CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

OPTATIVA PROFESIONALIZANTE II: MACHINE LEARNING Y DEEP LEARNING

10° "A"

PRIMERA EVALUACIÓN PARCIAL

Profesor: Dr. Francisco Javier Luna Rosas

Alumno: Joel Alejandro Espinoza Sánchez

Fecha de Entrega: Aguascalientes, Ags.**, 26** de febrero de 2023

Primera Evaluación Parcial

El alumno deberá elaborar un documento (`*.pdf`) y un archivo auto-reproducible (`*.html`), que analice e implemente una red neuronal con la técnica de propagación hacia atrás (Feed-Forward) para clasificar las críticas de cine. El documento deberá contener lo siguiente:

- Portada
- Evidencias de la actividad
- Conclusiones
- Referencias (formato APA)
- Letra Arial, tamaño 12 e interlineado 1.5

a) Una explicación del pre-procesamiento de datos para generar un formato adecuado de los datos.

El análisis de sentimientos, a veces también denominado minería de opiniones, es una conocida sub-disciplina del amplio campo del PLN (Procesamiento del Lenguaje Natural); está relacionado con el análisis de la polaridad de documentos. Una tarea popular en el análisis de sentimiento es la clasificación de documentos basados en las emociones u opiniones expresadas de los autores respecto a un tema en particular. El conjunto de datos de críticas de cine consiste en 50,000 críticas de cine polarizadas etiquetadas como negativas y como positivas. Aquí, positiva significa que una película ha sido clasificada con más de seis estrellas, mientras que negativa significa que una película ha sido clasificada con menos de cinco estrellas.

Es muy importante realizar un correcto preprocesamiento de datos en esta aplicación práctica de minería de opiniones, pues este paso puede afectar la calidad de los resultados obtenidos en la etapa de análisis.

En este paso se suelen aplicar las siguientes estrategias para darle una mejor forma al conjunto de datos:

1) **Eliminar los datos irrelevantes:** Eliminar los datos que no son útiles para el análisis de opiniones, como las fechas, las direcciones, etc.
2) **Limpiar el texto:** Eliminar caracteres especiales, signos de puntuación, números y otros caracteres no alfabéticos que no aportan información útil para el análisis.
3) **Convertir el texto a minúsculas:** Convertir todo el texto a minúsculas para facilitar el procesamiento y para que no se distingan las palabras en mayúsculas y minúsculas.
4) **Eliminar las palabras comunes:** Eliminar las palabras comunes que no aportan información útil para el análisis, como artículos, preposiciones, conjunciones, etc.
5) **Lematización:** Lematizar el texto para reducir las palabras a su forma base, lo que puede ayudar a reducir la complejidad del análisis y aumentar la precisión.
6) **Eliminar palabras clave irrelevantes:** Eliminar palabras clave irrelevantes que no aportan información útil para el análisis.
7) **Tokenización:** Separar el texto en palabras individuales para su posterior análisis.
8) **Normalización:** Normalizar el texto para que todas las palabras estén en la misma forma, como eliminar los acentos o caracteres especiales.
9) **Análisis de sentimientos:** Realizar un análisis de sentimientos para asignar a cada opinión un valor de positividad, negatividad o neutralidad.

Personalmente se realizaron algunas estrategias de limpieza general como la tokenización, eliminación de caracteres irrelevantes, transformación del texto a minúscula, entre otros. Esto pues porque se reconoce que es un procedimiento muy importante, pero no se desarrolló a mayor profundidad (o algunos no se desarrollan en este apartado) por realizar los demás incisos.

```
from IPython.display import clear_output
import pandas as pd

df = pd.read_csv('movie_data.csv')

for i in range(len(df)):
    # 1: Eliminación de datos irrelevantes (eliminamos fechas y números en general)
    df.loc[i,"review"] = df.loc[i,"review"].replace("1", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("2", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("3", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("4", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("5", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("6", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("7", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("8", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("9", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("0", "")

    # 2: Limpieza de texto (eliminamos caracteres irrelevantes)
    df.loc[i,"review"] = df.loc[i,"review"].replace(",", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace(".", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("*", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("!", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("?", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("#", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("$", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("%", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("&", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("/", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("(", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace(")", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("+", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("[", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("]", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("{", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("}", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("-", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("_", "")

    # 3: Conversión a minúscula
    df.loc[i,"review"] = df.loc[i,"review"].lower()

    # 4: Eliminación de otras cadenas comunes
    df.loc[i,"review"] = df.loc[i,"review"].replace("<br /><br />", "")
    df.loc[i,"review"] = df.loc[i,"review"].replace("<br ><br >", "")

    # 7: Tokenización (eliminamos espacios de más)
    df.loc[i,"review"] = df.loc[i,"review"].replace("  ", " ")

    # 8: Normalización (eliminación de letras en exceso)
    df.loc[i,"review"] = df.loc[i,"review"].replace("aa", "a")
    df.loc[i,"review"] = df.loc[i,"review"].replace("hh", "h")
    df.loc[i,"review"] = df.loc[i,"review"].replace("zz", "z")

    clear_output()
    print("Process: " + str(i) + "/50000.   " + str(round((100*i/50000), 3)) + "%")
clear_output()
print(df)
```

Obteniendo el siguiente resultado:

```
                                                  review  sentiment
0      in the teenager martha moxley maggie grace mov...          1
1      ok so i really like kris kristofferson and his...          0
2      spoiler do not read this if you think about wa...          0
3      hi for all the people who have seen this wonde...          1
4      i recently bought the dvd forgetting just how ...          0
...                                                  ...        ...
49995  ok lets start with the best the building altho...          0
49996  the british 'heritage film' industry is out of...          0
49997  i don't even know where to begin on this one "...          0
49998  richard tyler is a little boy who is scared of...          0
49999  i waited long to watch this movie also because...          1

[50000 rows x 2 columns]
```

b) Una explicación del modelo de bolsa de palabras o cualquier otro analizador lingüístico aplicado al dataset de críticas de cine.

Para el dataset de críticas de cine se usó la librería `textblob` como el analizador lingüístico. TextBlob es una biblioteca de procesamiento de lenguaje natural (NLP) de Python que proporciona una interfaz simple y fácil de usar para tareas comunes de NLP, como análisis de sentimientos, extracción de frases clave, corrección ortográfica, lemantización, tokenización, entre otras.

TextBlob también puede ser utilizado para crear una representación de bolsa de palabras que consiste en una representación vectorial simplificada de un documento que se utiliza comúnmente en el procesamiento de lenguaje natural. Para crear una bolsa de palabras con esta librería, primero se debe tokenizar el texto utilizando el método `words`, que divide el texto en una lista de palabras. Luego, se utiliza la función `Counter` del módulo `collections` de Python para contar la frecuencia de cada palabra en el texto. Finalmente, se puede representar el texto como un vector de frecuencia de palabras, donde cada entrada en el vector representa la frecuencia de una palabra particular en el texto.

Al conjunto de datos dado para la presente evaluación se le han añadido dos columnas más: la polaridad y la subjetividad. La polaridad y la subjetividad son dos aspectos importantes que se pueden analizar en el lenguaje natural y que pueden

ser útiles en una variedad de aplicaciones, como el análisis de opiniones, la evaluación de sentimientos, la clasificación de texto, la detección de spam, entre otros.

La polaridad se refiere a la medida en que un texto o una palabra expresa una opinión positiva o negativa. En TextBlob, la polaridad se representa como un valor numérico que oscila entre -1 y 1, donde los valores negativos indican una polaridad negativa y los valores positivos indican una polaridad positiva.

Por otro lado, la subjetividad se refiere a la medida en que un texto o una palabra son objetivos o subjetivos. En TextBlob, la subjetividad se representa como un valor numérico que oscila entre 0 y 1, donde los valores cercanos a 0 indican que el texto o la palabra son objetivos y los valores cercanos a 1 indican que son subjetivos.

```
import nltk
from textblob import TextBlob

#nltk.download('punkt')

polarities = []
subjectivities = []

for i in range(len(df)):
    text = df.iloc[i,0]
    blob = TextBlob(text)

    polarity = blob.sentiment.polarity
    subjectivity = blob.sentiment.subjectivity

    polarities.append(polarity)
    subjectivities.append(subjectivity)

    clear_output()
    print("Process: " + str(i) + "/50000.    " + str(round((100*i/50000), 3)) + "%")

clear_output()
df["polarity"] = polarities
df["subjectivity"] = subjectivities
print(df)
```

Obteniendo el siguiente resultado:

```
                                                  review  sentiment  polarity  \
0      in the teenager martha moxley maggie grace mov...          1  0.145312
1      ok so i really like kris kristofferson and his...          0  0.051587
2      spoiler do not read this if you think about wa...          0 -0.019565
3      hi for all the people who have seen this wonde...          1  0.520000
4      i recently bought the dvd forgetting just how ...          0  0.146667
...                                                  ...        ...       ...
49995  ok lets start with the best the building altho...          0 -0.134259
49996  the british 'heritage film' industry is out of...          0 -0.001936
49997  i don't even know where to begin on this one "...          0 -0.216667
49998  richard tyler is a little boy who is scared of...          0  0.251389
49999  i waited long to watch this movie also because...          1  0.250000

       subjectivity
0          0.452604
1          0.376190
2          0.527692
3          0.657778
4          0.458333
...             ...
49995      0.549306
49996      0.711111
49997      0.533333
49998      0.544444
49999      0.520000

[50000 rows x 4 columns]
```

Como segunda cuestión del inciso, es necesario tratar la representación de bolsa de palabras. Una bolsa de palabras (bag-of-words, BOW) consiste en la representación de un documento mediante el conjunto de las palabras que contiene, sin importar el orden. Dicho de otra manera, cada palabra diferente del documento sería un elemento de la bolsa, donde también se almacena el número de veces que esta palabra se repite a lo largo del documento. En el caso personal, se usa `CountVectorizer` de `scikit-learn` para obtener los recuentos por cada palabra y frase.

```python
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(stop_words='english')
matrix = cv.fit_transform(df.loc[:,"review"])

df_bow = pd.concat([
    pd.DataFrame(df.loc[:,"review"], columns = ["review"]),
    pd.DataFrame.sparse.from_spmatrix(
        matrix,
        columns = cv.get_feature_names_out())], axis = 1)

print(df_bow)
```

Observamos el resultado:

```
                                                  review  aa  aaaaaahhhhhhh  \
0      in the teenager martha moxley maggie grace mov...   0              0
1      ok so i really like kris kristofferson and his...   0              0
2      spoiler do not read this if you think about wa...   0              0
3      hi for all the people who have seen this wonde...   0              0
4      i recently bought the dvd forgetting just how ...   0              0
...                                                  ...  ..            ...
49995  ok lets start with the best the building altho...   0              0
49996  the british 'heritage film' industry is out of...   0              0
49997  i don't even know where to begin on this one "...   0              0
49998  richard tyler is a little boy who is scared of...   0              0
49999  i waited long to watch this movie also because...   0              0

       aaaah  aaaahhhggg  aaaargh  aaagh  aaah  aaargh  aaarrrrrrgggggghhh  \
0          0           0        0      0     0       0                   0
1          0           0        0      0     0       0                   0
2          0           0        0      0     0       0                   0
3          0           0        0      0     0       0                   0
4          0           0        0      0     0       0                   0
...      ...         ...      ...    ...   ...     ...                 ...
49995      0           0        0      0     0       0                   0
49996      0           0        0      0     0       0                   0
49997      0           0        0      0     0       0                   0
49998      0           0        0      0     0       0                   0
49999      0           0        0      0     0       0                   0

       ...  überwoman  ünel  ünfaithful  üvegtigris  üzümcü  ýs  þorleifsson  \
0      ...          0     0           0           0       0   0            0
1      ...          0     0           0           0       0   0            0
2      ...          0     0           0           0       0   0            0
3      ...          0     0           0           0       0   0            0
4      ...          0     0           0           0       0   0            0
...    ...        ...   ...         ...         ...     ...  ..          ...
49995  ...          0     0           0           0       0   0            0
49996  ...          0     0           0           0       0   0            0
49997  ...          0     0           0           0       0   0            0
49998  ...          0     0           0           0       0   0            0
49999  ...          0     0           0           0       0   0            0

       þór  כרמון  יגאל
0        0     0     0
1        0     0     0
2        0     0     0
3        0     0     0
4        0     0     0
...    ...   ...   ...
49995    0     0     0
49996    0     0     0
49997    0     0     0
49998    0     0     0
49999    0     0     0

[50000 rows x 201761 columns]
```

c) Una explicación de la transformación de las palabras en vectores de características (utilice la frecuencia de término – frecuencia inversa de documento TF-IDF o cualquier otra técnica que permita transformar palabras a vectores de características).

Hay que enriquecer esta exploración mediante la frecuencia de término y la frecuencia inversa de documento, o mejor conocida como (TF-IDF) la cual es una medida que refleja la importancia de una palabra dentro de un documento. Consiste en multiplicar dos términos: TF e IDF. Para el primero, se puede usar el recuento que se obtiene de la bolsa de palabras. El término IDF es una medida de cuánta información aporta una palabra, teniendo en cuenta el número de documentos analizados y el número de documentos que contienen dicha palabra:

$$idf = \log \frac{\#\ documentos}{\#\ documentos\ con\ la\ palabra + 1} + 1$$

En el caso propio se usa `TfidVectorizer` de `scikit-learn` para obtener el valor de TF-IDF por cada palabra y frase.

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(df.loc[:,"review"])

df_tfidf = pd.concat([
    pd.DataFrame(df.loc[:,"review"], columns = ["review"]),
    pd.DataFrame.sparse.from_spmatrix(
        tfidf_matrix,
        columns=cv.get_feature_names_out())], axis=1)

print(df_tfidf)
```

Como resultado obtenemos:

	review	aa	aaaaaahhhhhhh	\
0	in the teenager martha moxley maggie grace mov...	0.0	0.0	
1	ok so i really like kris kristofferson and his...	0.0	0.0	
2	spoiler do not read this if you think about wa...	0.0	0.0	
3	hi for all the people who have seen this wonde...	0.0	0.0	
4	i recently bought the dvd forgetting just how ...	0.0	0.0	
...	...	...	...	
49995	ok lets start with the best the building altho...	0.0	0.0	
49996	the british 'heritage film' industry is out of...	0.0	0.0	
49997	i don't even know where to begin on this one "...	0.0	0.0	
49998	richard tyler is a little boy who is scared of...	0.0	0.0	
49999	i waited long to watch this movie also because...	0.0	0.0	

	aaaah	aaaahhhggg	aaaargh	aaagh	aaah	aaargh	aaarrrrrrgggggghhh	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	
49995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
49996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
49997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
49998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
49999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	...	überwoman	ünel	ünfaithful	üvegtigris	üzümcü	ýs	þorleifsson	\
0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	
49995	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
49996	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
49997	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
49998	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
49999	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	þór	כרמון	יגאל
0	0.0	0.0	0.0
1	0.0	0.0	0.0
2	0.0	0.0	0.0
3	0.0	0.0	0.0
4	0.0	0.0	0.0
...	...	...	...
49995	0.0	0.0	0.0
49996	0.0	0.0	0.0
49997	0.0	0.0	0.0
49998	0.0	0.0	0.0
49999	0.0	0.0	0.0

[50000 rows x 201761 columns]

d) Una explicación del modelo de Machine Learning utilizando redes neuronales para clasificar las críticas de cine (el modelo de entrenamiento puede ser Tabla Testing, LOOCV, K-Folds, etc.). La precisión del modelo debe ser del 90% o mayor.

Se decidió omitir el uso de la bolsa de palabras y la representación TF-IDF debido a la gran cantidad de columnas generadas, lo cual produce un error de memoria:

```
MemoryError: Unable to allocate 105. GiB for an array with shape
(352960, 40000) and data type float64.
```

Evitando el error anterior se realizó el siguiente código:

```
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix

#df_final = pd.concat([df, df_bow.iloc[:,1:], df_tfidf.iloc[:,1:]], axis = 1, sort = False)

df_x = df.drop(['review', 'sentiment'], axis = 1)
df_y = df.drop(['review', 'polarity', 'subjectivity'], axis = 1)

x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.2, random_state=42)

# Creación de un objeto MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(100, ),
                    activation = 'logistic',
                    solver = 'lbfgs',
                    alpha = 0.001,
                    learning_rate = 'adaptive',
                    learning_rate_init = 1.9,
                    max_iter=120)

# ajusta el modelo a los datos de entrenamiento
mlp.fit(x_train, y_train)

# evalúa el modelo en los datos de prueba
accuracy = mlp.score(x_test, y_test)

print(accuracy)
```

Obteniendo el siguiente resultado:

```
C:\Users\alexe\Anaconda3\envs\ici-thesis\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:1109: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
0.7666
```

No se logró obtener una mayor precisión debido a la falta de tiempo.

e) Una explicación del análisis comparativo del modelo de Machine Learning utilizando redes neuronales vs otro clasificador. Compare la precisión del modelo, el error del modelo, la precisión negativa (especificidad), la precisión positiva (sensibilidad), falsos positivos, falsos negativos, asertividad positiva y asertividad negativa.

No se desarrolló otro clasificador para realizar el análisis comparativo debido a la falta de tiempo.

Conclusión: Esta evaluación nos permite implementar una red neuronal en un caso de estudio práctico notorio como lo es el procesamiento del lenguaje natural. Además concluyo que esta evaluación nos permite hacer un análisis completo de datos a esta práctica, realizando desde las cuestiones de preprocesamiento pasando por el procesamiento matemático previo a la red neuronal como lo es la representación de bolsa de palabras y la frecuencia de palabras y la frecuencia de palabra por documento. Es también necesario aprender a interpretar los datos como lo hacemos en la comparación de los dos modelos por lo que me pareció una evaluación desafiante pero como alumnos nos deja un muy buen aprendizaje y evaluación de los mismos.

Referencias:

- Anónimo (s.f.) "*Red neuronal artificial*". Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Red_neuronal_artificial.
- Data Scientest (2021) "*Perceptrón. ¿Qué es y para qué sirve?*". Obtenido de Data Scientest: https://datascientest.com/es/perceptron-que-es-y-para-que-sirve.
- Luna, F. (2023) "*El Modelo de McCulloch – Pitts*". Apuntes de ICI 10°.