UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FCFM
FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

PROYECTO INTEGRADOR DE APRENDIZAJE

APRENDIZAJE AUTOMATICO

CÉSAR ALEJANDRO HERNÁNDEZ OROZCO

Matricula: 1990010

GRUPO 003

MAESTRO: JOSÉ ANASTASIO HERNÁNDEZ SALDAÑA

INTRODUCCIÓN:

En este proyecto Integrador de aprendizaje nos vimos con la tarea de aplicar los conocimientos aprendidos en la materia de Aprendizaje Automatico para crear un modelo de Clasificación. Para este proyecto se investigarón cuales serian los mejores modelos para trabajar con este conjunto de datos pero al final se decidio que la manera en que se resolveria este problema seria con una red neuronal.

El proyecto se centra en la predicción de la variable de compromiso (engagement) en un conjunto de datos relacionados con películas, buscando optimizar el rendimiento del modelo para alcanzar un valor de ROC AUC superior a 0.75 tanto en el conjunto de validación como en el de prueba.

Este proyecto se realizo usando el conjunto de datos proporcinado por el profesor que imparte la materia. En este conjunto de datos se presentan distintos datos de cada pelicula y se busca predecir si la pelicula fue vista.

DESARROLLO:

El proyecto consiste en el desarrollo y evaluación de un modelo de red neuronal para predecir el compromiso de los usuarios con el contenido cinematográfico.

Lo primero que se hizo fue identificar el tipo de cada una de las variables:
Variables

1. **id**:
    - Tipo de dato: Entero.
2. **title_word_count**:
    - Tipo de dato: Entero.
3. **document_entropy**:
    - Tipo de dato: Flotante.
4. **freshness**:
    - Tipo de dato: Entero.
5. **easiness**:
    - Tipo de dato: Flotante.
6. **fraction_stopword_presence**:

- Tipo de dato: Flotante.

7. **normalization_rate**:
    - Tipo de dato: Flotante.
8. **speaker_speed**:
    - Tipo de dato: Flotante.
9. **silent_period_rate**:
    - Tipo de dato: Flotante.
10. **engagement**:
    - Tipo de dato: Booleano

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9239 entries, 0 to 9238
Data columns (total 10 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id                          9239 non-null   int64
 1   title_word_count            9239 non-null   int64
 2   document_entropy            9239 non-null   float64
 3   freshness                   9239 non-null   int64
 4   easiness                    9239 non-null   float64
 5   fraction_stopword_presence  9239 non-null   float64
 6   normalization_rate          9239 non-null   float64
 7   speaker_speed               9239 non-null   float64
 8   silent_period_rate          9239 non-null   float64
 9   engagement                  9239 non-null   bool
dtypes: bool(1), float64(6), int64(3)
memory usage: 658.8 KB
```

Para asegurarnos que nuestro modelo no tenga ningun problema nos aseguramos que no existan datos vacios:

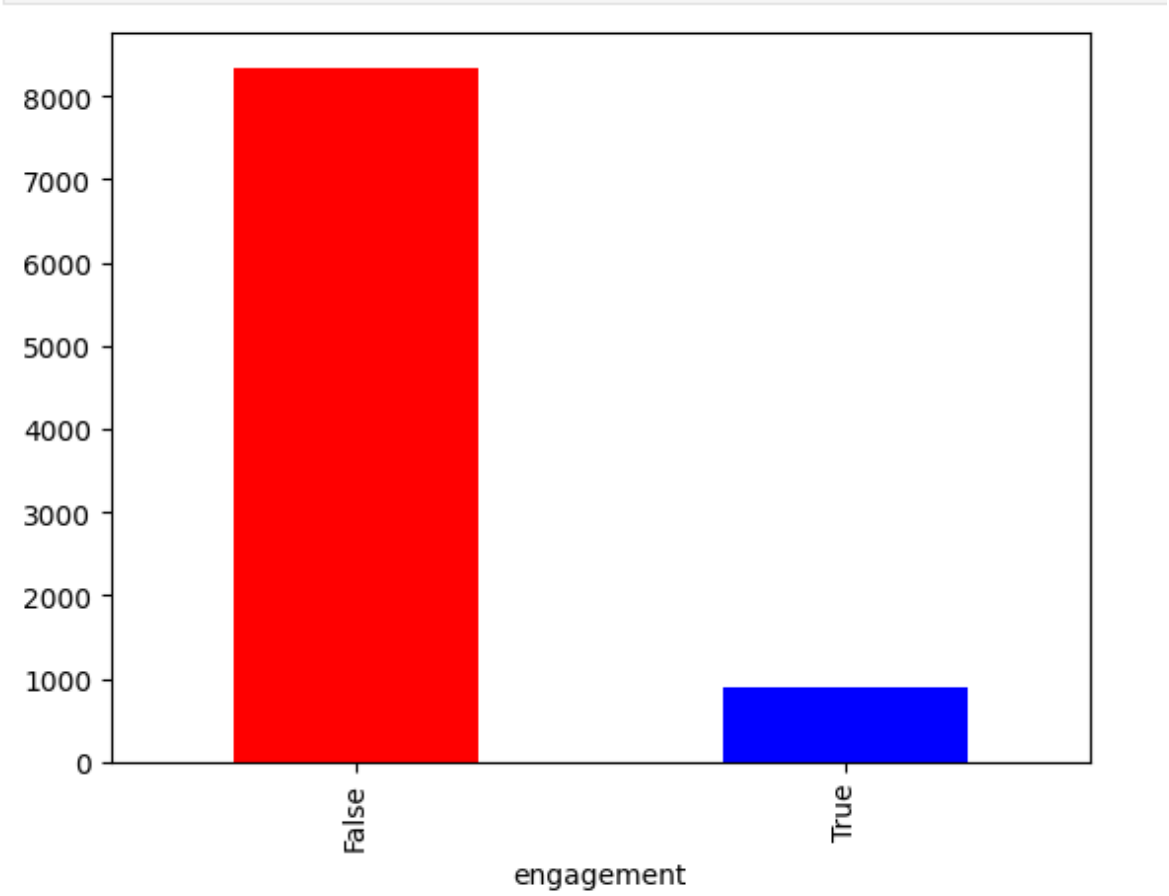
```
id                          0
title_word_count            0
document_entropy            0
freshness                   0
easiness                    0
fraction_stopword_presence  0
normalization_rate          0
speaker_speed               0
silent_period_rate          0
engagement                  0
dtype: int64
```

Despues buscamos conocer un poco como son los datos con los que vamos a trabajar obteniendo los primeros datos y despues realizando un analisis exploratorio de todas las variables:

	id	title_word_count	document_entropy	freshness	easiness	fraction_stopword_presence	normalization_rate	speaker_speed	silent_period_rate	engagement
0	1	9	7.753995	16310	75.583936	0.553664	0.034049	2.997753	0.0	True
1	2	6	8.305269	15410	86.870523	0.584498	0.018763	2.635789	0.0	False
2	3	3	7.965583	15680	81.915968	0.605685	0.030720	2.538095	0.0	False
3	4	9	8.142877	15610	80.148937	0.593664	0.016873	2.259055	0.0	False
4	5	9	8.161250	14920	76.907549	0.581637	0.023412	2.420000	0.0	False

	id	title_word_count	document_entropy	freshness	easiness	fraction_stopword_presence	normalization_rate	speaker_speed	silent_period_rate
count	9239.000000	9239.000000	9239.000000	9239.000000	9239.000000	9239.000000	9239.000000	9239.000000	9239.000000
mean	4620.000000	7.701050	7.792685	14808.587509	84.756029	0.612214	0.021354	2.413320	0.146606
std	2667.213902	3.785066	0.697710	1208.953646	8.303489	0.051872	0.009545	1.588296	0.172030
min	1.000000	1.000000	0.000000	10830.000000	28.210966	0.000000	0.000000	0.000302	0.000000
25%	2310.500000	5.000000	7.594235	14070.000000	80.415060	0.589632	0.014974	1.976727	0.000000
50%	4620.000000	7.000000	7.875103	14750.000000	84.482122	0.613127	0.019843	2.267133	0.104915
75%	6929.500000	10.000000	8.164166	15600.000000	88.388711	0.634585	0.026288	2.539207	0.250927
max	9239.000000	33.000000	9.278573	17430.000000	122.032000	1.000000	0.101990	50.850000	1.168239

Algo que es importante para trabajar tambien es conocer como se distribuye nuestra variable de respuesta. Para ello realizamos una grafica que nos muestre como esta distribuido los datos de la variable engagement:



```
engagement
False    8342
True      897
Name: count, dtype: int64
```

Con esto podemos ver que la variable esta bastante desbalanceada con la mayoria de los datos diciendonos que las personas no han visto la pelicula.

Una vez ya realizamos todos estos analisis empezamos con al creación del modelo. Como se menciono para este trabajo se decidio trabajar con una red neuronal. Asi que asignamos los valores de las variables explicativas a todo el conjunto excepto engagement y la variable de respuesta a engagement.

Para la realización de este trabajo decidi usar la librería de redes neuronales con la que me siento mas familiarizado que es Pytorch.

Lo primero que hicimos fue dividir los datos en cuales serian de entrenamiento y cuales serian de prueba. Se decidio que el 80% de los datos fueran de entrenamiento. Una vez hecho esto se normalizaron todos los conjuntos de datos:

```
array([[ 0.12135517, -0.71506764,  0.89231581, ..., -0.82144258,
        -0.21607589,  0.98189391],
       [-1.18125315, -0.18642899, -0.42136957, ..., -1.06533369,
         0.39699606, -0.849056  ],
       [ 0.33789364, -0.45074832, -0.13432288, ...,  1.24195122,
        -0.14395427,  0.42666111],
       ...,
       [ 0.29098322, -1.77234495, -0.04686395, ..., -1.11863044,
        -0.30552372,  0.84931092],
       [-1.40905011,  0.07789034,  0.26246359, ...,  0.34777309,
         0.14099098, -0.849056  ],
       [ 0.99651582, -0.45074832, -0.17760988, ..., -0.05439541,
         0.50648945, -0.849056  ]])
```

Una vez estuvieron normalizados se convirtieron en tensores que se puedan usar en Pytorch:

```
tensor([[ 0.1214, -0.7151,  0.8923,  ..., -0.8214, -0.2161,  0.9819],
        [-1.1813, -0.1864, -0.4214,  ..., -1.0653,  0.3970, -0.8491],
        [ 0.3379, -0.4507, -0.1343,  ...,  1.2420, -0.1440,  0.4267],
        ...,
        [ 0.2910, -1.7723, -0.0469,  ..., -1.1186, -0.3055,  0.8493],
        [-1.4091,  0.0779,  0.2625,  ...,  0.3478,  0.1410, -0.8491],
        [ 0.9965, -0.4507, -0.1776,  ..., -0.0544,  0.5065, -0.8491]])
```

Para la obtención de este modelo se intentaron distintos modelos de redes neuronales, algunas con muchas capas y otras con menos. Al final se termino trabajando con una red neuronal de 2 capas unicamente.

Una vez se creo el modelo lo unico que se hizo fue incializarlo. Para el entrenamiento de este modelo se trabajo con 100 epochs, se trabajo con lotes de 64 datos y con una tasa de aprendizaje de 0.001.

Una vez se definio el entrenamiento del modelo se configuraron tambien las cosas necesarias para trabajar adecuadamente con la validación cruzada de 5 divisiones.
El proceso de entrenamiento fue algo tardado debido a que tenia que hacer muchas epochs y el conjunto de datos era bastante grande. Pero una vez terminaron las epocas se obtuvieron los siguientes valores de la validación cruzada:

```
Cross-Validation AUC Scores: [0.8765398051112336, 0.883391953597433, 0.8736219095808136, 0.9060224608169815, 0.8582774958217697]
Mean Cross-Validation AUC: 0.8795707249856463
```

Podemos ver que en general nuestro modelo conto con un valor de curva de ROC del 87% en los conjuntos de entrenamiento. Un valor bastante adecuado para lo que se busca obtener con este proyecto.

Una vez obtuvimos un modelo adecuado lo comparamos contra nuestros valores de prueba y obtuviemos los siguientes resultados:

```
Test ROC AUC Score: 0.8983345036820974
Test Accuracy: 0.9383116960525513
Classification Report:
              precision    recall  f1-score   support

         0.0       0.95      0.98      0.97      1683
         1.0       0.72      0.50      0.59       165

    accuracy                           0.94      1848
   macro avg       0.84      0.74      0.78      1848
weighted avg       0.93      0.94      0.93      1848
```

Como podimos ver con todo lo realizado obtuvimos un modelo bastante fuerte con una precision muy cercana al 90%. Podemos consultar todo el codigo y el modelo para obtener estos resultados en el archivo de Jupyter adjunto a esta practica.

Con estos resultados obtenidos podemos estar satisfechos con los resultados obtenidos.

Conclusión:

Este proyecto nos permitio consolidar todos los conocimientos adquiridos durante todo el curso. Al desarrollar un modelo de red neuronal para la predicción del compromiso del usuario, el proyecto contribuye a mejorar las estrategias de personalización y marketing en la industria del entretenimiento, demostrando la capacidad de utilizar técnicas avanzadas de machine learning para resolver problemas reales.

Fue un trabajo bastante complicado y necesito bastante trabajo debido a que tuve que pasar mucho tiempo asegurandome que la red neuronal cumpliera con lo solicitado en el trabajo.

Se tuvieron que cambiar bastante cosas como el numero de neuronas, el rate de aprendizaje o cuantas epochs usaria el modelo.

En resumen, el proyecto integrador ha proporcionado una valiosa experiencia en la aplicación práctica de redes neuronales para la clasificación de datos, contribuyendo al desarrollo de habilidades técnicas avanzadas y ofreciendo una base sólida para futuros trabajos en el campo del machine learning.

Bibliografia:

Keita, Z. (2024, March 5). *Clasificación en machine learning: Introducción*. Datacamp.com; DataCamp. https://www.datacamp.com/es/blog/classification-machine-learning