Logotipo

Descripción generada automáticamenteDibujo en blanco y negro

Descripción generada automáticamente con confianza baja**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

Aprendizaje de Máquina

**Práctica 4: “Regresión”**

Hernández Martínez Ernesto Ulises

Grupo 5BM1

5° Semestre

Periodo: agosto 2025 – enero 2026

Profesor:

Abdiel Reyes Vera

Índice de Contenido

[Introducción. 1](#_Toc207582276)

[Desarrollo. 2](#_Toc207582277)

[Escritura y lectura de datos. 2](#_Toc207582278)

[Arbol de decisión binario. 2](#_Toc207582279)

[Lista de nodos. 3](#_Toc207582280)

[Ejemplo en ejecución. 4](#_Toc207582281)

[Conclusión. 6](#_Toc207582282)

[Referencias. 7](#_Toc207582283)

**Índice de Tablas y Figuras**

[Tabla 1Ejemplo de categorización del águila, el ajolote y la ardilla 2](#_Toc207582509)

[Tabla 2 Comparación entre pregunta y número de items en el ejemplo 5](#_Toc207582510)

[Figura 1 Primera pregunta de ejemplo 4](#_Toc207582539)

[Figura 2 Segunda pregunta de ejemplo 4](#_Toc207582540)

[Figura 3 Tercera pregunta de ejemplo 4](#_Toc207582541)

[Figura 4 Cuarta pregunta de ejemplo 4](#_Toc207582542)

[Figura 5 Quinta pregunta de ejemplo 5](#_Toc207582543)

[Figura 6 Mensaje de victoria de la computadora 5](#_Toc207582544)

# Introducción.

La presente práctica de la materia de aprendizaje de máquina aborda el juego de “20 preguntas”, el cual consiste en adivinar en lo que piense el usuario en veinte preguntas o menos, teniendo en cuenta que el usuario solo podrá responder sí o no (respuestas binarias). Esto se logró mediante la abstracción del problema empleando un árbol de decisión binario, para posteriormente implementarlo codificando en Python.

Python nos ofrece simpleza para desarrollar nuestras ideas claramente, siendo más sencillo de leer y de comprender para futuras actualizaciones, revisiones o trabajos.

El razonamiento detrás de la resolución del problema fue transformar el problema inicial en un problema más sencillo, mucho más parecido a un búsqueda binaria. Para esto, el conjunto de datos o dataset se compuso de una serie de cosas/objetos/items que fueron clasificados en varias columnas (que respondían a la preguntas) y etiquetados de manera binaria, con 1 si pertenecian al grupo (respondían a la pregunta) y 0 en el caso contrario.

El siguiente paso, una vez cargado el dataset a utilizar fue la creación de una lista de nodos, donde cada nodo era una representación de cada columna en el dataset. El nodo se compone de id, nombre (pregunta), cantidad de 1’s, cantidad de 0’s, total (1’s + 0’s), probabilidad de 1 (#1’s / total), probabilidad de 0 (#0’s / total) y la entropía calculada con los datos anteriores. Es importante destacar que la lista de nodos se irá actualizando en cada recursión, buscando mantener los datos actualizados para el correcto funcionamiento del programa.

Luego viene la función principal para realizar el cuestionario al usuario, se trata de una función recursiva, que irá formando un árbol de decisión binario, donde cada pregunta o nodo es calculada al instante (en cada recursión) con el objetivo de dividir a los datos de la manera más equitativa posible, para esto se usa una función auxiliar, donde una vez actualizada la lista de nodos se busca aquel con mayor entropía. Una vez seleccionada la pregunta, se le presenta al usuario, con la opción de responder SÍ o NO y dependiendo de su decisión se mantendrán los datos que cumplan con la condición, eliminando de nuestra búsqueda a los restantes, esperando a haber eliminado a la mitad en cada recursión.

La recursión se dentendrá cuando solo quede 1 item en el conjunto de datos (haber encontrado una posible respuesta), cuando haya varios items y todos cumplan las mismas condiciones (se probará luego uno a uno) o cuando no haya items en el conjunto de datos (haber fallado en encontrar una respuesta). En el peor de los casos el item no es encontrado, pero es posible agregarlo al dataset respondiendo a una serie de preguntas sobre este.

# Desarrollo.

## Escritura y lectura de datos.

Para la escritura de los datos se buscó conseguir una gran cantidad de sustantivos, que pudieran ser utilizados de manera común en el día a día. Se pensó en hablar acerca de animales, cosas, lugares, personas, incluso sentimientos, emociones o representaciones de conceptos no físicos.

Posteriormente estos sustantivos fueron categorizados de acuerdo a preguntas, si respondían a estas preguntas se etiquetaban con 1, de lo contrario con 0, estas preguntas representan categorías a las que cada sustantivo podría o no pertenecer, dejando entonces a cada item perteneciente a una o más categorías.

Para los casos de prueba esto no era tan complejo, pues al ser pocos no eran necesarias tantas categorías e inlcuso podían ser clasificadas manualmente, sin embargo para la versión final era necesario incluir tantas como fueran posibles, lo que complicó la categorización, esto nos orilló a usar herramientas para poder terminar este trabajo.

Véase a continuación un ejemplo de los items en el dataset de prueba, se puede apreciar que cada cabezal de columna lleva el nombre de la categoría y dentro de la columna están los números 0 o 1, que indican el estado de pertenencia.

Tabla 1Ejemplo de categorización del águila, el ajolote y la ardilla



Para el dataset de prueba se trataron items unicamente pertenecientes a la categoría de animales (sin humanos), lo que nos facilitó la inspección del funcionamiento del programa. Finalmente, el dataset es guardado como un archivo con extensión “.csv”.

Para la lectura de los datos se hizo uso de la librería “pandas” de Python, que nos apoya a abrir el archivo y obtener la información del dataset con el que se quiera trabajar.

## Arbol de decisión binario.

Es común en las prácticas de “adivinación” el uso de árboles de decisión, donde los caminos se bifurcan, sin embargo, para el caso especial de esta práctica, se le hizo una pequeña adaptación al concepto del árbol. En vez de tener una pregunta esperando tras de otra, formadas, con la condición de acuerdo a donde se iba la rama, en este caso la pregunta (representada por el nodo en el árbol) es elegida al momento.

Para este proceso, desde el llamado a la función entrevistadora se entra en una recursión, donde en cada llamada nueva a la función (incluyendo a la raíz) se selecciona la pregunta que mejor divide a los datos en dos partes iguales. Nos apoyamos en el concepto de Entropía, una métrica que nos indica que tan diferentes son un conjunto de datos, mientras más se acerque el resultado a 1, mayor entropía y mejor canditado para ser la siguiente pregunta en el árbol.

Para el cálculo de la entropía nos basamos en la siguiente fórmula, donde:

: representa a la Entropía

: representa a la probabilidad de que ocurra

Para nuesto caso, usaremos la fórmula en cada llamada recursiva para actualizar el valor de la entropía de cada pregunta (o nodo en la lista), quedando de la siguiente forma:

Donde:

: es la probabilidad de 1’s dentro de la columna/categoría/pregunta. Resultado de la división entre el número de 1’s existentes sobre el total.

: es la probabilidad de 0’s dentro de la columna/categoría/pregunta. Resultado de la división entre el número de 0’s existentes sobre el total.

Para el cálculo del logaritmo base dos nos apoyamos en la librería “numpy”.

## Lista de nodos.

Cómo hemos mencionado anteriormente la lista de nodos es una parte fundamental para el correcto funcionamiento del programa, es creado al iniciar el programa y leer los datos cargados, en cada nodo de esta lista se almacenan datos importantes usados en el resto del programa.

A lo largo de la ejecución del programa esta misma lista de nodos se mantiene, pero con cambios en cada llamada recursiva, junto con el progreso de la aplicación irán cambiando los datos que contiene cada nodo, como la cantidad de 1’s, de 0’s, lo que nos genera la necesidad de recalcular en todo momento la entropía, para después volver a buscar el mejor canditado para la pregunta que viene.

El progama irá filtrando en cada pregunta a los items dentro del dataset, eliminando a quienes no cumplan con la condición actual de la pregunta, esto nos ayuda a ir reduciendo a la mitad del dataset en cada llamada recursiva.

## Ejemplo en ejecución.

Ahora es hora de probar, con el dataset de prueba, el funcionamiento del programa. Veamos como elije la mejor pregunta posible.

Pensemos en un animal, pero no se lo digamos a nadie: “Ornitorrinco”.

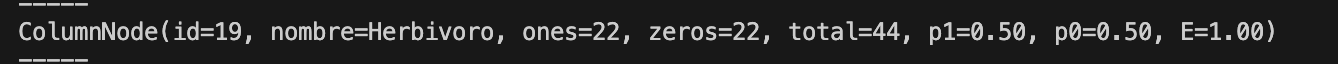




Figura 1 Primera pregunta de ejemplo

Vemos que al ejecutar el programa la entroía más alta la logra la característica de ser Herbívoro, es entonces que nos llega la primera pregunta.





Figura 2 Segunda pregunta de ejemplo

Esta segunda pregunta, al igual que la anterior tiene una alta entropía, y nos cuestiona si nuesto animal es acuático.



A black background with white text

AI-generated content may be incorrect.

Figura 3 Tercera pregunta de ejemplo

Nuestra tercera pregunta con entropía de 0.98 nos recuerda a la canción de Phineas y Ferb, es un mamífero semiacuático agente P, por lo que responderemos que sí.



A black background with white text

AI-generated content may be incorrect.

Figura 4 Cuarta pregunta de ejemplo

Sabemos que es semiacuático, lo que nos lleva a responder sí a la cuarta pregunta, con entropía de 0.97.



A black background with white text

AI-generated content may be incorrect.

Figura 5 Quinta pregunta de ejemplo

El ornitorrinco es de los únicos animales mamíferos que pone huevos, por lo que respondemos que sí.

A black background with white text

AI-generated content may be incorrect.

Figura 6 Mensaje de victoria de la computadora

Podemos observer que para un pequeño número de items dentro del dataset, el programa logra exitosamente adivinar nuestro animal en pocas preguntas.

Es importante notar el total en cada momento del programa, para esto véase la siguiente tabla.

Tabla 2 Comparación entre pregunta y número de items en el ejemplo

|  |  |
| --- | --- |
| **Pregunta** | **Total de items** |
| 1 | 44 |
| 2 | 22 |
| 3 | 12 |
| 4 | 5 |
| 5 | 3 |

Vemos que al menos para las primeras preguntas se cumple la reducción a la mitad, lo que nos indica que el método propuesto tiene una respuesta positiva, al menos para este ejemplo.

# Conclusión.

En esta práctica se implementó árbol binario junto con un dataset que abarcará la mayor cantidad de elementos posibles y con la capacidad de agregar elementos que no se encuentren en el dataset para que el modelo crezca con información del usuario, de esta manera se logró acertar en lo que se encontraba dentro del dataset, lamentablemente se encuentra limitado por el tamaño actual del dataset, sin embargo, tiene el potencial de crecer entre más usuarios lo usen.

De esta manera se comprendió la importancia de obtener gran cantidad de datos para mejorar la predicción del programa.

El conocimiento y experiencia adquiridos en esta primera práctica es fundamental para aplicarlos a las próximas prácticas, no solo de esta materia sino de la carrera en general.

# Referencias.

*UCI Machine Learning Repository*. (2021). Uci.edu. https://archive.ics.uci.edu/dataset/53/iris

Manimala. (2017). *Boston House Prices*. Kaggle.com. https://www.kaggle.com/datasets/vikrishnan/boston-house-prices/data

aleksandrapozorska. (2025, September 19). *Boston House Prices | Regression*. Kaggle.com; Kaggle. https://www.kaggle.com/code/aleksandrapozorska/boston-house-prices-regression/notebook

Mehmet Akturk. (2020). *Diabetes Dataset*. Kaggle.com. https://www.kaggle.com/datasets/mathchi/diabetes-data-set

*UCI Machine Learning Repository*. (2023). Uci.edu. https://archive.ics.uci.edu/dataset/186/wine+quality

Bhavik Jikadara. (2023). *Car Price Prediction Dataset*. Kaggle.com. https://www.kaggle.com/datasets/bhavikjikadara/car-price-prediction-dataset

*UCI Machine Learning Repository*. (2019). Uci.edu. https://archive.ics.uci.edu/dataset/165/concrete+compressive+strength