Trabajo Práctico N° 3: Machine Learning

Tomás Francisco Amundarain, 108155
Universidad de Buenos Aires

Análisis de las palabras utilizadas..... 3
Análisis de los sentimientos..... 4
**Random Forest..... 7**
**Xg boost..... 7**

# Análisis Exploratorio

Primero se realizó una exploración de los datos que se disponían para realizar el trabajo con el fin de encontrar posibles features o patrones interesantes para poder detectar cuando un tweet era una noticia real y cuando no.

## Análisis de las palabras utilizadas

Se hizo un tokenizado de los textos de cada tweet y se extrajo las palabras más utilizadas tanto para los tweets que trataban de desastres naturales y para los que no.
La primera conclusión que se alcanzó es que el 66% de los tweets con target 1 tienen una url a otro sitio web mientras que el target cero el 41% lo tiene, por lo que se estableció la feature *tiene_url.*
Luego se obtuvo un top de palabras más utilizadas para ambos grupos:

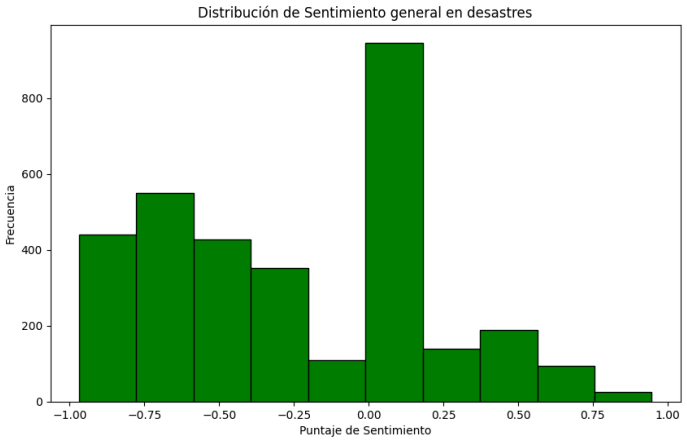
| NOTICIAS | |
|---|---|
| fire | 266 |
| kill | 155 |
| news | 135 |
| disaster | 119 |
| california | 115 |
| bomb | 113 |
| suicide | 112 |
| crash | 112 |
| year | 110 |
| police | 109 |

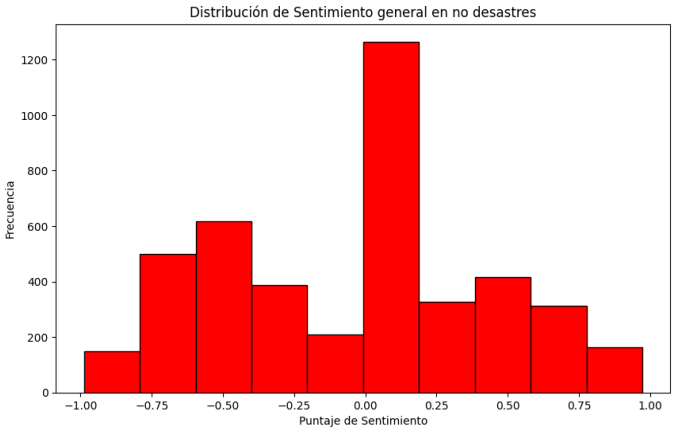
| NO NOTICIAS | |
|---|---|
| like | 294 |
| amp | 192 |
| new | 172 |
| go | 158 |
| get | 144 |
| body | 118 |
| good | 117 |
| love | 116 |
| come | 107 |
| time | 102 |

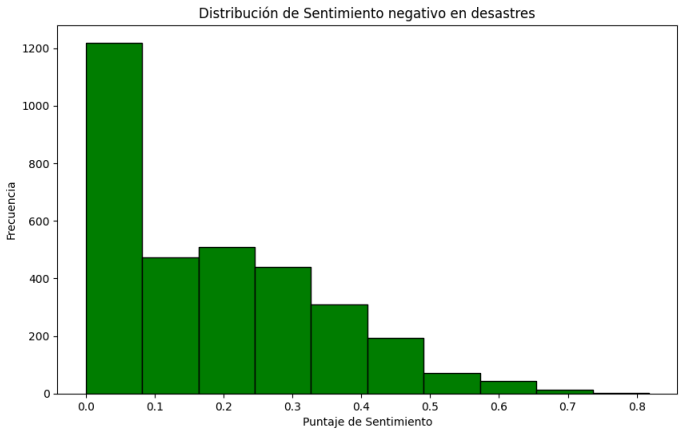
Comparando los valores de los tops de ambos grupos se observa que son totalmente distintos y hay cierta connotación negativa muy marcada en el grupo de las noticias que en el otro. Por esa misma razón se creó la feature *palabras_claves* para representar cuantas palabras del top de las noticias tiene en el texto ese tweet.
Otras features que se crearon fueron la cantidad de hashtags que tiene el tweet si bien sólo el 27% de los tweets que tratan sobre desastres tienen y del otro grupo el 20% puede servir.

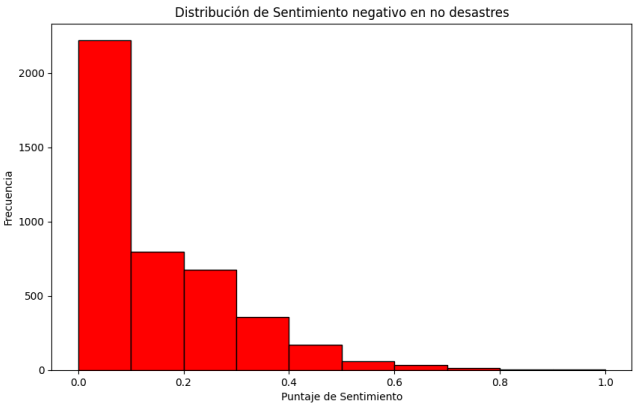
# Análisis de los sentimientos

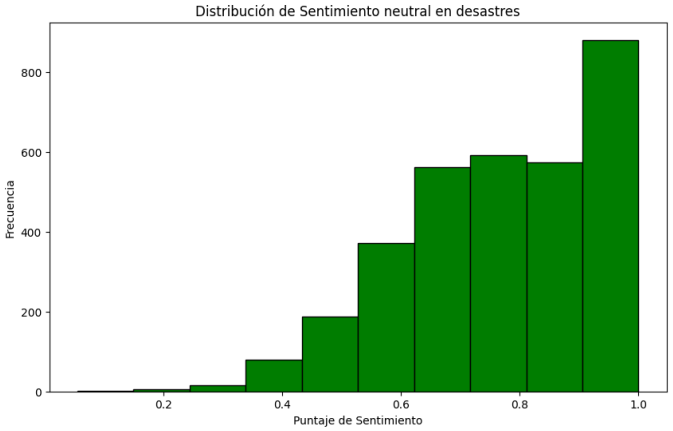
También se analizó el sentimiento del texto en el tweet, para esto se utilizaron dos modelos para clasificarlos. Uno que puntuaba numéricamente el por sentimiento cuanto tiene de cada uno y el otro modelo define qué sentimiento es el predominante (define si es positivo o negativo).
A continuación se establecieron algunos gráficos para buscar algún patrón útil.

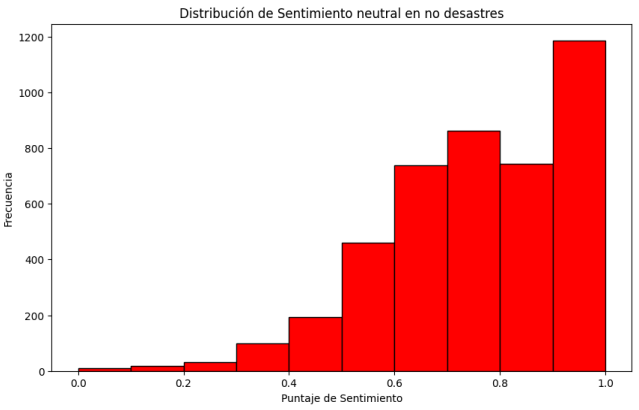


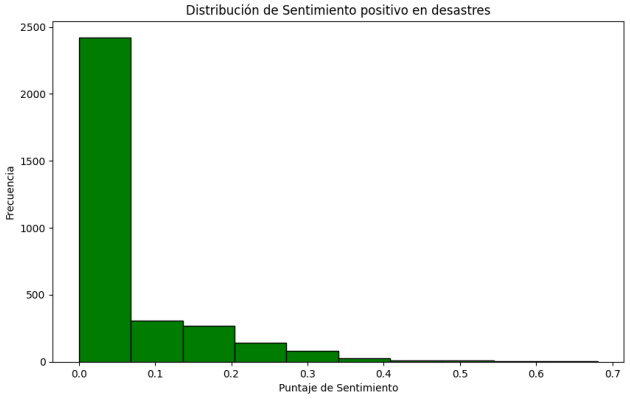



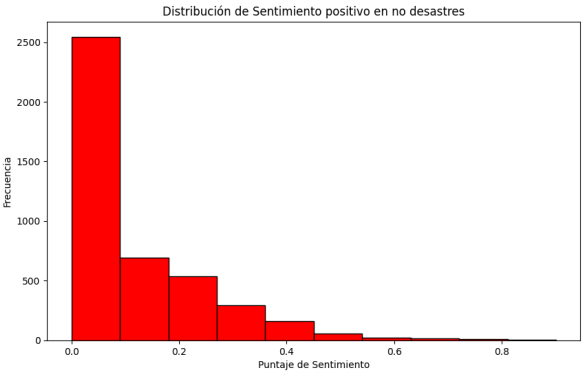



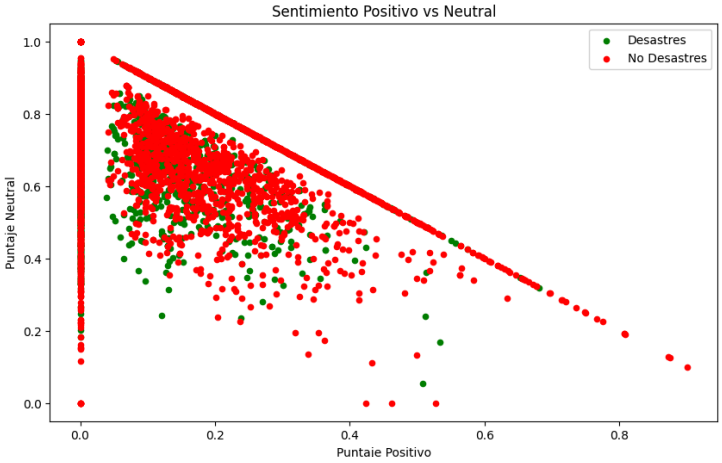



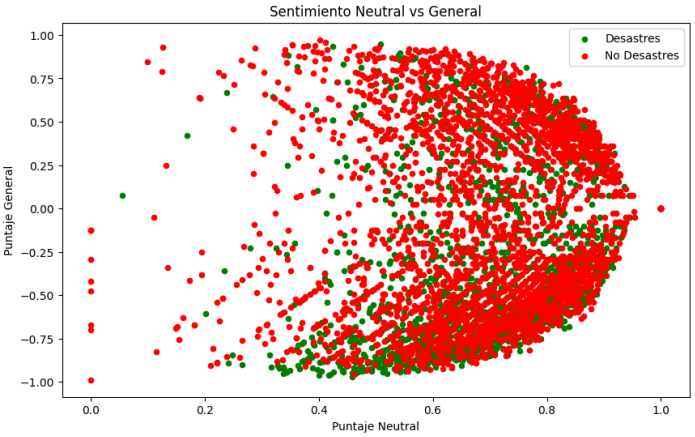



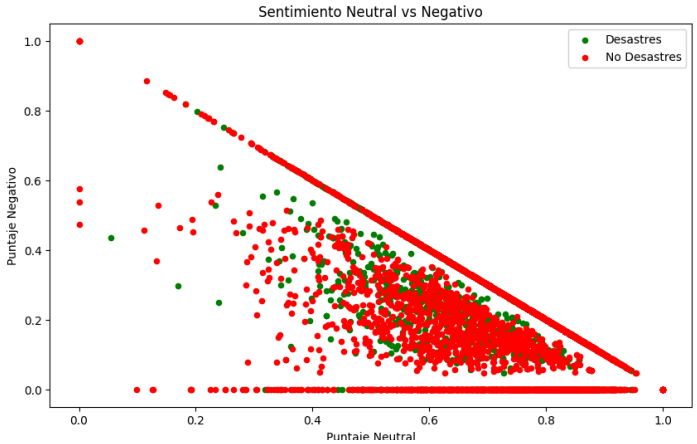



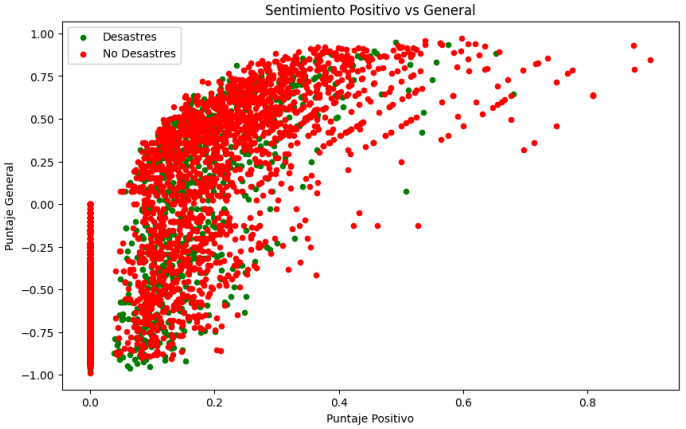
}

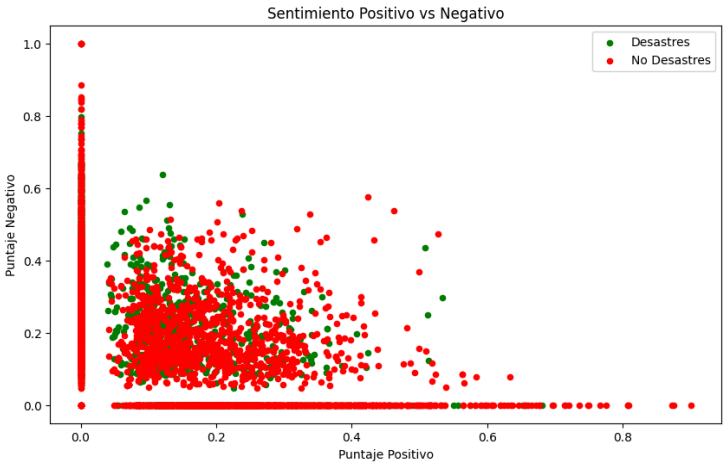
En cuanto a los gráficos por emoción se puede observar que los sentimientos correspondientes a tweets que son noticias sobre desastres naturales tienden a valores más negativos, cabe resaltar que la diferencia es leve.

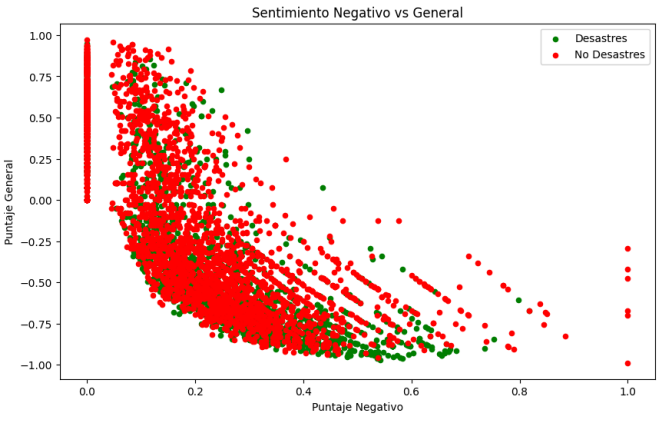


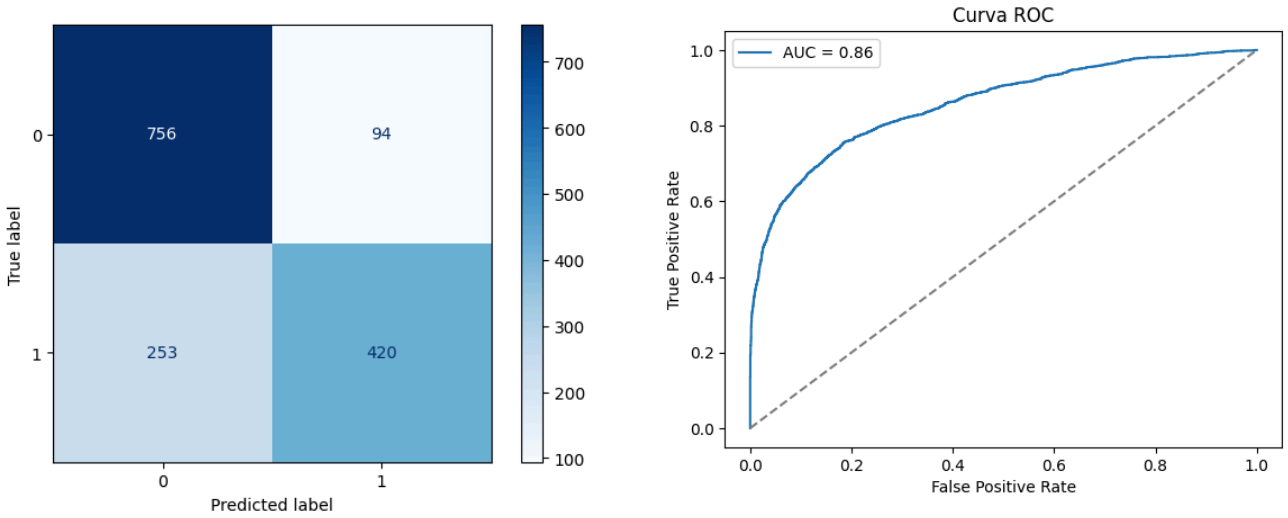



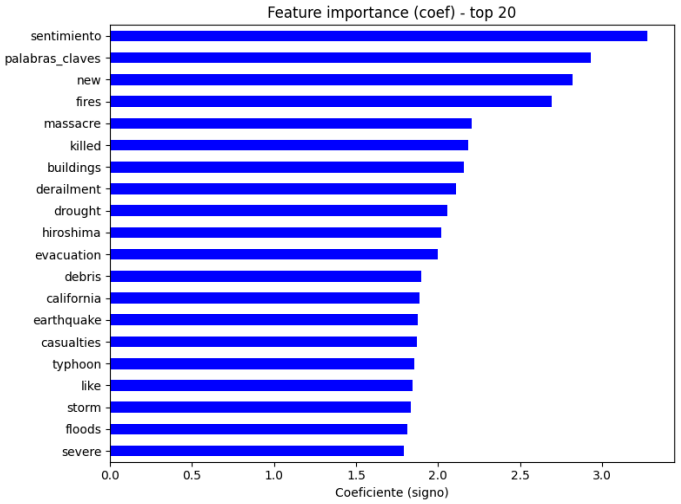




Con estos gráficos comparando las emociones y el tipo de tweet se puede confirmar aún más que el tema de las emociones no es algo que diferencie bien un grupo del otro. Lo que deja como conclusión que la clave debe estar en entender diferentes patrones que se dan en los textos para los dos grupos. En un principio además de las features mencionadas hasta el momento se va a agregar un embedding básico como lo es un bag of words.

# Machine learning baseline

Para este primer modelo se usaron como features las mencionadas anteriormente en el análisis exploratorio y como modelo una regresión logística. Para este modelo se alcanzó un puntaje máximo de 0.76555 en kaggle, dicho puntaje corresponde a la métrica f1.
Algunos gráficos importantes del modelo final:



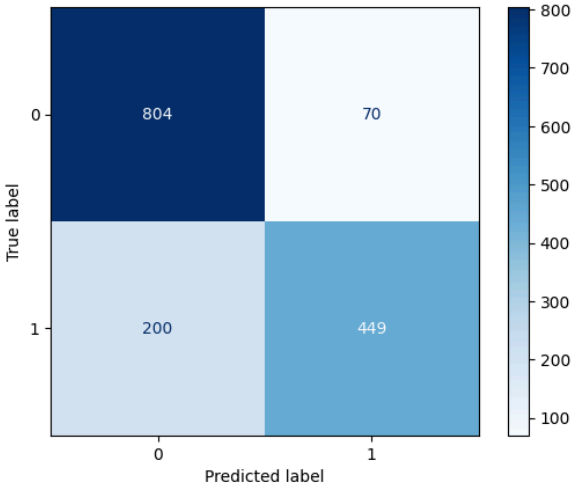

De acuerdo al feature importance, se puede observar que se le está dando mucha más importancia a las palabras que se obtuvieron en el bow que a las features encontradas. Si bien hay dos features como *sentimiento* y *palabras_claves* que son las más importantes, el gráfico del features importance más la conclusión del análisis exploratorio me afirma más que la clave está en el embedding del texto por lo que para los próximos modelos se buscará mejorar en esos puntos.

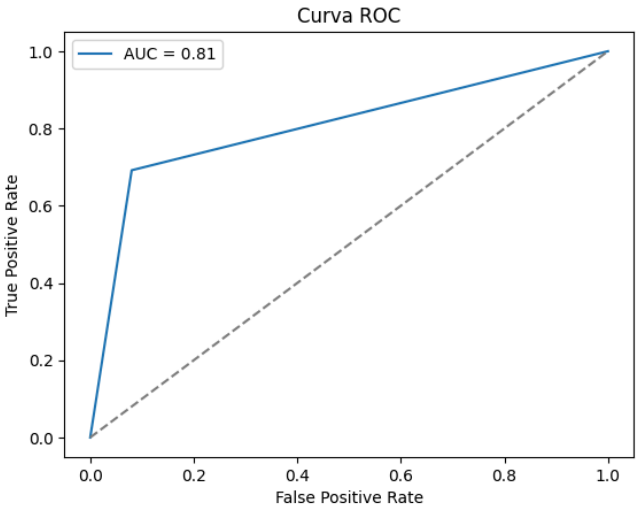
# Modelos más óptimos

Para este punto se optó por utilizar un random forest y un xg boost. Además para el embedding se utilizó un modelo pre-entrenado de la familia SBERT con el fin de obtener un resultado más preciso y poder aprender mejor sobre los patrones que existen en las noticias sobre desastres. Por lo que solo se dejaron las features categóricas y se eliminaron las demás features numéricas para ser reemplazadas por el embedding.

## Random Forest

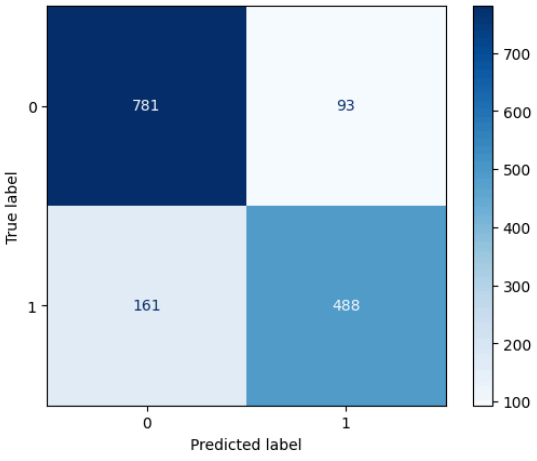
El mejor resultado que se obtuvo en kaggle fue de 0.80845, si bien dió arriba de 0.8 este tipo de modelo no es el óptimo, ya que al tener muchas features como se lo tiene con el embedding puede ser menos preciso que el modelo de xg boost.
Algunos gráficos que se obtuvieron del modelo:

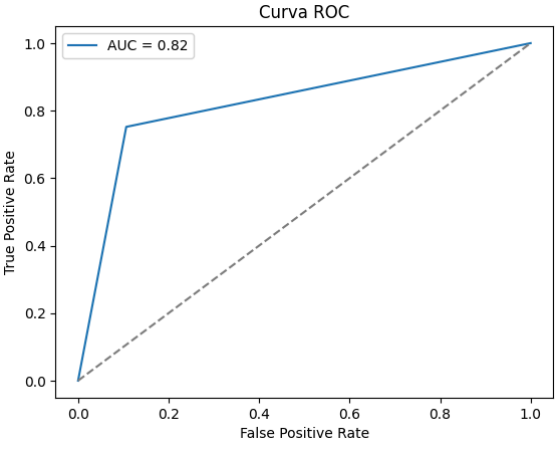



## Xg boost

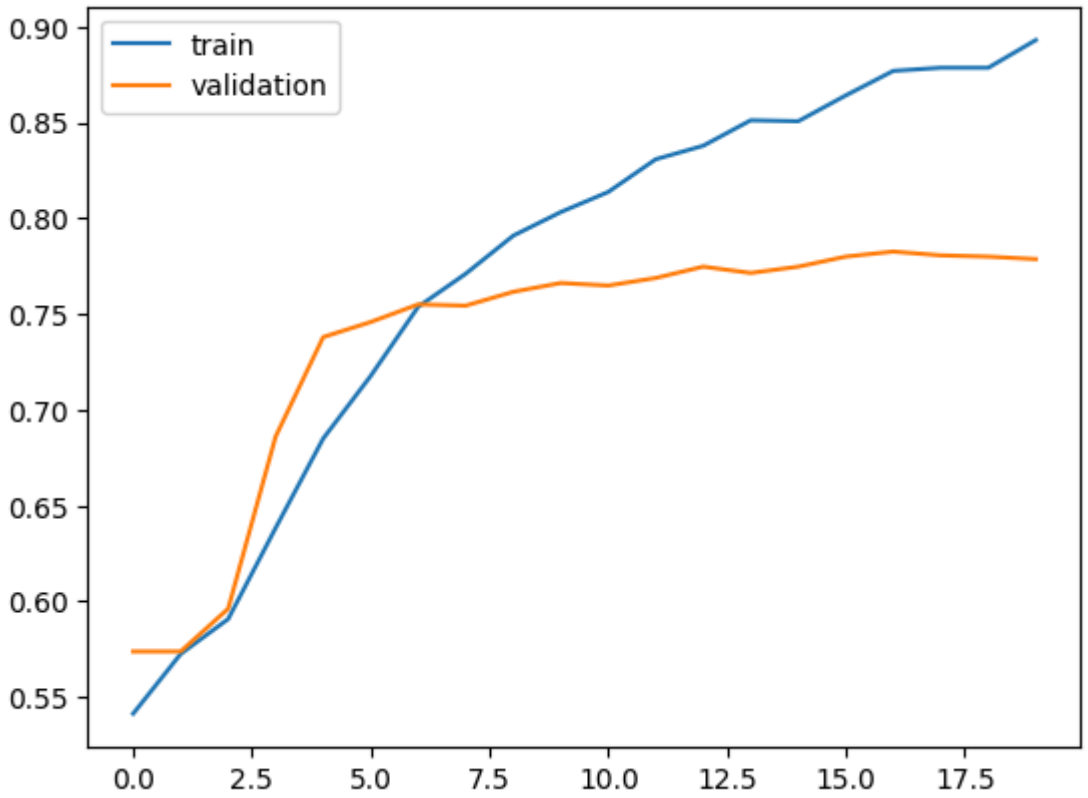
El mejor resultado que se obtuvo en kaggle fue de 0.82715, comprobando así que a este modelo le va mejor en problemas donde hay muchas variables numéricas.
Algunos gráficos que se obtuvieron del modelo:



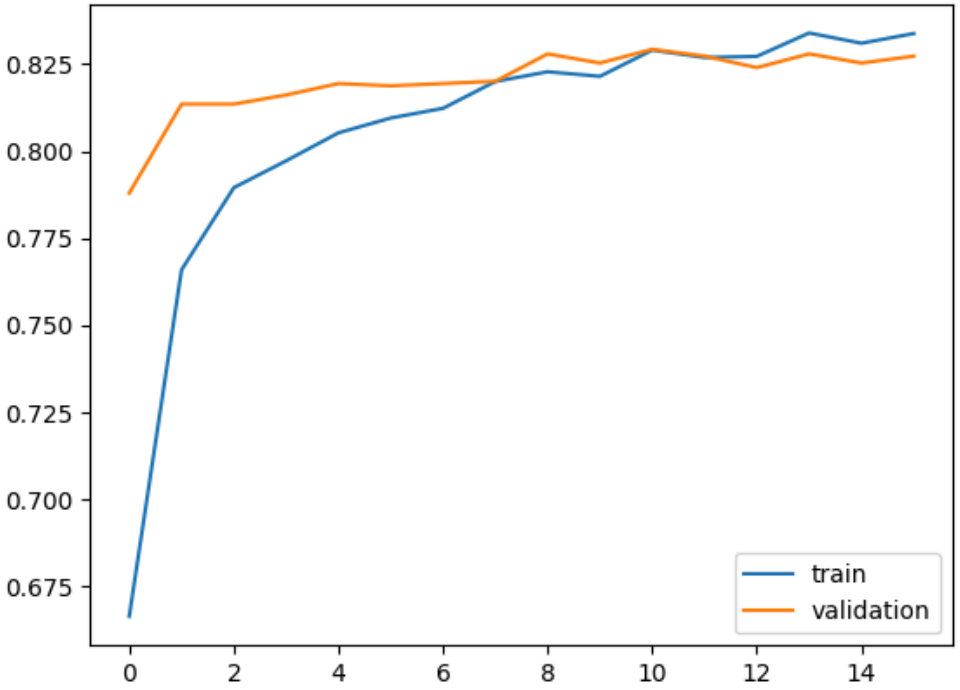



# Ejercicio Opcional

Para el punto opcional se realizó el ejercicio de la red neuronal con una capa GRU. Además se desarrollaron tres versiones diferentes de la red, siendo la última la versión final del ejercicio, pero quiero hacer algunas aclaraciones y explicar el por qué de los cambios entre versión y versión.
En la versión 1 se buscó hacer algo más simple por lo que se hizo un bow como embedding, luego una capa en donde se realizaba una convulsión de 1D, seguido por un GRU bidireccional para captar el contexto total de la palabra o patrones que la convulsión devolvía, por último una capa densa y luego el output.
En esta versión se detectó que al ser un dataset relativamente pequeño era muy fácil caer en el overfitting.




Lo que se observa es un gráfico del accuracy en cada iteración que se hacía. Se puede visualizar como van creciendo los dos accuracy, señal de que verdaderamente va aprendiendo patrones, y luego solo aumenta el accuracy del train mostrando así un poco de overfitting, ya que a partir de cierto punto el loss de validation comenzó a subir. Para evitar este problema, se agregaron dropouts en cada capa y además para la siguiente versión también se mejoró el embedding, agregando un transformer de la familia BERT, el cual no solo mejoró la calidad del embedding sino que pudo encontrar mejores patrones para aprender que junto con la capa GRU le permitió aprender también el contexto de esos patrones.




Aquí se puede observar que hasta el accuracy de validation mejoro muchisimo en la primera iteración y luego fue acompañando al de train hasta que se detuviera el entrenamiento. Para este punto se tenía un modelo bastante bueno que para el tamaño del dataset arrojaba resultados acordes, con una puntuación de 0,81642.

Investigando sobre los transformers, se encontró que existía un modelo de la familia BERT que estaba entrenado para analizar tweets por lo que para la versión final se cambió al modelo BERTWEET, el cual alcanzó una puntuación de 0.82837. Esta versión se puede mejorar pero el margen de mejora era muy poco y costaría mucho más tiempo de entrenamiento, por esa razón se dejó a esta versión como la final.

# Links

## Código

- Análisis exploratorio
- Armado de features
- Primer modelo base
- Random Forest
- Xg boost
- Red Neuronal v1
- Red Neuronal v2
- Red Neuronal v3

## Resultados

- Primer Modelo Base
- Random Forest
- Xg Boost
- Red Neuronal v3