|  |  |
| --- | --- |
| **Nombre:** | Luis de la Garza González |
| **Matrícula:** | al03101869 |
| **Nombre del curso:** | Machine Learning. |
| **Trabajo:** | Actividad 03.- Cálculo Vectorial y Probabilidades. |
| **Nombre del profesor:** | Mtro. Igor García Atutxa |
| **Fecha:** | 28 de septiembre de 2025 |

**Contenido**

[**Objetivo** 3](#_Toc205456566)

[**Instrucciones:** 4](#_Toc205456567)

[**Objetivo** 5](#_Toc205456568)

[**Desarrollo** 6](#_Toc205456569)

[**Conclusiones** 23](#_Toc205456570)

[**Liga al código en Github** 24](#_Toc205456571)

# **Objetivo**

Aplicar técnicas de diferenciación y análisis de funciones utilizando Python en Jupyter Notebook, integrando conceptos de cálculo y probabilidad.

# **Instrucciones:**

**Utiliza un cuaderno de Jupyter Notebook y el lenguaje de programación Python para realizar las siguientes operaciones:**

1. **Derivadas**

Primera derivada de las siguientes funciones:

1. f(x)= 5x(3x+1) – 8x2 − 3x

Segunda derivada de las siguientes funciones:

* A. f(x)=4x4+6x2+e4x2f(x) = 4x^4 + 6x^2 + e^{4x^2}
* B. f(x)=12x3+8x

1. **.**

Calcula las primeras derivadas parciales de las siguientes funciones:

* A. f(x,y) = 4x3 y + 5y2 − 7x
* B. f(x,y) = y2 / 1 − ex

Calcula las segundas derivadas de las siguientes funciones:

* A. f(x,y)= 3x5 y + y3 / 3
* B. f(x,y)=sin−1 x3 / (cos 2y)2

1. **Investiga sobre diferentes implementaciones del algoritmo de propagación inversa y construye un ejemplo en Python donde se compruebe su funcionamiento.**
2. **Dada la siguiente situación:**

Los dispositivos electrónicos construidos con transistores en la década de 1990 presentaban dos fallas principales: la fuente de alimentación y el circuito principal (tarjeta principal). Según las estadísticas de esa época, un taller promedio recibía de manera mensual un 30% de equipos por fallas en la fuente de alimentación y la efectividad de reparación era del 75% si la falla estaba en la tarjeta principal.

* Calcula la probabilidad de que un equipo que haya ingresado al taller por problemas en la tarjeta principal.
* Implementa tu solución en un cuaderno de Jupyter Notebook utilizando Python.

1. **Utilizando un cuaderno de Jupyter Notebook y el lenguaje de programación Python, realiza las siguientes operaciones:**

* Importa a tu espacio de trabajo el siguiente dataset: Forest Fires Dataset en Kaggle
* Crea un arreglo que contenga los valores de la columna etiquetada por: #temp
* Construye el histograma referente al arreglo creado en el punto anterior.
* Evalúa si la gráfica obtenida tiene semejanzas con la distribución normal y obtén los siguientes estadísticos: Media, Mediana y Desviación estándar.
* Dibuja la gráfica de la distribución gaussiana inferida por los datos, en caso de que corresponda, y evalúa la suposición realizada en el punto anterior.

# **Desarrollo**

**Utilizando un cuaderno de Jupyter Notebook y el lenguaje de programación Python realiza las siguientes operaciones:**

1. **Derivadas**
2. Primera derivada de las siguientes funciones:
3. Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

   El contenido generado por IA puede ser incorrecto.f(x)= 5x(3x+1) – 8x2 − 3x
4. Segunda derivada de las siguientes funciones:

* A. f(x)=4x4+6x2+e\*\*(4x2)
* B. f(x)=1 / 2x3 + 8x

**Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.**

**3.- Investiga sobre diferentes implementaciones del algoritmo de propagación inversa y construye un ejemplo en Python donde se compruebe su funcionamiento.**

La propagación inversa es un proceso clave en el entrenamiento de redes neuronales. Se basa en el principio de optimización por descenso de gradiente y permite ajustar los pesos y sesgos de las neuronas en cada capa de la red. Este proceso se divide en dos fases: la propagación hacia adelante, donde se calculan las salidas y se compara con las salidas deseadas, y la propagación hacia atrás, donde se calculan los gradientes del error y se utilizan para actualizar los pesos y sesgos. La retropropagación es fundamental para aprender representaciones complejas y mejorar la precisión de las predicciones en redes neuronales profundas.

**Los tipos de propagación inversa en inteligencia artificial incluyen:**

* **Propagación inversa**: Es el algoritmo de entrenamiento más común para redes neuronales, que permite ajustar los pesos del modelo para minimizar la pérdida.
* **Redes neuronales recurrentes (RNN)**: Utilizan datos de series temporales y tienen memoria de lo que sucedió en capas anteriores, lo que les permite procesar información en tiempo real.
* **Redes neuronales de memoria a largo/corto plazo (LSTM)**: Son una forma avanzada de RNN que puede recordar información a lo largo de varias capas, útil en aplicaciones como el reconocimiento de voz.
* **Propagación inversa en Python:** Se implementa para entrenar redes neuronales, donde se calculan los gradientes y se ajustan los pesos para reducir el error.
* **Con momento**: Añade un término que suaviza el aprendizaje y evita oscilaciones
* **Con regularización**: Penaliza pesos grandes para evitar sobreajuste
* **Método de corrección de errores:** Comienza con el resultado de la red neuronal y realiza el proceso en sentido inverso para llegar al principio, detectando errores para mejorar la exactitud.

Texto

El contenido generado por IA puede ser incorrecto.**Ejemplo en Python:**

**Ejecutamos la celda con 50,000 épocas y obtenemos los siguientes resultados:**

|  |
| --- |
| Salida final en la época: 0  [[0.73497243]  [0.75492189]  [0.76945373]  [0.78668195]]  Salida final en la época: 1  [[0.69155319]  [0.70849376]  [0.72111898]  [0.73632909]]  Salida final en la época: 2  [[0.64827323]  [0.66195116]  [0.67197968]  [0.68461396]]  .  .  .  Salida final en la época: 49998  [[0.00614437]  [0.99346787]  [0.99366943]  [0.00636152]]  Salida final en la época: 49999  [[0.0061443 ]  [0.99346794]  [0.99366949]  [0.00636146]]  Resultados esperados:  [[0]  [1]  [1]  [0]]  Salida final después del entrenamiento:  [[0.0061443 ]  [0.99346794]  [0.99366949]  [0.00636146]] |

Podemos ver como en cada época del ciclo, la salida final se va aproximando a los resultados esperados.

1. **Dada la siguiente situación:**

Los dispositivos electrónicos construidos con transistores en la década de 1990 presentaban dos fallas principales: la fuente de alimentación y el circuito principal (tarjeta principal). Según las estadísticas de esa época, un taller promedio recibía de manera mensual un 30% de equipos por fallas en la fuente de alimentación y la efectividad de reparación era del 75% si la falla estaba en la tarjeta principal.

* Calcula la probabilidad de que un equipo que haya ingresado al taller por problemas en la tarjeta principal.
* Implementa tu solución en un cuaderno de Jupyter Notebook utilizando Python.

**Vamos a interpretar el enunciado como: calcular la probabilidad de que un equipo tenga una falla en la tarjeta principal dado que fue reparado, entonces usamos la regla de Bayes**

**Aplicando la regla de Bayes**

Queremos calcular:

Donde:

* (asumimos que las fuentes siempre se reparan)

Entonces:

Finalmente:

**Implementa tu solución en un cuaderno de Jupyter Notebook utilizando Python.**

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

# **Conclusiones**

1. Cálculo Diferencial y Modelado de IA

Las derivadas y los gradientes nos permiten descubrir el comportamiento local de las funciones, optimizar modelos y, en general, ver la sensibilidad en sistemas complejos. Con la diferenciación, sin embargo, no solo facilitamos una técnica matemática, sino que nos permitimos ir más allá en algoritmos como la retropropagación (donde también se utiliza la regla de la cadena) y el descenso de gradiente para cambiar parámetros en redes neuronales.

El cambio a funciones vectoriales y matrices jacobianas hace que la conexión entre el cálculo multivariable y el álgebra lineal sea más robusta, lo que permite analizar sistemas con múltiples

Una función vectorial: la matriz jacobiana, una matriz cuadrada cuyos elementos son derivadas parciales. Ilustra cómo cambian los valores de la función con respecto a sus entradas.

2. El Papel de la Probabilidad en el Razonamiento Estadístico

Cuando estudiamos probabilidad, estamos modelando rigurosamente fenómenos aleatorios. Distribuciones como la de Bernoulli/Binomial y Gaussiana modelan la incertidumbre, que luego puedes usar de manera directa para tomar decisiones. Aplicar teoremas como el teorema de Bayes y las reglas de suma y producto puede ser muy útil para analizar probabilidades condicionales y conjuntas.

Una de las cosas más importantes en la IA es poder ver qué distribución se ajusta mejor a nuestros datos. La probabilidad, entonces, proporciona un respaldo numérico si la intuición falla para apoyar o refutar hipótesis y guiar la toma de decisiones.

3. Aplicación Computacional con Python

Usando Jupyter Notebook, `SymPy`, `NumPy`, `Matplotlib` y `Pandas`, puedes intentar aplicar cosas teóricas. Ya sea cálculo simbólico de derivadas, visualización de histogramas y evaluación de distribuciones, Python es la herramienta ideal entre una teoría matemática y la experimentación computacional.

Reflexión Final

La amalgama de cálculo, probabilidad y programación aprovecha el potencial para que los problemas del mundo real en inteligencia artificial sean modelados, examinados y, en última instancia, resueltos. Estas ayudas nos proporcionan no solo ideas sobre cómo se comporta el modelo, sino también formas de optimizarlos y validarlos matemáticamente. El dominio de estos fundamentos es muy crucial para crear soluciones inteligentes, confiables e impecables.

# **Liga al código en Github**

<https://github.com/luisgg121/ML-Actividad-03.git>