DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Inteligencia de negocios 202220 – Proyecto #1

PROFESORA: Haydemar Nuñez

Nombres	Apellidos	Código	Login
María Sofía	Álvarez López	201729031	ms.alvarezl
Brenda Catalina	Barahona Pinilla	201812721	bc.barahona
Alvaro Daniel	Plata Márquez	201820098	ad.plata

1. Comprensión del negocio y enfoque analítico

Oportunidad/Problema de negocio	Ayudar al personal médico a automatizar el proceso de clasificación de los pacientes en 5 diferentes tipos de condiciones o enfermedades a partir de las descripciones escritas por ellos
Descripción del requerimiento desde el punto de vista de aprendizaje de máquina	Crear 3 diferentes tipos de modelo de aprendizaje automático de análisis de textos para evaluar las descripciones de los pacientes y determinar en qué categoría de condición o enfermedades clasificarlos

Detalles de la actividad de minería de datos		
Tipo de aprendizaje	**Tarea de aprendizaje**	**Algoritmo e hiper-parámetros utilizados (con la justificación respectiva)**
Supervisado	Clasificación	Multinomial Naive Bayes. Utilizamos los hiperparámetros alpha (1), fit_prior (True) y ngram_range (1,1). Los valores para estos hiperparámetros fueron definidos después de hacer una exploración de hiperparámetros y encontrar que estos serían los valores que mejores resultados darían.
Supervisado	Clasificación	One-vs-rest. Utilizamos los hiperparámetros c (2), kernel (rbf) y Degree (1). Los valores para estos hiperparámetros fueron definidos después de hacer una exploración de hiperparámetros y encontrar que estos serían los valores que mejores resultados darían
Supervisado	Clasificación	LSTM (Long Short Term Memory) es un tipo de red neuronal recurrente, RNN. En este caso, los hiperparámetros ajustados fueron la cantidad de capas LSTM (se consideró la opción de tener entre 1 y 3 capas ocultas – además de la de inicio y final). Se ajustó también el número de neuronas por capa (entre 8, 16, 32 y 64) y la tasa de *dropout* de información, entre 0.1, 0.25 y 0.5 (Siendo 0.5 el valor máximo).

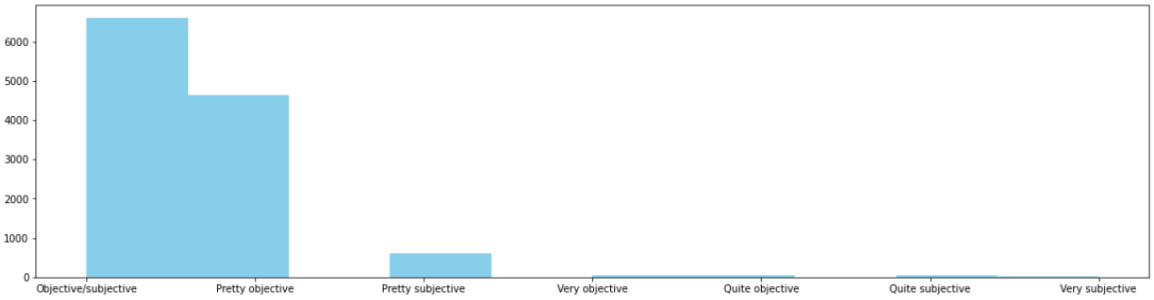
2. **Comprensión y preparación de los datos**

Los datos con los que se trabajará en este proyecto consisten en un conjunto de textos de historias clínicas de 12000 pacientes escritos en lenguaje natural por personal médico, y la categoría de condición o enfermedad a la que esta historia clínica corresponde. Estas categorías pueden ser las siguientes:

a. Neoplasms (Neoplasias).
b. Digestive system diseases (Enfermedades del sistema digestivo).
c. Nervous system diseases (Enfermedades del sistema nervioso).
d. Cardiovascular diseases (Enfermedades cardiovasculares).
e. General pathological conditions (Condiciones patológicas generales).

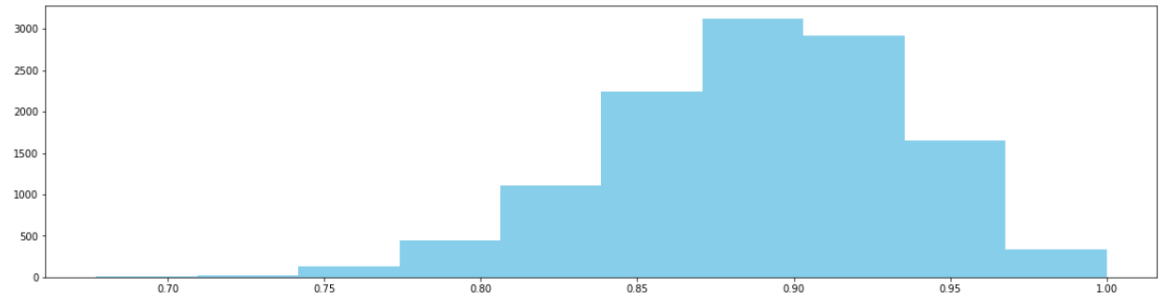
Durante el proceso de perfilamiento de los datos pudimos observar que contamos con un data set de 12000 registros que corresponden a las historias de los pacientes. Estos registros se componen del campo “medical_abstracts” (las historias médicas) y “problems_described” (categoría).

Realizamos un análisis de objetividad en los textos de las historias clínicas y vemos que la mayoría no son clasificadas ni como objetivas ni subjetivas (es decir, parecen ser bastante neutras), y la siguiente categoría corresponde a aquellas que son muy objetivas, como es de esperarse en un contexto médico.

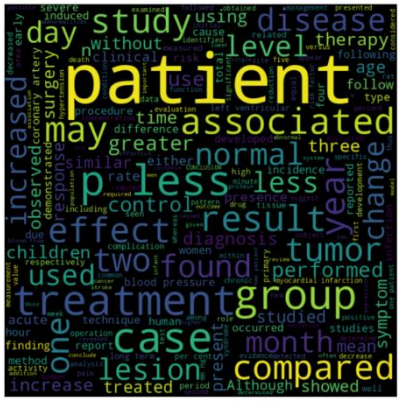


Posteriormente generamos un análisis de la calidad ortográfica de las historias clínicas y podemos ver que, en general, nuestro dataset tiene buena ortografía, pues los puntajes se distribuyen entre 0.7 y 1, y tienen una media de 0.89.

La media de la calidad de ortografía es 0.89



Finalmente, generamos una nube con las palabras más populares entre las historias, sin ningún pre-procesamiento hasta ahora:



En esta nube de palabras no hay mayores sorpresas: encontramos mayoritariamente términos que esperaríamos encontrar en historias clínicas. Note que las palabras: "patient", "tumor", "treatment", "study", "diagnosis" y "disease" resaltan. No obstante, note también que hay otras palabras irrelevantes como: "may", "use", "using", "used", e incluso algunas muy similares como "increase" e "increased". Todo esto será tratado más adelante en las fases de eliminación de stop words del inglés, lematización y demás partes del pre-procesamiento de lenguaje natural.

En el análisis encontramos que no hay ningún registro sin una categoría definida, y que cada historia está asociada a una única categoría, lo que corresponde a una clasificación multiclase. De igual forma observamos que no existen historias ni categorías que correspondan a cadenas vacías (“”), de un solo espacio (“ “) o nulas. Tampoco encontramos registros duplicados en el data set. A partir de estos análisis podemos concluir que la calidad del conjunto de datos es bastante buena en términos de nulos y duplicados.

En el proceso de preparación de datos, procedemos inicialmente a dividir nuestro conjunto en la parte de datos de entrenamiento (80%) y la de prueba de validación (20%).

Observamos de igual forma la distribución de la cantidad de historias que existen para cada categoría, y encontramos lo siguiente:

	problems_described
5	3194
1	2103
4	2029
3	1280
2	994

Podemos ver claramente que las clases están notablemente desbalanceadas. Vemos que la clase con más datos es la 5, lo cual es consecuente con el hecho de que esta es la clase más amplia, al tratar condiciones patológicas generales. Debemos clasificar todas las clases igual de bien, pues nos interesa conocer con precisión qué enfermedad posee cada paciente particular. Por esta razón, debemos utilizar técnicas de balanceo de clases. Podemos abordar este problema desde el preprocesamiento, probaremos realizando un submuestreo de los datos, y usando técnicas de oversampling como SMOTE, o en la implementación del algoritmo, diciéndole que estamos trabajando con clases desbalanceadas (aunque esto depende de los algoritmos a utilizar, los cuales se definirán más adelante. En particular, una variación de Naïve-Bayes permite esta implementación). Quisiéramos ver todos los comportamientos, entonces intentaremos explorar estas alternativas. Esto se realiza más adelante, en las secciones de preprocesamiento y modelado (respectivamente) de este documento.

Utilizando el modelo preentrenado de SpaCy, extraemos las palabras más importantes del contexto médico de nuestros datos en una columna adicional. Podemos ver que la mayoría de las oraciones se han reducido a solamente algunas pocas palabras del léxico médico. Con el fin de no sesgar mucho nuestro algoritmo, haremos los algoritmos sobre ambos conjuntos de datos: el preprocesado con y el normal.

Posteriormente realizamos todo el proceso de tokenizacion, donde manejaremos contracciones de las palabras y separaremos las palabras en una lista. De igual forma, aplicamos técnicas como "Stemming" y "Lemmatization" sobre la columna "medical_abstracts".

Posteriormente, eliminamos y/o modificamos el ruido de nuestros datos, es decir, los caracteres no ASCII, pasar de mayúsculas a minúsculas, eliminar la puntuación, reemplazar los números por su versión en palabras, y quitar las palabras vacías (artículos, pronombres, preposiciones).

Nos dimos cuenta de que hay algunas palabras aparecen frecuentemente en varias categorías, lo que hace que no aporten información importante para la clasificación, por lo que decidimos eliminarlas. Algunas de estas palabras son “and”, “Paty”,

“patients”, “p”, “one”, “two”, “three”, “study” y “result”.

3. Modelado y evaluación

Nota: *En este apartado del informe se mencionan únicamente los pasos más importantes de la implementación de los modelos. Para mayor detalle en cada paso y ver pasos adicionales, remitimos al lector al notebook asociado a cada algoritmo.*

Para el objetivo del negocio de automatizar el proceso de clasificar historias creamos 3 modelos de aprendizaje automático utilizando los siguientes algoritmos:

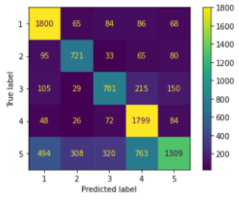
a. Multinomial Naive Bayes (Implementado por Álvaro Plata):

Naive Bayes es una técnica de clasificación que se basa en el teorema de Bayes con el supuesto de que todas las características que predicen el valor objetivo son independientes entre sí. Calcula la probabilidad de cada clase y luego elige la que tiene la mayor probabilidad. Se ha utilizado con éxito para muchos propósitos, pero funciona particularmente bien con problemas de procesamiento del lenguaje natural (NLP). (ICHI.PRO)

Por las razones expuestas en el apartado de “Comprensión y preparación de los datos” decidimos crear 6 modelos con este algoritmo:

i. Vectorización con TF-iDF y undersampling
ii. Vectorización con TF-iDF y oversampling (+ reducción de dimensionalidad)
iii. Sin resampling, usando TF-iDF
iv. Vectorización con Count-Vectorizer y subsampling
v. Vectorización con Count-Vectorizer y oversampling.
vi. Sin resampling, usando Count-Vectorizer

Analizamos la calidad de los modelos creando una matriz de confusión para cada uno. A partir de esto determinamos que el modelo que elegiremos de este algoritmo será el VI, pues tuvo los mejores resultados para las métricas de Precisión y Recall.



Test Report for Multinomial NB Pipe 6

	precision	recall	f1-score	support
1	0.71	0.86	0.78	2103
2	0.63	0.73	0.67	994
3	0.61	0.61	0.61	1280
4	0.61	0.89	0.73	2029
5	0.77	0.41	0.54	3194
accuracy			0.67	9600
macro avg	0.67	0.70	0.66	9600
weighted avg	0.69	0.67	0.65	9600

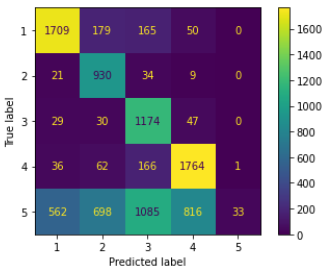
Exportamos el archivo “multinomial_nb.joblib” para utilizarlo en la etapa de validación más adelante

b. OneVsRest (Implementado por Brenda Barahona)

Este algoritmo es una heurística que implementa una estrategia en la cual usa algoritmos de clasificación binaria para hacer la clasificación multiclase. En otras palabras, este algoritmo clasifica por clase, es decir que, para cada uno de los problems described (1,2,3,4,5) habrá un clasificador. Cada uno de estos clasificadores es comparado con los otros.

Ahora, para la implementación de este modelo, usamos varios pipelines, ya que queríamos rectificar cual de todas las configuraciones de preprocesamiento que contemplamos sería la que mejor le sentaría al modelo. Los pipelines que se usaron para este algoritmo fueron: (i, ii, iv, v) de los mencionados anteriormente en el algoritmo Multinomial Naive Bayes

Para cada uno de estos pipelines se hizo la búsqueda de los hiperparámetros, teniendo en cuenta que nuestro objetivo es tener la mejor precisión posible. Con esto en mente, el modelo que tenía un mejor Weighted avg, que en este caso fue el del pipeline 2, con una precisión de 0.73. El valor de los hiperparámetros obtenidos se encuentran en la tabla al comienzo de este documento



Test Report for Multinomial NB Pipe 2

	precision	recall	f1-score	support
1	0.73	0.81	0.77	2103
2	0.49	0.94	0.64	994
3	0.45	0.92	0.60	1280
4	0.66	0.87	0.75	2029
5	0.97	0.01	0.02	3194
accuracy			0.58	9600
macro avg	0.66	0.71	0.56	9600
weighted avg	0.73	0.58	0.48	9600

c. LSTM (Implementado por Sofía Alvarez)

La LSTM (Long-Short Term Memory) es un tipo de red neuronal recurrente (RNN, por sus siglas en inglés) que se desempeña mejor que las RNN tradicionales en términos de memoria [1]. Una RNN es un tipo de red neuronal que permite a las salidas de capas previas ser utilizadas como entradas, teniendo estados ocultos [2]. Las LSTM tienen múltiples capas ocultas. A medida que se pasa a través de una capa, la información relevante se mantiene y la irrelevante se desecha en cada neurona (celda) individual [1]. Asimismo, las LSTM solucionan el problema de desvanecimiento de gradientes que las RNN enfrentan a menudo. Las LSTM cuentan con tres compuertas: INPUT, FORGET y OUTPUT, las cuales se encuentran mejor detalladas en el notebook.

Una vez elegido el modelo, procedemos a hacer la vectorización de los datos que obtuvimos en la fase de preprocesamiento. Como dijimos, el preprocesamiento para los algoritmos basados en secuencias, como las redes neuronales, es diferente. En este caso, optaré por usar dos opciones de embedding.

BioSentVec: Módulo basado en las investigaciones de Zhang et. al [4]. Similar a Sent2Vec (algoritmo de embedding de Google), está basado en la librería fast_text de Facebook para embeddings y clasificación de textos. La librería utiliza las base de datos de PubMed y las notas clínicas de MIMIC-III Clinical Database como corpus para entrenar una red que genera vectores de 700 dimensiones. El procedimiento para usar este modelo es largo y tedioso, pero puede encontrarse anexo a este laboratorio. Debido a que la dimensionalidad sigue siendo elevada, para evitar overfitting, solamente se usará sobre los abstracts y no sobre las entities.

BioWordVec: Desarrollada por los mismos investigadores de [4], es un embedding muy usado en contextos biomédicos. Como es sobre palabras únicamente, preferimos utilizarlo para las entidades médicas extraídas en el preprocesamiento. Este preprocesamiento no arrojó mejores resultados que el realizado con BioSentVec, por lo que en este informe solo hablo de dicho modelo. Para el otro, remítase al notebook asociado.

Para el problema de desbalanceo de las clases hacemos uso de la alternativa de considerar pesos. Así, el modelo podrá prestar mayor atención a las clases minoritarias. No consideramos hacer SMOTE ni subsampling porque para una red neuronal este tipo de entrenamiento es mucho más costoso.

Para la búsqueda de hiperparámetros, queremos encontrar los mejores para el número de capas, de neuronas, y la mejor tasa de dropout. Consideramos solo tres capas ocultas (adicionales a las capas LSTM de inicio y final), así como neuronas entre 8, 16, 32 y 64 para evitar que el modelo haga overfitting. Para el dropout, evaluamos valores entre 0.1, 0.25 y 0.5, siendo 0.5 el máximo dropout. Para mayores detalles, remítase al notebook.

Encontramos que la mejor combinación de hiperparámetros fue:

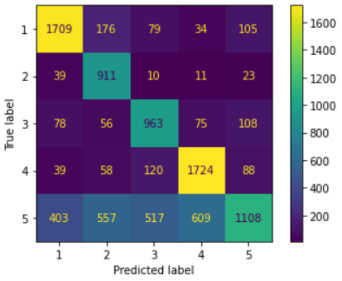
```
Mejor combinacion:
 primera_capa_adicional = False
 segunda_capa_adicional = False
 tercera_capa_adicional = False
 n_neuronas = 64
 dropout = 0.1
```

Es decir, un modelo con únicamente dos capas LSTM, cada una de 64 neuronas, y con capas de *dropout* asociadas con costo de 0.1.

Finalmente, para analizar el desempeño del modelo construimos su matriz de confusión, que da los siguientes resultados:

---Reporte para el modelo construido---

	precision	recall	f1-score	support
1	0.75	0.81	0.78	2103
2	0.52	0.92	0.66	994
3	0.57	0.75	0.65	1280
4	0.70	0.85	0.77	2029
5	0.77	0.35	0.48	3194
accuracy			0.67	9600
macro avg	0.66	0.74	0.67	9600
weighted avg	0.70	0.67	0.65	9600



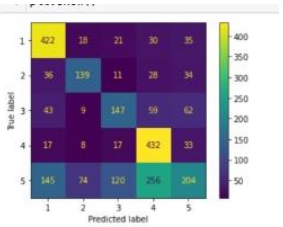
En el notebook podrá encontrar gráficas de la precisión y la función de pérdida a lo largo de las épocas. Podemos ver que, en promedio, la precisión de este modelo no es mala. De hecho, para tres de las 5 clases (neoplasias, enfermedades cardiovasculares y patologías generales, se tiene una precisión mayor o igual al 70%. Esperamos poder mejorar esto de alguna forma, para ello, intentamos otras técnicas. Asimismo, podemos ver que la clase que más se confunde es la 5: esto tiene sentido, pues las palabras que la conforman pueden caber en varias de las otras categorías. De esto, el modelo clasifica bien (sin mucho error) las otras clases. Note que las clases peor clasificadas son las minoritarias, incluso a pesar de que consideramos los pesos en la construcción del modelo.

Exportamos el archivo “model_94.h5” para utilizarlo en la etapa de validación más adelante.

4. Resultados y validación

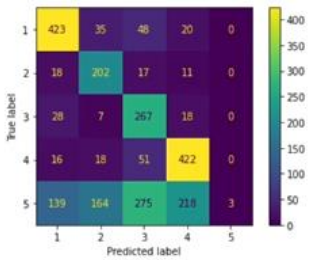
Evaluamos el desempeño del modelo con el conjunto de datos “Test” que habíamos separado al comienzo por medio de una matriz de confusión, consiguiendo los siguientes resultados:

a. Algoritmo Naive Bayes



	precision	recall	f1-score	support
1	0.64	0.80	0.71	526
2	0.56	0.56	0.56	248
3	0.47	0.46	0.46	320
4	0.54	0.85	0.66	507
5	0.55	0.26	0.35	799
accuracy			0.56	2400
macro avg	0.55	0.59	0.55	2400
weighted avg	0.56	0.56	0.53	2400

b. Algoritmo One VS Rest



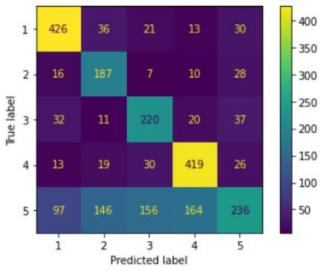
	precision	recall	f1-score	support
1	0.68	0.80	0.74	526
2	0.47	0.81	0.60	248
3	0.41	0.83	0.55	320
4	0.61	0.83	0.71	507
5	1.00	0.00	0.01	799
accuracy			0.55	2400
macro avg	0.63	0.66	0.52	2400
weighted avg	0.71	0.55	0.45	2400

c. Algoritmo LSTM

```
---Reporte para el modelo construido---
              precision    recall  f1-score   support

           1       0.73      0.81      0.77       526
           2       0.47      0.75      0.58       248
           3       0.51      0.69      0.58       320
           4       0.67      0.83      0.74       507
           5       0.66      0.30      0.41       799

    accuracy                           0.62      2400
   macro avg       0.61      0.67      0.62      2400
weighted avg       0.64      0.62      0.60      2400
```



Finalmente, estudiamos la precisión para el algoritmo LSTM. Podemos ver que el modelo clasifica muy bien las clases 1 y 4, pero las que peor clasifica son la 2 y la 3 (es decir, las minoritarias) incluso cuando consideramos pesos en la construcción del modelo. A pesar de que clasificó relativamente bien, este modelo claramente no tiene la robustez de otros, como BERT, los cuales están específicamente para el propósito de clasificación de textos.

d. Algoritmo seleccionado

A grandes rasgos, puede verse que el algoritmo con mejor precisión ponderada es el OneVsRest. No obstante, esto ocurre porque la clase 5 tiene una precisión de 1. Esto se debe a que las únicas tres instancias que clasifica como 5, efectivamente lo son. Sin embargo, esto no es tampoco un comportamiento deseable para el modelo, porque se están perdiendo muchos textos de la clase 5 en otras clases (por eso el recall es 0). Para mitigar un poco este problema, lo que hacemos es quedarnos con el siguiente mejor algoritmo en cuanto a la precisión Este es LSTM, sustentado además por el hecho de que es un algoritmo robusto. Por lo tanto, elegimos el modelo de LSTM-RNN como el de esta entrega.

e. Conclusiones

- **i.** Los textos de la categoría 5 son muy amplios, según nuestro perfilamiento. Esto significa que pueden caer en muchas de las otras clases, arruinando las métricas de precisión. Por esto, los algoritmos tienen problemas al momento de clasificarlos. Se recomienda eliminar la categoría 5 o intentar distribuirla en las demás enfermedades.
- **ii.** La precisión para la clase minoritaria (la clase 2) es mucho más baja que en los demás (47%). Por lo que se recomienda a los médicos revisar los textos que se predigan de esta clase, pues muchos estarían mal clasificados.
- **iii.** Por otro lado, aunque los resultados obtenidos no son particularmente malos, es posible mejorarlos utilizando otras técnicas de Machine Learning como transfer learning. En este caso, el mejor modelo sería usar una variación de BERT (denominada Blue BERT) que es especial para contextos médicos.

5. Descripción del trabajo en grupo

a. **Roles:**

i. **Sofía Álvarez:** Líder de datos y de analítica
ii. **Brenda Barahona:** Líder de proyecto
iii. **Alvaro Plata:** Líder de negocio

b. **Retos enfrentados en el proyecto**

- No teníamos claro cuáles algoritmos implementar para el objetivo de negocio. Después de investigar entre los algoritmos más populares y eficientes llegamos a la conclusión de implementar los 3 descritos anteriormente.
- No teníamos una referencia de cómo implementar estos algoritmos como en los laboratorios anteriores, por lo que fue necesario buscar referencias en múltiples fuentes para encontrar la manera correcta de hacer dichas implementaciones.
- Debido a la diversidad de maneras posibles de implementar un modelo (Con subsampling, con SMOTE, con CountVectorizer…) fue necesario realizar múltiples implementaciones de cada algoritmo y posteriormente escoger el mejor de cada uno, para al final tener 3 modelos, uno por cada algoritmo y evaluar cuál de estos es el más eficiente.

c. **Tareas realizadas:**

i. **Perfilamiento y preprocesamiento de los datos:** Brenda, Sofía y Alvaro – 3 horas
ii. **Implementación de cada algoritmo y construcción de modelos:** Brenda (OneVSRest), Sofía (LSTM) y Alvaro (Multinomial Naive Bayes) – 4 horas
iii. **Validación de los modelos y conclusiones:** Brenda, Sofía y Alvaro – 1 hora
iv. **Presentación y Tableros de control:** Brenda, Sofía y Alvaro – 2 horas
v. **Creación del informe:** Brenda, Sofía y Alvaro – 2.5 horas
vi. **Video:** Brenda, Sofía y Alvaro – 0.5 horas

d. **Repartición de puntos:**
Si tuviéramos que repartir 100 puntos entre los integrantes del grupo, repartiríamos 33.33 a cada uno.

e. **Puntos para mejorar en la siguiente entrega:**
i. Distribuir el trabajo en el tiempo de manera más homogénea, para no tener una gran cantidad de trabajo por hacer poco tiempo antes de la fecha de entrega.
ii. Dedicar un mayor tiempo a la comprensión de la teoría de cada algoritmo para entender su funcionamiento interno con más detalle

Referencias

[1] https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/

[2] https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

[3] Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-structure-of-the-Long-Short-Term-Memory-LSTM-neural-network-Reproduced-from-Yan_fig8_334268507 [accessed 28 Mar, 2022]

[4] Zhang Y, Chen Q, Yang Z, Lin H, Lu Z. BioWordVec, improving biomedical word embeddings with subword information and MeSH. Scientific Data. 2019.

[5] https://towardsdatascience.com/simplified-math-behind-dropout-in-deep-learning-6d50f3f47275

[6] https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

[7] https://towardsdatascience.com/7-tips-to-choose-the-best-optimizer-47bb9c1219e