

Tarea #: 4

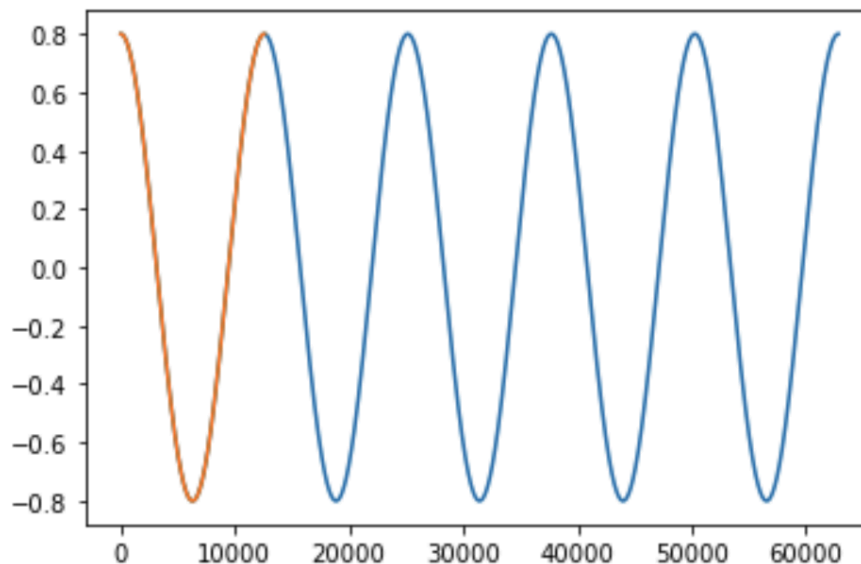
Tema: Clasificación de datos utilizando texto

Fecha entrega: 11:59 pm Junio 06 de 2025

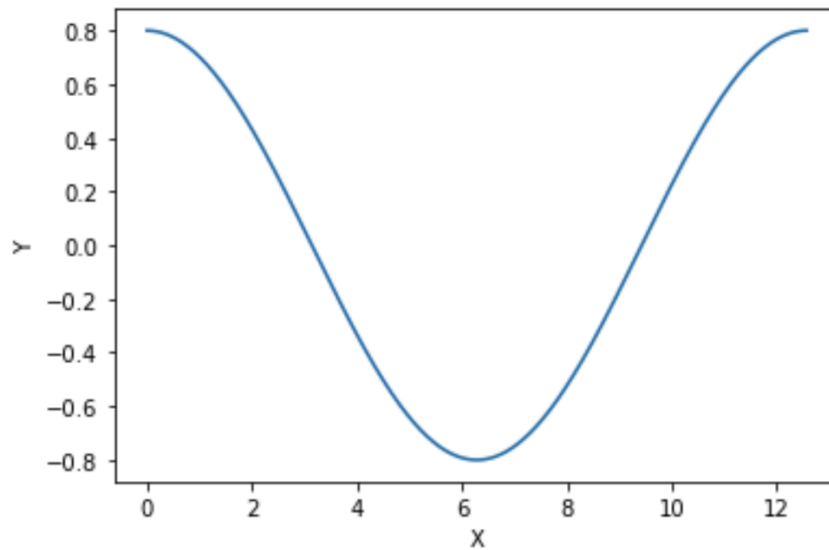
Objetivo: Utilizar modelos de regresión logística y lstm para crear un modelo de clasificación utilizando datos reales .

Entrega: Crear una rama utilizando el mismo repositorio de la tarea 1, 2 y 3, crear otra carpeta llamada tarea 4, solucionar el problema y crear un pull request sobre la master donde me debe poner como reviewer (entregas diferentes tienen una reducción de 0.5 puntos).

1. Gradiente descendiente (15%), estaba en un parque de la ciudad y escuche un ruido constante. Entonces tomé mi celular y realice la grabación del sonido, observando la siguiente gráfica.



Cualquier sonido cosenoidal tiene la forma $\hat{y} = a * \cos(bx + c) + d$, nuestro error debería ser LSE (Least Square Error) entonces nuestra función de error se puede escribir como $e(x) = (y - \hat{y})^2$. Ahora con la función de error tenemos que derivar a, b, c y $d=0$ (Calcula el gradiente que son las derivadas parciales con respecto a a, b, c) y aplicar el algoritmo de gradiente descendiente (Considerando que el gradiente funciona solo con funciones convexas vamos a tomar la parte de la señal convexa, para encontrar los parámetros).



En kaggle Descarga el archivo training.csv de la competencia <https://www.kaggle.com/competitions/descubrir-senal-sonido-2025> el archivo training.csv aplicar el algoritmo de gradiente descendente que consiste en:

- Calcular los gradientes que son las derivadas parciales e iniciar los valores en random
- Iterar por muchos epochs (Muchas iteraciones a los datos)
- Por cada epoch toma un numero de muestras aleatorias, calcula el gradiente y ajusta los valores anteriores utilizando el learning rate.
- Termina cuando el numero de epochs es alcanzado

Validar el error y utilizar el siguiente código como referencia

```
def y_predict(a,b,c,d,x):
    return a * math.cos(b*x + c) + d

lr = 0.05
n = len(x)
batch = 100
epochs = 3000
rsl = []
a = random.random()
b = random.random()
c = random.random()
d = 0
for i in range(epochs):
    a_gradiente = 0
    b_gradiente = 0
    c_gradiente = 0
    d_gradiente = 0
    e = 0
    for m in range(batch):
        ix = int(random.uniform(0,n))
        e += (y[ix] - y_predict(a,b,c,d,x[ix]))* (y[ix] - y_predict(a,b,c,d,x[ix]))
        a_gradiente += # * de/da
        b_gradiente += # * de/db
        c_gradiente += # * de/dc

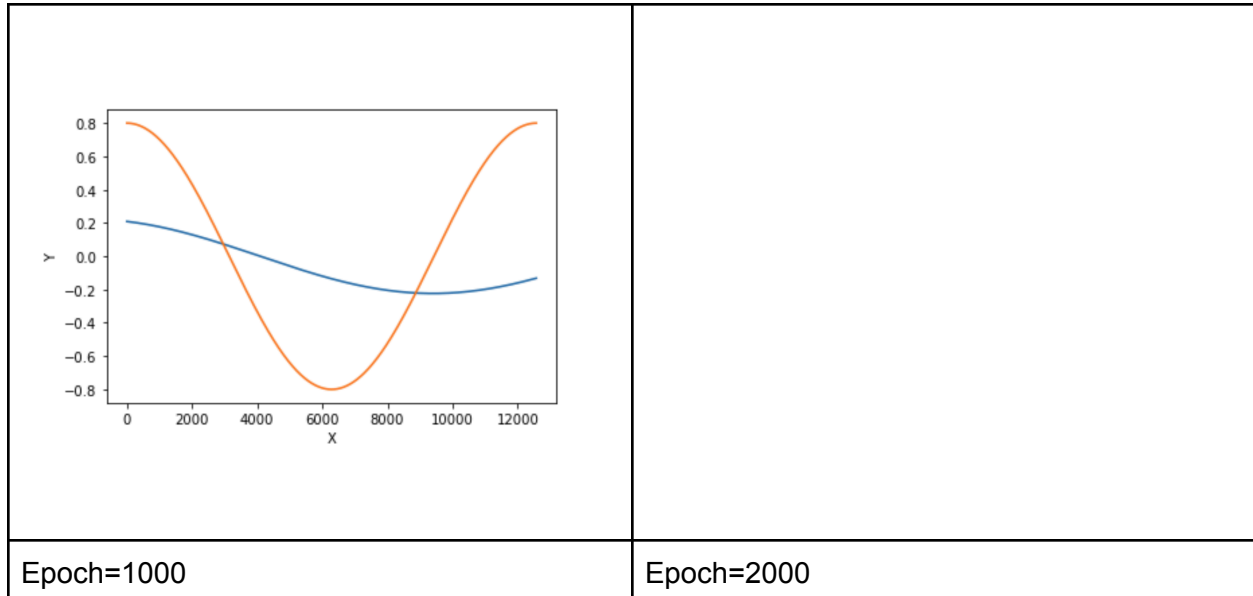
    a = a - lr * a_gradiente/batch
    b = b - lr * b_gradiente/batch
    c = c - lr * c_gradiente/batch

    e = e/batch
    rsl.append([a,b,c,d,e])
    print(f"error:{e} period:{b} amplitud:{a} constant: {c} ")
```

$$E = (y - a \cos(bx + c) + d)^2$$

Utiliza los parámetros del vector rsl y crea la siguiente tabla con los datos del dataset y la función predict aprendida hasta ese epoch..

Epoch=10 (Ejemplo blue es lo aprendido)	Epoch=400



Por último utiliza los parámetros aprendidos, y crea la señal por un periodo más largo de tiempo
`long_s = [y_predict(a,b,c,d,xi) for xi in np.arange(x_min , 20*math.pi, 0.001).astype(np.float32)]`
 , dibuja la señal, y escúchala utilizando la librería (`import sounddevice as sd` y el método `sd.play(long_s)`)
 Ajunta el audio generado en el github y en kaggle envía la respuesta de la predicción del archivo test.

2. En kaggle <https://www.kaggle.com/t/fdbb3da93b784db2ad8c98d4e616a736> Descarga el archivo `training.csv` de la competencia , vamos a hacer un clasificador del tipo de contenido basados en el título:
 - a. Leer los datos en un dataframe (Puede utilizar la librería `request`), la variable objetivo es predecir la columna categoría (Se debe transformar en Entretenimiento, Deportes, Película y Animación, Educación y Otros) utilizando el texto del título. Para eso es necesario hacer las siguientes tareas :
 - i. Utilizar todos los datos, y explorar el numero total de palabras únicas en todos los títulos de train y el numero total de repeticiones (Crear un diccionario para saber si la palabra ya fue observada antes e intentar remover tildes y poner el texto en minúsculas), ¿Cuántas palabras hay en el data set?, generar la siguiente matriz.

	Palabra 1	Palabra 2	Palabra 3	...	Palabra n
Título 1	0 (#veces palabra 1 en titulo1)				

Titulo 2		2 (#veces palabra 2 en titulo2)			
Titulo m					

- ii. Normalmente las palabras más comunes son llamadas stop words. Y corresponden a los artículos o preposiciones remover las stop words. Y crear una matriz de correlación utilizando dummy variables para las variables objetivo “categoria”. También generar la matrix nueva eliminando los stopwords y dividimos por el numero total de palabras en cada título. Esta matriz se llama TF (Term Frequency)¹

	Palabra 1	Palabra 2	Palabra 3	...	Palabra n
Titulo 1	0 (#veces palabra 1 en titulo1)/ Total de palabras en el Titulo 1				
Titulo 2		2 (#veces palabra 2 en titulo2)/# Total de palabras en el Titulo 2			
Titulo m					

- iii. Ahora vamos a crear un vector contando el numero de titulos que tiene cada palabra. Este vector tiene “n” elementos y el valor es el número de títulos que tiene la palabra. Por último vamos a transformar el numero calculando el $\log(\text{Total de documentos} / (\text{Numero de documentos con la Palabra } i + 1))$. El +1 es para q no quede el numero indeterminado cuando algún valor es “0”, el vector es llamado IDF(Inverse Document Frequency)²

	Palabra 1	Palabra 2	Palabra 3	...	Palabra n
	$\log((m+1))$				

¹ <https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/>

² <https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/>

	/ (# títulos con la Palabra 1 + 1))				
--	---	--	--	--	--

- iv. Ahora vamos a multiplicar el vector TF (La matriz) * IDF (Vector transpuesto), El resultado es una matriz de m títulos por n palabras. Y dividir el dataset en test y train. ***En producción se debería dividir los datos antes de calcular el TF-IDF , y para calcular la matriz TFIDF en testing es necesario calcular la frecuencia de las palabras en testing utilizando el orden y las palabras en training y utilizar el IDF de training, pero no es necesario para esta tarea.***
- v. Utilizando la matriz vamos a entrenar 3 modelos, una regresión logística, un random forest y una LSTM (utilizando la capa de embedding como la vimos en clase por lo tanto no es necesario el tf idf). Vamos a crear una matriz de confusión y vamos a comparar los 3 modelos. ¿Cuál es el mejor modelo?, incluir métricas como accuracy, precision y recall para cada modelo.
- vi. Envía el resultado de cada modelo a kagle y documenta el accuracy de cada uno. Regresión logística, random forest y lstm.