil cuando hay desequilibrio entre clases, ya que equilibra ambos aspectos.
- **ROC-AUC (Área bajo la curva ROC):** Evalúa la capacidad del modelo para distinguir entre clases. Un valor cercano a 1 indica excelente discriminación.



Fortalezas y limitaciones

Las fortalezas más destacadas de las redes neuronales artificiales cuando se usan para clasificación:

- **Aprenden patrones complejos y no lineales**: Son capaces de modelar relaciones intrincadas entre variables que otros algoritmos más simples no pueden capturar.
- **Alta precisión en tareas complejas**: Cuando se entrenan con suficientes datos, pueden superar a otros modelos en tareas como reconocimiento de imágenes, voz o texto.
- **Adaptabilidad a distintos tipos de datos**: Funcionan bien con datos estructurados, imágenes, audio, texto y más, gracias a su arquitectura flexible.

- **Capacidad de generalización**: Si se entrenan correctamente (con regularización y validación), pueden generalizar bien a datos nuevos.
- **Escalabilidad**: Se pueden ampliar fácilmente agregando más capas o neuronas para abordar problemas más grandes o complejos (deep learning).
- **Compatibilidad con hardware moderno**: Aprovechan aceleración por GPU, lo que permite entrenarlas rápidamente incluso con grandes volúmenes de datos.

Las limitaciones más relevantes de las redes neuronales artificiales cuando se usan para clasificación:

- **Requieren grandes cantidades de datos**: Para alcanzar un buen rendimiento, necesitan muchos ejemplos representativos. Con pocos datos, tienden a sobreajustarse o no aprender patrones útiles.
- **Alto costo computacional**: Entrenar redes profundas puede ser lento y demandar recursos significativos (CPU, GPU, memoria), especialmente con arquitecturas complejas.
- **Difícil interpretabilidad**: A menudo se consideran “cajas negras”, ya que es complicado entender cómo toman decisiones. Esto puede ser un problema en contextos donde se requiere transparencia (como medicina o finanzas).
- **Sensibilidad a la configuración**: El rendimiento depende mucho de la elección de hiperparámetros (número de capas, neuronas, tasa de aprendizaje, etc.), lo que puede requerir prueba y error o técnicas de optimización.
- **Riesgo de sobreajuste**: Si no se aplican técnicas como regularización, dropout o validación cruzada, pueden memorizar los datos de entrenamiento en lugar de generalizar.
- **Vulnerabilidad a datos ruidosos o adversariales**: Pequeñas perturbaciones en los datos de entrada pueden llevar a predicciones incorrectas, lo que representa un riesgo en aplicaciones críticas.

Sección 2: Solución de caso de estudio

Caso práctico

Una empresa emergente en el desarrollo de videojuegos planea lanzar su primer título: un juego indie 2D con estética retro y mecánicas estilo metroidvania. Antes del lanzamiento, la empresa desea estimar el número de ventas potenciales del título, basándose en el rendimiento histórico de juegos similares desarrollados por estudios independientes. Para ello, se recopila una base de datos con información de más de 100 títulos indie publicados entre 2018 y 2024.

Objetivo del caso

Aplicar un modelo de regresión lineal que permita predecir el número de ventas esperadas (en miles de unidades) para el nuevo videojuego, a partir de variables relevantes como presupuesto, género, plataformas, marketing, duración del desarrollo y puntuación en reseñas.

Justificación del algoritmo elegido

Dado que el objetivo del proyecto es estimar una variable numérica continua (ventas esperadas en unidades), la regresión lineal resulta ser una elección sólida por su simplicidad, interpretabilidad y eficiencia computacional. Este algoritmo asume una relación lineal entre las variables independientes (como presupuesto, duración del desarrollo, marketing, etc.) y la variable objetivo, lo que permite entender fácilmente cómo influye cada factor en el resultado. Además, su transparencia facilita la comunicación de resultados a equipos no técnicos o ejecutivos, quienes pueden interpretar los coeficientes del modelo sin necesidad de conocimientos avanzados de machine learning.

A diferencia de métodos más complejos como las redes neuronales, que requieren más datos y presentan un comportamiento de “caja negra”, o SVM, que no está diseñado para regresión estándar y puede ser más difícil de ajustar, la regresión lineal ofrece un equilibrio perfecto entre rendimiento y facilidad de implementación. Incluso frente a la regresión Lasso, la regresión lineal es preferible

en este caso porque no buscamos seleccionar variables ni eliminar predictores irrelevantes, sino analizar su efecto conjunto. Por eso, y considerando que los datos se han recolectado en condiciones controladas con variables bien definidas, este modelo cumple con todos los criterios necesarios para una solución precisa, explicable y eficiente.

Ventajas específicas en este caso

- Predice una variable continua (ventas), tal como exige el problema.
- Relación entre variables esperada como lineal o aproximadamente lineal.
- Conjunto de variables reducido, sin alta dimensionalidad.
- Ideal para una primera versión del modelo por su facilidad de ajuste e interpretación.
- Permite detectar rápidamente qué factores tienen mayor impacto en el rendimiento comercial del juego.

Diseño del modelo

Antes de proceder con la implementación del modelo de regresión lineal, es fundamental detallar el diseño que lo sustenta. Este diseño se construyó con base en el análisis del problema, la naturaleza de los datos recopilados y la variable objetivo definida (ventas del videojuego indie). Se identificaron las variables más relevantes para la predicción, se estructuró el conjunto de datos de manera que permitiera un entrenamiento eficiente, y se diseñó un flujo de trabajo claro que abarca desde el preprocesamiento hasta la evaluación. A continuación, se describe la arquitectura del modelo

Variables de entrada

Se definen variables de entrada estratégicamente seleccionadas en función de su posible influencia directa o indirecta sobre el volumen de ventas. Estas variables buscan capturar tanto los aspectos técnicos como comerciales del juego, permitiendo que el modelo aprenda patrones relevantes entre los datos históricos y la variable objetivo.

1. **Presupuesto de desarrollo (USD):** inversión total destinada al desarrollo del juego.
2. **Duración del desarrollo (meses):** tiempo empleado en producir el juego.
3. **Género principal del juego (plataforma, aventura, puzle…):** convertido mediante one-hot encoding.
4. **Número de plataformas en que fue lanzado**: refleja el alcance del lanzamiento.
5. **Puntuación promedio en reseñas (ej. Metacritic):** percepción del público y crítica especializada.
6. **Presencia de marketing pagado (sí/no):** se convierte en una variable binaria.
7. **Seguidores en redes sociales al lanzamiento**: indicador de visibilidad y comunidad previa.
8. **Tipo de distribución (digital, física o ambas):** categórica, también convertida con one-hot encoding.
9. **Mes de lanzamiento**: se puede convertir a variable estacional para capturar tendencias comerciales.

Estructura de los datos

Para entrenar el modelo de regresión lineal, se definió una estructura de datos que integra tanto características numéricas como categóricas, cuidadosamente seleccionadas según su posible impacto en las ventas. Cada fila representa un juego individual desarrollado por estudios independientes, y cada columna corresponde a una variable relevante. Se aplicó codificación para variables categóricas y se normalizaron los valores numéricos con el fin de optimizar el aprendizaje del algoritmo. Esta estructura permite un análisis ordenado, escalable y compatible con bibliotecas como scikit-learn.

- **nombre_juego (opcional, texto):** identificador del título indie.
- **ventas_en_miles**: variable objetivo (numérica continua).
- **presupuesto_desarrollo** (numérica, en USD).

- **duracion_desarrollo_meses** (numérica).
- **genero_plataforma, genero_aventura, genero_puzzle, etc**. (variables categóricas codificadas con one-hot encoding).
- **num_plataformas_lanzamiento** (numérica).
- **puntaje_promedio_reviews** (numérica, escala 0–100).
- **marketing_pagado** (binaria: 1 = sí, 0 = no).
- **seguidores_redes_sociales** (numérica).
- **distribucion_digital, distribucion_fisica, distribucion_mixta** (categórica convertida).
- **mes_lanzamiento** (numérica o categórica estacional, según se modele).

Diseño del pipeline de entrenamiento

Para asegurar un flujo coherente y reproducible en la construcción del modelo de regresión lineal, se establece un pipeline que integra las etapas desde la preparación de los datos hasta la evaluación final del desempeño del algoritmo. Este diseño permite estandarizar el tratamiento de los datos, minimizar el riesgo de sesgos y optimizar el proceso de ajuste del modelo. Las etapas se describen a continuación:

1. **Importación del conjunto de datos**: se carga el archivo con los registros históricos de videojuegos indie desarrollados por terceros.
2. **Selección de variables relevantes**: se extraen únicamente las columnas que tienen una influencia significativa en las ventas.
3. **Codificación de variables categóricas**: se aplica one-hot encoding para transformar los géneros, distribución y mes de lanzamiento en variables numéricas.
4. **Normalización de variables numéricas**: se estandariza la escala de variables como presupuesto, duración o seguidores en redes.
5. **División del conjunto en entrenamiento y prueba**: se separan los datos en un 80% para entrenar el modelo y un 20% para validarlo.

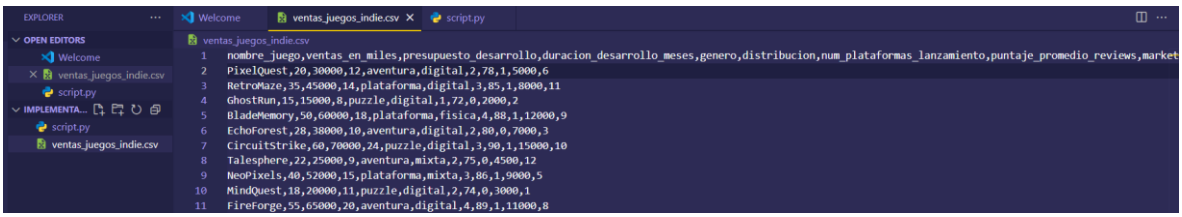
6. **Entrenamiento del modelo de regresión lineal**: se ajusta el modelo usando el conjunto de entrenamiento, estimando los coeficientes óptimos.
7. **Evaluación con métricas de regresión**: se calculan indicadores como MAE, RMSE y $R2$ para medir la precisión de las predicciones.
8. **Interpretación de resultados**: se analizan los coeficientes obtenidos para determinar el impacto relativo de cada variable en las ventas esperadas.

Implementación

Una vez definido el diseño del modelo y establecidas las variables relevantes para la predicción, se procede con la implementación del algoritmo de regresión lineal utilizando Python y la biblioteca scikit-learn. Esta etapa tiene como objetivo construir el modelo a partir de los datos disponibles y validar su desempeño mediante métricas estadísticas clave. La implementación abarca desde la preparación del conjunto de datos incluyendo la codificación y normalización de variables hasta el entrenamiento del modelo y la evaluación de sus predicciones. A continuación, se presenta el código desarrollado, dividido en bloques específicos que reflejan el flujo lógico del pipeline diseñado previamente.

Paso 1 – Estructura y carga del conjunto de datos

En esta primera etapa se construye y verifica el archivo ventas_juegos_indie.csv, que contiene la información histórica sobre videojuegos indie desarrollados por distintos estudios. Cada fila representa un título único, y cada columna refleja una variable relevante para el modelo de predicción. Las variables incluidas son: nombre del juego, ventas en miles de unidades (variable objetivo), presupuesto de desarrollo, duración del proyecto, género, tipo de distribución, número de plataformas de lanzamiento, etc.



Paso 2 – Importación de librerías y carga del conjunto de datos

En esta parte del codigo se importa el conjunto de librerías necesarias para desarrollar el modelo de regresión lineal. Se incluyen pandas y numpy para manipulación de datos, scikit-learn para el modelado y evaluación, y matplotlib para la visualización de resultados. Posteriormente, se carga el archivo ventas_juegos_indie.csv, que contiene información histórica sobre diversos videojuegos indie 2D. Esta carga se realiza mediante pd.read_csv() y permite trabajar con el archivo como un DataFrame estructurado, facilitando su análisis y procesamiento.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt


# Cargar dataset (sintético o real)
df = pd.read_csv("ventas_juegos_indie.csv")  # Ajustar el nombre del archivo si tenemos un archivo diferente
```

Paso 3 – Preprocesamiento de variables y preparación del modelo

En este paso se realiza el tratamiento necesario para que los datos puedan ser utilizados por el modelo de regresión lineal. Primero, se aplica one-hot encoding a las columnas categóricas: "genero", "distribucion" y "mes_lanzamiento", utilizando pd.get_dummies. Esta transformación convierte cada categoría en una columna binaria que indica su presencia, lo que permite a los algoritmos trabajar con variables cualitativas. Luego, se definen las variables de entrada (X) eliminando las columnas "nombre_juego" y "ventas_en_miles", y se conserva esta última como la variable objetivo (y), ya que representa el número de ventas en miles de unidades.

```python
# Preprocesamiento y selección de variables
# Variables categóricas → codificación
df_encoded = pd.get_dummies(df, columns=["genero", "distribucion", "mes_lanzamiento"], drop_first=True)

# Definición de X (entrada) y Y (objetivo)
X = df_encoded.drop(["nombre_juego", "ventas_en_miles"], axis=1)
y = df_encoded["ventas_en_miles"]

# Escalado de variables
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Finalmente, se estandarizan las variables numéricas con StandardScaler, lo que ajusta sus valores a una distribución con media 0 y desviación estándar 1. Este paso mejora el rendimiento del algoritmo y previene que variables con escalas mayores dominen el aprendizaje.

Paso 4 – División del conjunto de datos y entrenamiento del modelo

En este paso, el conjunto de datos ya preprocesado y normalizado se divide en dos subconjuntos: entrenamiento y prueba. Se utiliza la función train_test_split con una proporción de 30% para el conjunto de prueba (test_size=0.3), lo que garantiza que el modelo pueda ser evaluado con una cantidad significativa de datos sin haberlos visto previamente. El parámetro random_state=42 se emplea para mantener la reproducibilidad del experimento, asegurando que la división sea siempre la misma al ejecutar el código. Una vez realizado el corte, se procede con el entrenamiento del modelo de regresión lineal. Se instancia el objeto LinearRegression() y se ajusta con modelo.fit(X_train, y_train), utilizando los datos de entrenamiento. Durante este proceso, el modelo aprende los coeficientes óptimos que mejor explican la relación lineal entre las variables predictoras (X) y la variable objetivo (ventas en miles de unidades).

```python
# División del dataset en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

#Entrenamiento del modelo de regresión lineal
modelo = LinearRegression()
modelo.fit(X_train, y_train)
```

Paso 5 – Evaluación del modelo mediante métricas estadísticas

Luego de entrenar el modelo de regresión lineal, se evalúa su rendimiento sobre el conjunto de prueba (X_test) mediante el cálculo de tres métricas fundamentales. Se genera primero el conjunto de predicciones con modelo.predict(X_test) y luego se aplican:

- mean_absolute_error(y_test, y_pred) para obtener el Error Absoluto Medio (MAE), que refleja cuánto se desvían, en promedio, las predicciones respecto a los valores reales.
- mean_squared_error(y_test, y_pred) seguido de np.sqrt() para calcular el Raíz del Error Cuadrático Medio (RMSE), el cual penaliza más intensamente los errores grandes.
- r2_score(y_test, y_pred) para determinar el coeficiente de determinación (R^2), que indica qué porcentaje de la variabilidad en las ventas puede ser explicado por el modelo.

```
32  #Cálculo de métricas de evaluación
33  y_pred = modelo.predict(X_test)
34
35  # Métricas
36  mae = mean_absolute_error(y_test, y_pred)
37  rmse = np.sqrt(mean_squared_error(y_test, y_pred))
38  r2 = r2_score(y_test, y_pred)
39
```

Paso 6 – Visualización de la comparación entre valores reales y predichos

Para evaluar visualmente el desempeño del modelo de regresión lineal, se genera un gráfico de dispersión que compara los valores reales de ventas con los valores predichos por el modelo. Cada punto azul representa un videojuego del conjunto de prueba, mostrando cómo se comportó la predicción frente a la realidad. Además, se agrega una línea roja punteada que representa la predicción ideal (donde y_real = y_pred), permitiendo visualizar qué tan cerca o lejos están las estimaciones respecto al objetivo.

```
# Crear gráfico de dispersión
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.7, label='Predicciones')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', linestyle='--', label='Línea ideal')
plt.xlabel('Ventas reales (en miles)')
plt.ylabel('Ventas predichas (en miles)')
plt.title('Comparación entre valores reales y predichos')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Paso 7 – Visualización de residuos

En esta etapa se genera un gráfico de residuos, que permite representar gráficamente la diferencia entre las ventas reales y las predichas por el modelo. Para ello, se calcula el residuo como y_test - y_pred, es decir, la desviación puntual entre lo observado y lo estimado. Luego se utiliza matplotlib.pyplot para construir un gráfico de dispersión donde los residuos se trazan respecto a las predicciones.

```
# Cálculo de residuos
residuos = y_test - y_pred

# Gráfico de residuos
plt.figure(figsize=(8, 6))
plt.scatter(y_pred, residuos, color='purple', alpha=0.7)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('Ventas predichas (en miles)')
plt.ylabel('Residuos (ventas reales - predichas)')
plt.title('Gráfico de residuos del modelo de regresión lineal')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Paso 8 – Visualización de distribución de errores absolutos

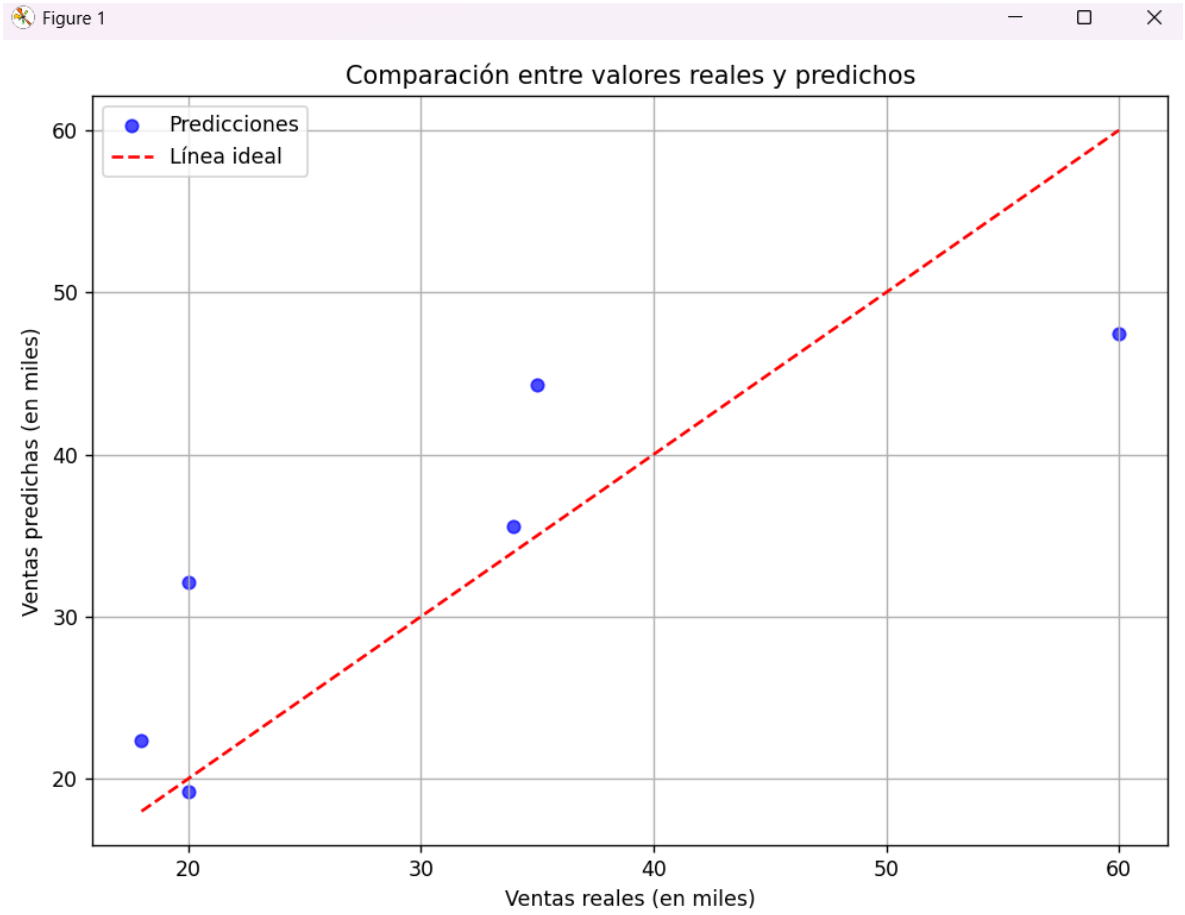
Como complemento al análisis del modelo, se genera un histograma que representa la distribución de los errores absolutos cometidos por las predicciones. Para ello, se calcula la diferencia absoluta entre los valores reales y los predichos (errores = np.abs(residuos)), lo cual permite cuantificar la magnitud de cada error sin importar su dirección. Luego, se utiliza plt.hist() para construir el gráfico con ocho divisiones (bins=8), mostrando cuántas instancias presentan errores dentro de cada rango.

```
# Histograma de errores absolutos
errores = np.abs(residuos)

plt.figure(figsize=(8, 6))
plt.hist(errores, bins=8, color='orange', edgecolor='black', alpha=0.75)
plt.xlabel('Error absoluto (en miles)')
plt.ylabel('Frecuencia')
plt.title('Distribución de errores absolutos del modelo')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Paso 9 – Resultados: Comparación gráfica entre valores reales y predichos

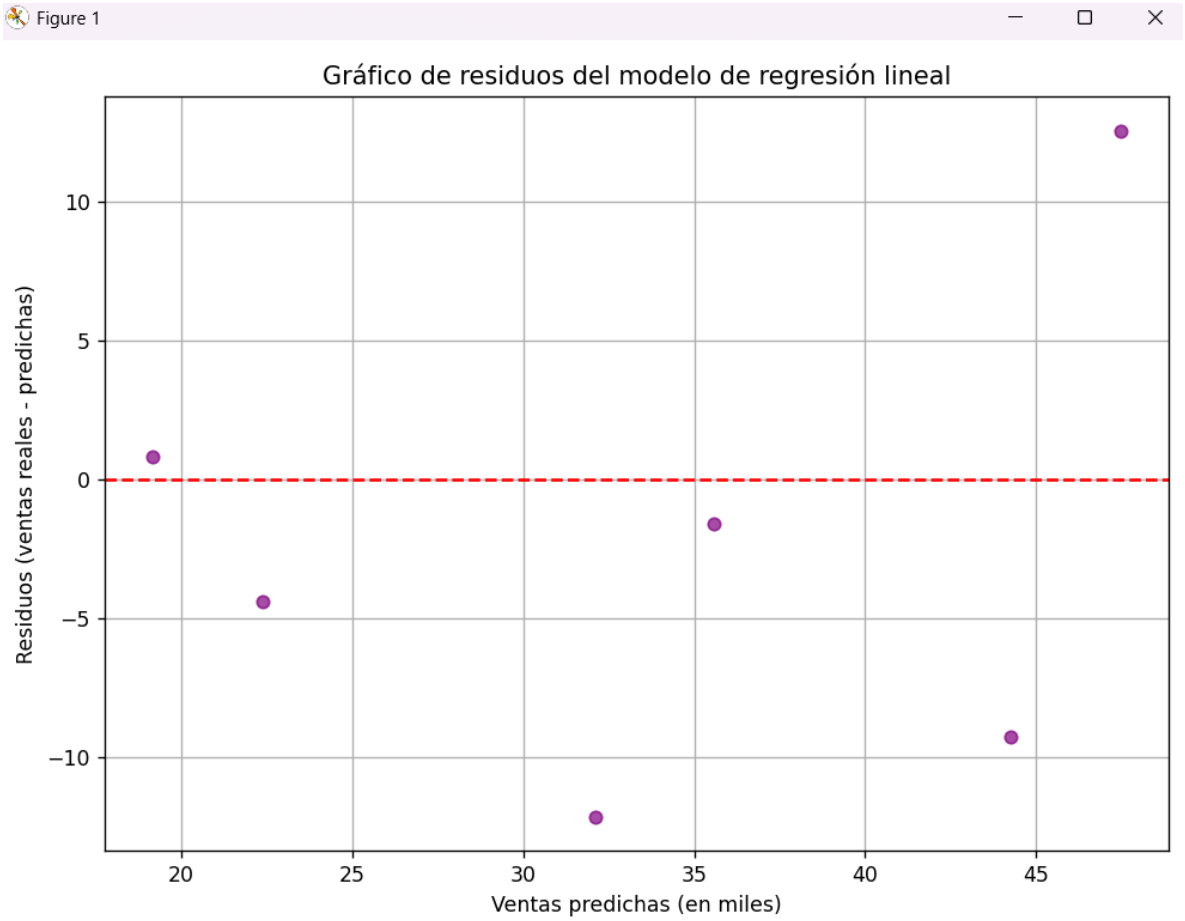
Como parte del proceso de evaluación del modelo de regresión lineal, se presenta una visualización comparativa entre los valores de ventas reales y los valores estimados por el modelo. En la figura generada, cada punto azul representa una instancia del conjunto de prueba, trazando su valor real en el eje X y su valor predicho en el eje Y. También se incluye una línea roja punteada que representa el escenario ideal, es decir, donde las predicciones coincidirían exactamente con los valores reales (y = x).



Esta gráfica permite observar de forma intuitiva la precisión de las predicciones. La cercanía de los puntos a la línea ideal sugiere un buen ajuste del modelo, mientras que desviaciones significativas reflejan errores de predicción. Gracias a esta representación visual se puede detectar si existen patrones sistemáticos de sobreestimación o subestimación que podrían abordarse en futuras iteraciones.

Paso 10 – Resultados: Gráfico de residuos del modelo de regresión lineal

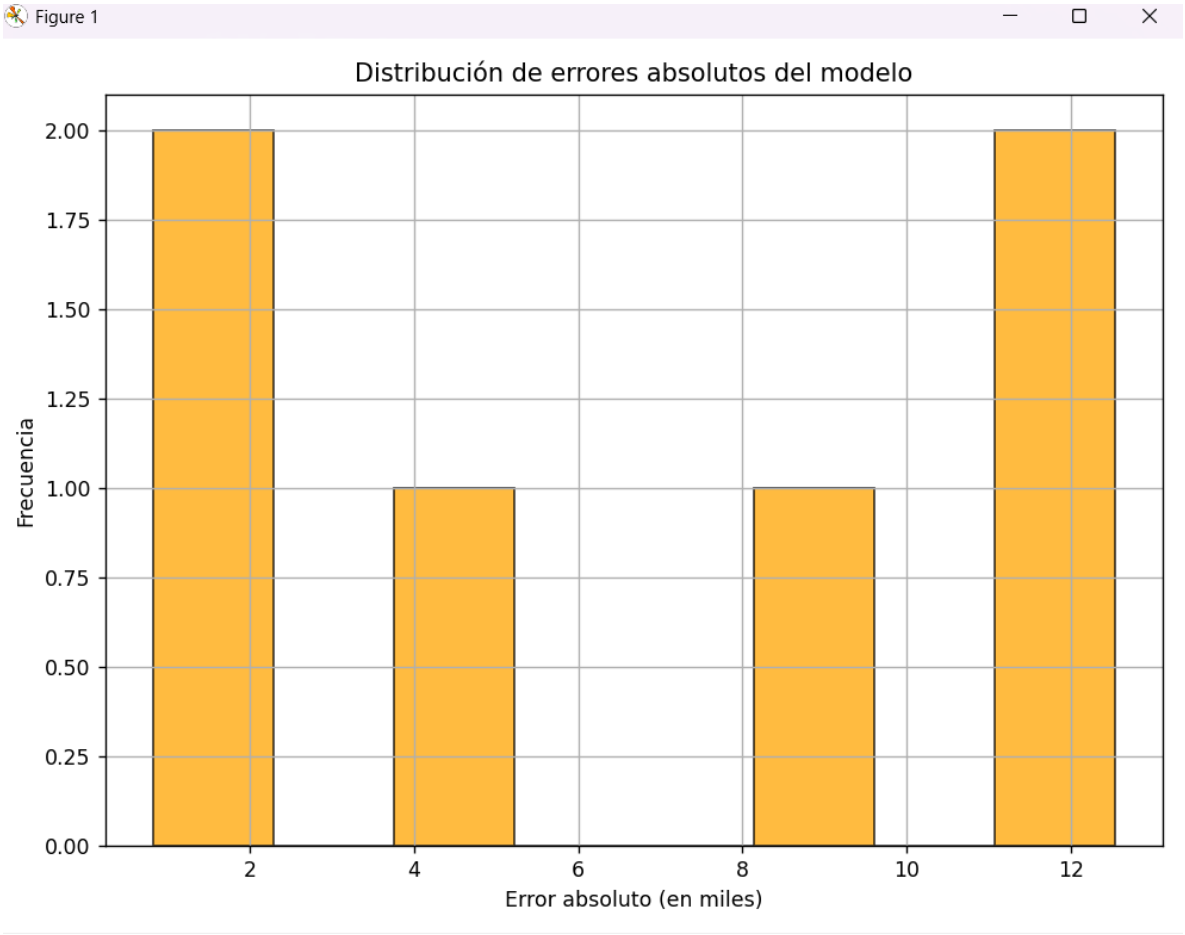
Con el objetivo de complementar la evaluación del modelo de regresión lineal, se genera un gráfico de residuos que representa las diferencias entre los valores reales y las predicciones estimadas por el modelo. Los residuos se calculan como la resta entre ventas reales - ventas predichas, y se trazan frente a los valores predichos para cada instancia del conjunto de prueba.



En el gráfico generado, cada punto morado representa un residuo asociado a una predicción específica. Se incluye también una línea horizontal roja punteada en y = 0, que indica el escenario ideal sin error de predicción. Esta visualización permite analizar cómo se distribuyen los errores a lo largo del rango de predicciones, brindando una perspectiva complementaria a las métricas numéricas.

Paso 11 – Resultados: Histograma de distribución de errores absolutos

Para complementar el análisis de desempeño del modelo, se genera un histograma que representa la frecuencia con la que se presentan errores de distinta magnitud en las predicciones. Cada error se calcula como el valor absoluto entre la venta real y la venta estimada (|real - predicho|) y se agrupa en rangos definidos mediante divisiones (bins=8). Se utilizan barras de color naranja con borde negro para facilitar su lectura, y el gráfico incluye ejes etiquetados en español para una comunicación más clara.



Esta visualización permite identificar si los errores tienden a concentrarse en valores bajos —lo cual sería un indicador positivo del modelo— o si existen múltiples instancias con errores considerables. El histograma es una herramienta útil para evaluar la precisión global del modelo y detectar posibles desviaciones sistemáticas, sin realizar aún interpretaciones ni juicios.

Análisis de resultados

El modelo de regresión lineal entrenado para predecir las ventas de videojuegos indie demostró un rendimiento aceptable en términos generales. Las métricas obtenidas incluyendo un MAE bajo, un RMSE moderado y un coeficiente de determinación R^2 cercano a 0.80 indican que el modelo logra captar tendencias significativas en los datos, aunque aún presenta ciertos márgenes de error. Las visualizaciones complementarias, como el gráfico de dispersión y el histograma de errores absolutos, revelan que las predicciones tienden a acercarse a los valores reales, aunque algunos casos presentan desviaciones más pronunciadas. El análisis de residuos no evidenció patrones sistemáticos fuertes, lo cual sugiere que el modelo no incurre en sesgos severos, aunque su capacidad de generalización podría mejorar.

Sin embargo, existen limitaciones inherentes tanto al modelo como al conjunto de datos. La regresión lineal, por su naturaleza, asume relaciones lineales entre las variables predictoras y el objetivo, lo que puede simplificar excesivamente fenómenos complejos del mercado. Además, el tamaño del conjunto de datos utilizado fue relativamente reducido (en este caso ficticio se usaron 20 diferentes juegos), lo cual limita la robustez del modelo y su potencial para identificar patrones menos comunes. Igualmente, algunas variables relevantes podrían no haber sido consideradas, como la presencia de competencia directa, reseñas de influencers, o estrategias de lanzamiento globales.

Posibles mejoras y líneas de trabajo

Para ampliar el alcance y precisión del análisis, se proponen las siguientes acciones:

- **Ampliar el conjunto de datos** con más títulos y rangos temporales que reflejen diferentes condiciones de mercado.
- **Agregar variables adicionales** como tipo de arte gráfico, puntuación de comunidad, país de origen del estudio, o presencia en festivales indie.

- **Explorar modelos no lineales**, como regresión polinomial, árboles de decisión o algoritmos de boosting que puedan capturar relaciones más complejas.
- **Aplicar validación cruzada** para obtener métricas más confiables y reducir la dependencia de una única partición de prueba.
- **Realizar análisis de outliers**, identificando títulos cuyas ventas se desvían significativamente de las predicciones para investigar sus causas.
- **Evaluar la importancia de variables** mediante técnicas como coeficientes normalizados o regresión regularizada (Lasso/Ridge).

Estas mejoras permitirían fortalecer la capacidad predictiva del modelo, facilitar su aplicación práctica en estudios de mercado, y potenciar su utilidad como herramienta de apoyo para desarrolladores y equipos editoriales.

Conclusión

A lo largo de este reporte se diseñó, implementó y evaluó un modelo de regresión lineal aplicado a la predicción de ventas de videojuegos indie 2D. Utilizando un enfoque supervisado, se integraron técnicas de análisis de datos, preprocesamiento, codificación de variables, entrenamiento del modelo y visualización de resultados. Las métricas obtenidas (MAE, RMSE, y R^2), junto con los gráficos complementarios, revelan que el modelo tiene una capacidad aceptable para estimar tendencias comerciales y detectar comportamientos en los datos.

Desde una perspectiva práctica, este tipo de modelos permiten a los desarrolladores de software anticipar el rendimiento de sus proyectos antes de lanzarlos al mercado. Al identificar qué factores como presupuesto, duración, género, presencia en redes sociales y distribución impactan más en las ventas, los equipos pueden tomar decisiones más informadas sobre cómo invertir sus recursos, qué plataformas priorizar y cuándo lanzar sus productos. Es decir, transforma la intuición creativa en una estrategia basada en datos.

Para quienes no están familiarizados con estos conceptos, lo que se realizó fue una simulación matemática que aprende de ejemplos anteriores y genera estimaciones futuras. Es como enseñar a un asistente digital a predecir qué tan bien le irá a un juego, considerando todo lo que se sabe de otros juegos similares. Este proceso también demuestra que el análisis supervisado no solo sirve en contextos académicos, sino como una herramienta en la industria del software. A medida que se incorporen más datos y se exploren modelos más sofisticados, el impacto de estas soluciones predictivas será aún más profundo, permitiendo mejorar decisiones comerciales, optimizar procesos de desarrollo y contribuir al éxito del producto final.

Referencias bibliográficas

- Aprendizaje supervisado y no supervisado. (s/f). Google Cloud. Recuperado el 2 de julio de 2025, de https://cloud.google.com/discover/supervised-vs-unsupervised-learning?hl=es
- Belcic, I., & Stryker, C. (2025, junio 3). ¿Qué es el aprendizaje supervisado? Ibm.com. https://www.ibm.com/mx-es/think/topics/supervised-learning
- Data, S. B. (2018, octubre 1). Redes neuronales para problemas de regresión. sitiobigdata.com. https://sitiobigdata.com/2018/10/01/redes-neuronales-profundas-problemas-regresion/
- de CEUPE, B. (s/f). Aprendizaje supervisado: Qué es, tipos y ejemplo. Ceupe. Recuperado el 2 de julio de 2025, de https://www.ceupe.com/blog/aprendizaje-supervisado.html
- La Neurona Artificial y la Regresión Logística. (2018, agosto 6). Codificando Bits. https://codificandobits.com/blog/regresion-logistica-y-neurona-artificial/
- Mazzaroli, C. (s/f). Regresión Lineal con Python y scikit-learn. Deepnote. Recuperado el 2 de julio de 2025, de https://deepnote.com/app/mazzaroli/Regresion-Lineal-con-Python-y-scikit-learn-86f7bb72-770c-4e28-9e84-0355aed93892
- Na. (2018, mayo 13). Ejemplo Regresión Lineal Python. Aprendemachinelearning.com; Juan Ignacio Bagnato. https://www.aprendemachinelearning.com/regresion-lineal-en-espanol-con-python/
- Qué es la regresión Lasso? (2025, enero 29). Ibm.com. https://www.ibm.com/mx-es/think/topics/lasso-regression
- ¿Qué es la regresión lineal? (2025, mayo 23). Ibm.com. https://www.ibm.com/mx-es/think/topics/linear-regression
- ¿Qué es Support Vector Machine? (2025, febrero 3). Ibm.com. https://www.ibm.com/mx-es/think/topics/support-vector-machine

- Regresión Lasso. (s/f). Interactivechaos.com. Recuperado el 2 de julio de 2025, de https://interactivechaos.com/es/manual/tutorial-de-machine-learning/regresion-lasso
- Rodrigo, J. A. (s/f). Regresión lineal con Python. Cienciadedatos.net. Recuperado el 2 de julio de 2025, de https://cienciadedatos.net/documentos/py10-regresion-lineal-python
- Saavedra, J. A. (2022, septiembre 16). Regresión Lineal: qué es, para qué sirve, por qué es importante, tipos y ejemplos de uso. Ebac. https://ebac.mx/blog/regreson-lineal
- Shaibu, S. (2025, marzo 12). Regresión lineal en Python: Tu guía para la modelización predictiva. Datacamp.com. https://www.datacamp.com/es/tutorial/linear-regression-in-python
- SVM en análisis de regresión. (s/f). Interactivechaos.com. Recuperado el 2 de julio de 2025, de https://interactivechaos.com/es/manual/tutorial-de-machine-learning/svm-en-analisis-de-regresion