PREDICCION DE GÉNERO DE PELÍCULAS

Tópicos Avanzados

Grupo:

Anguie Garcia - Mili Galindo - Sonia Ramírez - Lourdes Rodil

3.
Preprocesamien to del Texto

4.
Feature Ingineering

2.
Limpieza del Texto

5.
Modelación

1.
Adquisición de los datos

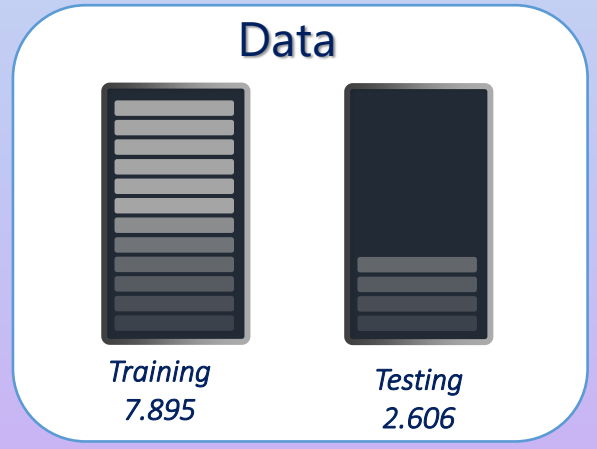
6.
Evaluación

NLP Pipeline

METODOLOGIA DEL PROYECTO

Objetivo del proyecto

Clasificar el género cinematográfico de una película en función de su argumento (plot)

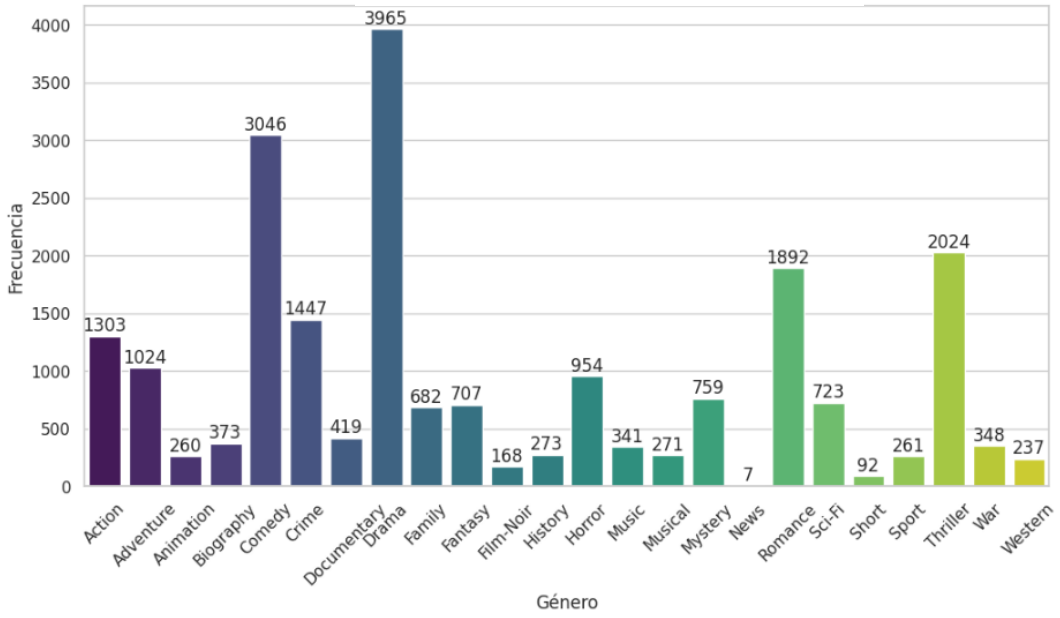


1. Adquisición de los datos & EDA

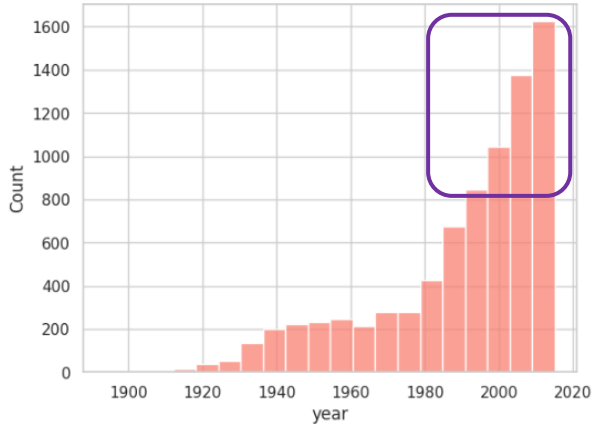
Campos dataset:

- Titulo
- Año estreno
- Rating
- Argumento /plot
- Géneros

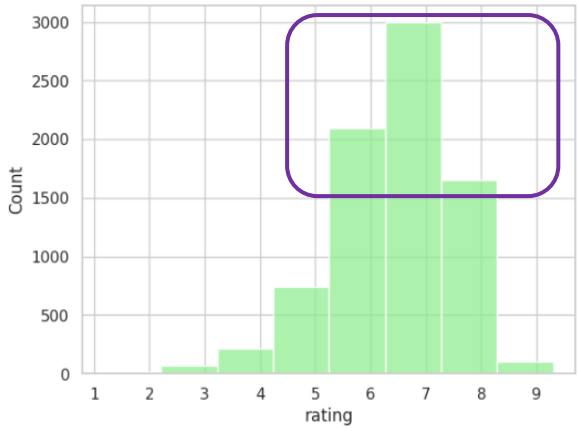
Histograma por genero de película



Histograma de años de películas



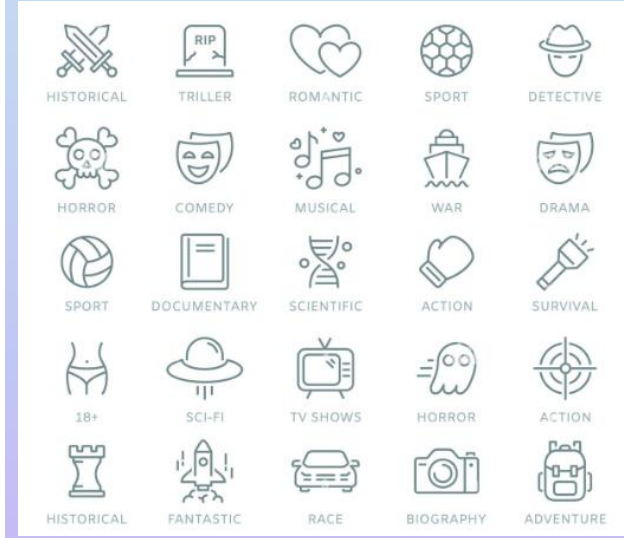
Histograma de rating de películas

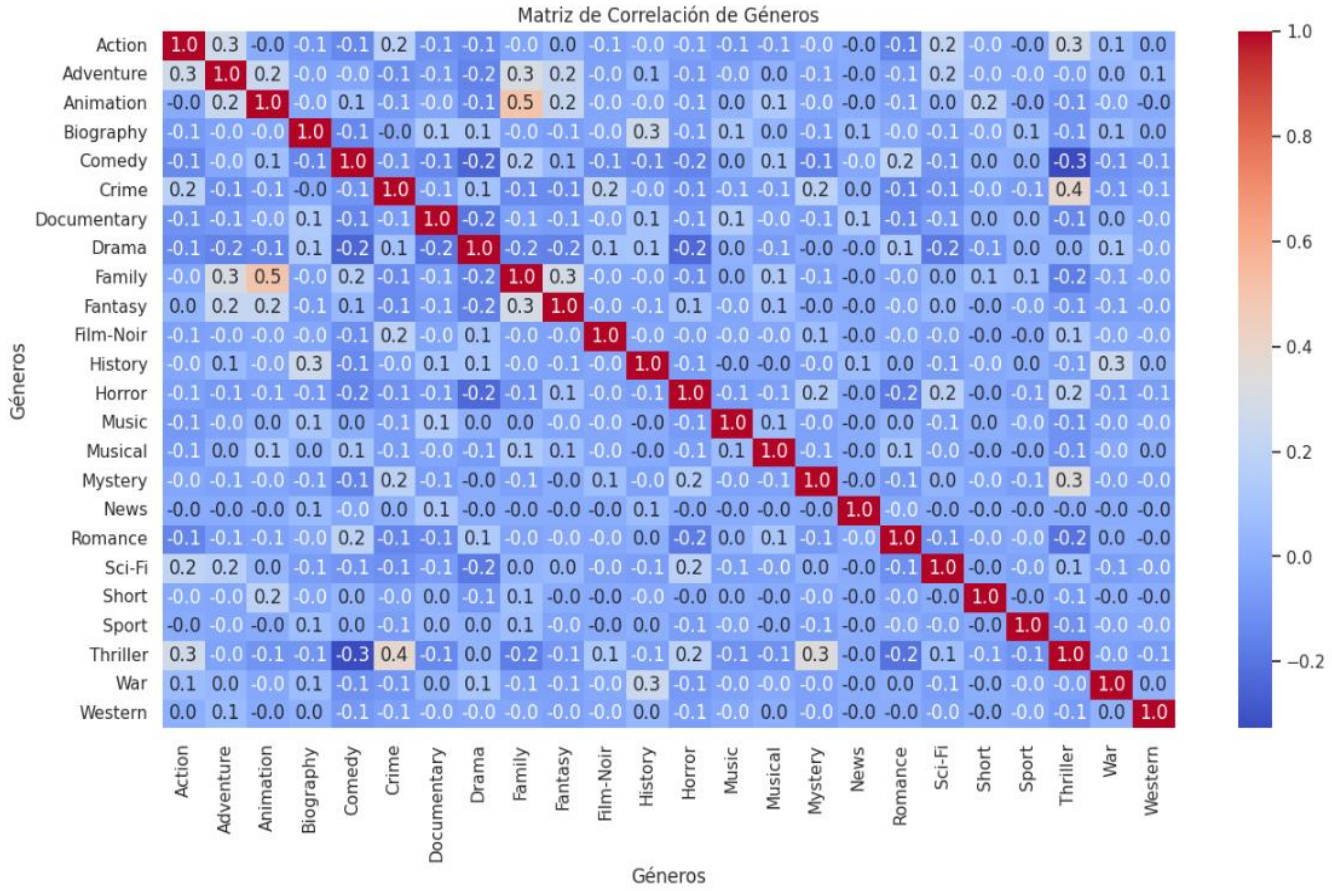


1. Adquisión de los datos & EDA

Mayor correlación positiva es entre “Familia” y “Animation” con un 0.5 y la mayor correlación negativa es entre “Triller” y “Comedy” con un -0.3 :

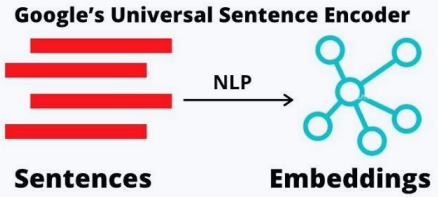
Matriz de Correlación





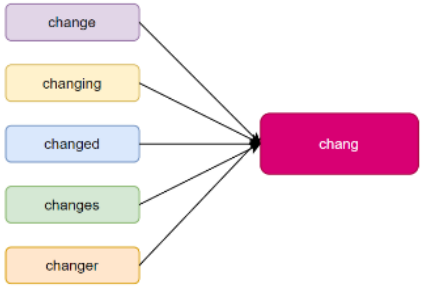
Limpieza: Para el proyecto no fueron requeridas labores de limpieza de la base de datos, previo al paso de procesamiento, ya que no se evidenciaron datos nulos

3. Preprocesamiento del Texto & 4. Feature Ingineering



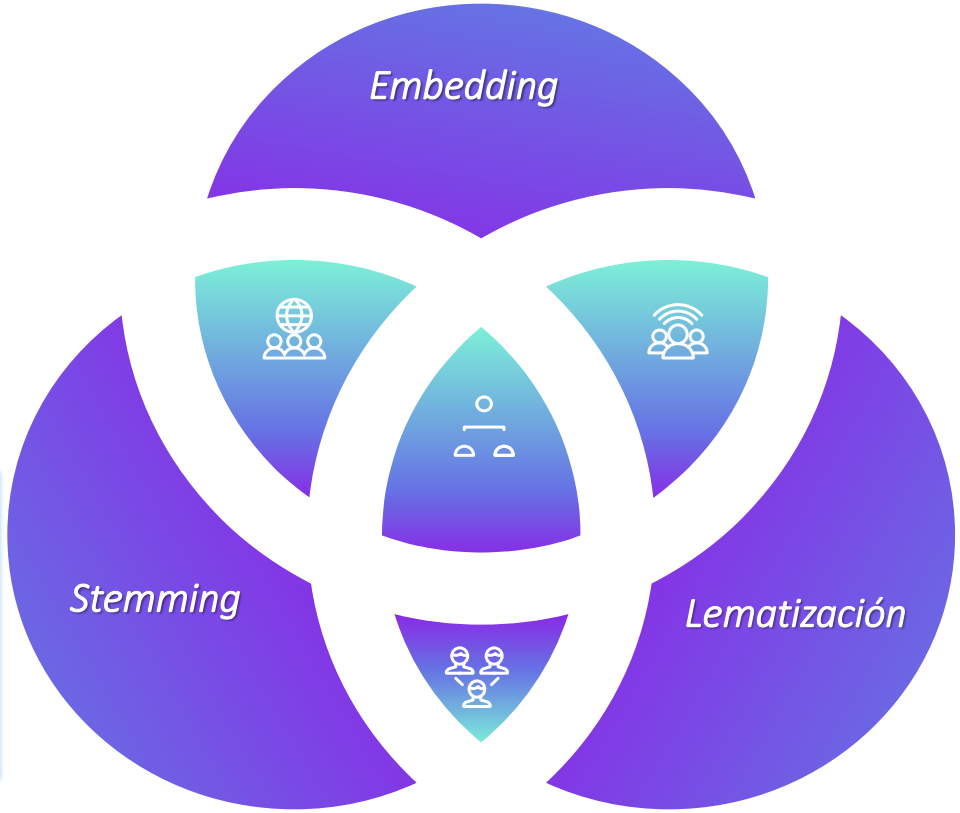

```
import tensorflow as tf
import tensorflow_hub as hub
```

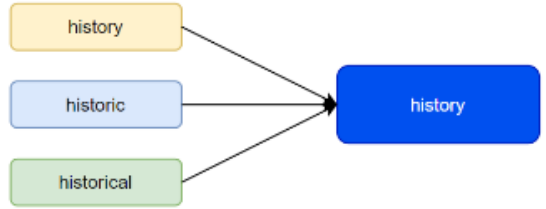
```
module_url = 'https://tfhub.dev/google/universal-sentence-encoder/4'
use_embed = hub.load(module_url)
```




```
stemmer = SnowballStemmer('english')
import re

def tokenize_and_stem(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z0-9]', ' ', text)
    text = re.sub(r'[^\w\s]', '', text)
    text = ' '.join(text.split())
    tokens = [word for word in nltk.word_tokenize(text)]
    stems = [stemmer.stem(token) for token in tokens]
    return stems
```

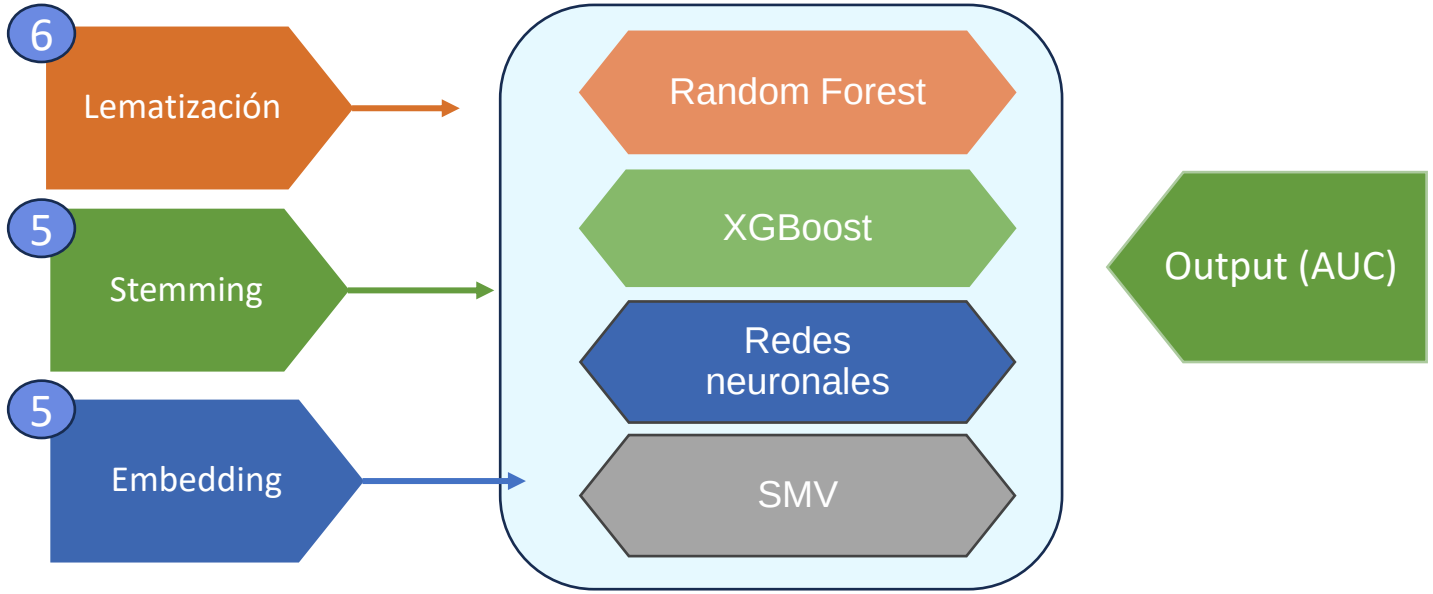





```
lemmatizer = WordNetLemmatizer()
# define a function that accepts text and returns a list of lemmas
def split_into_lemmas(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z0-9]', ' ', text)
    text = re.sub(r'[^\w\s]', '', text)
    words = [word for word in nltk.word_tokenize(text)]
    return [lemmatizer.lemmatize(word) for word in words]
```

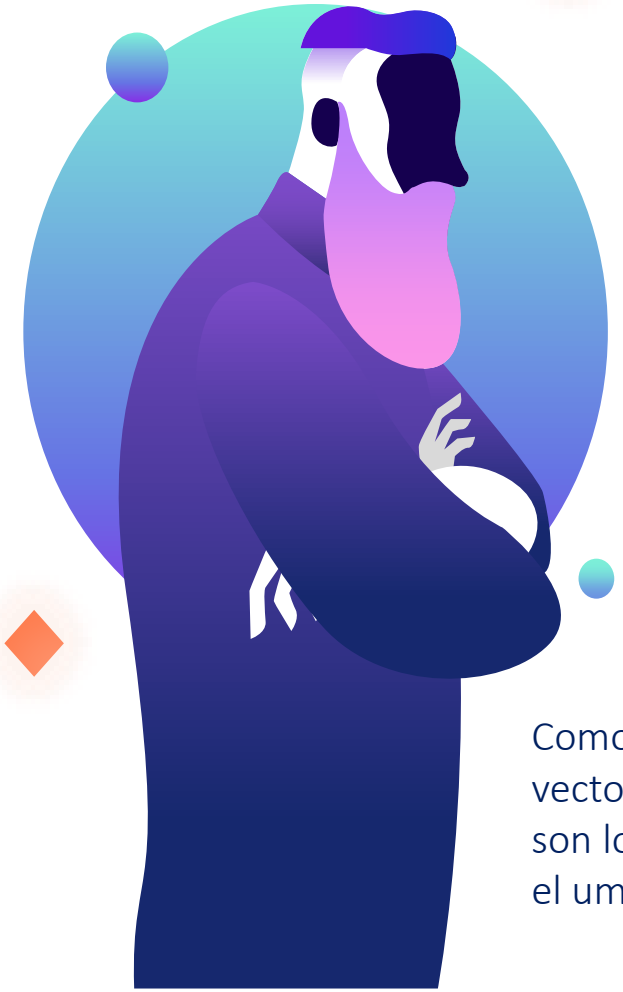
5. Modelación

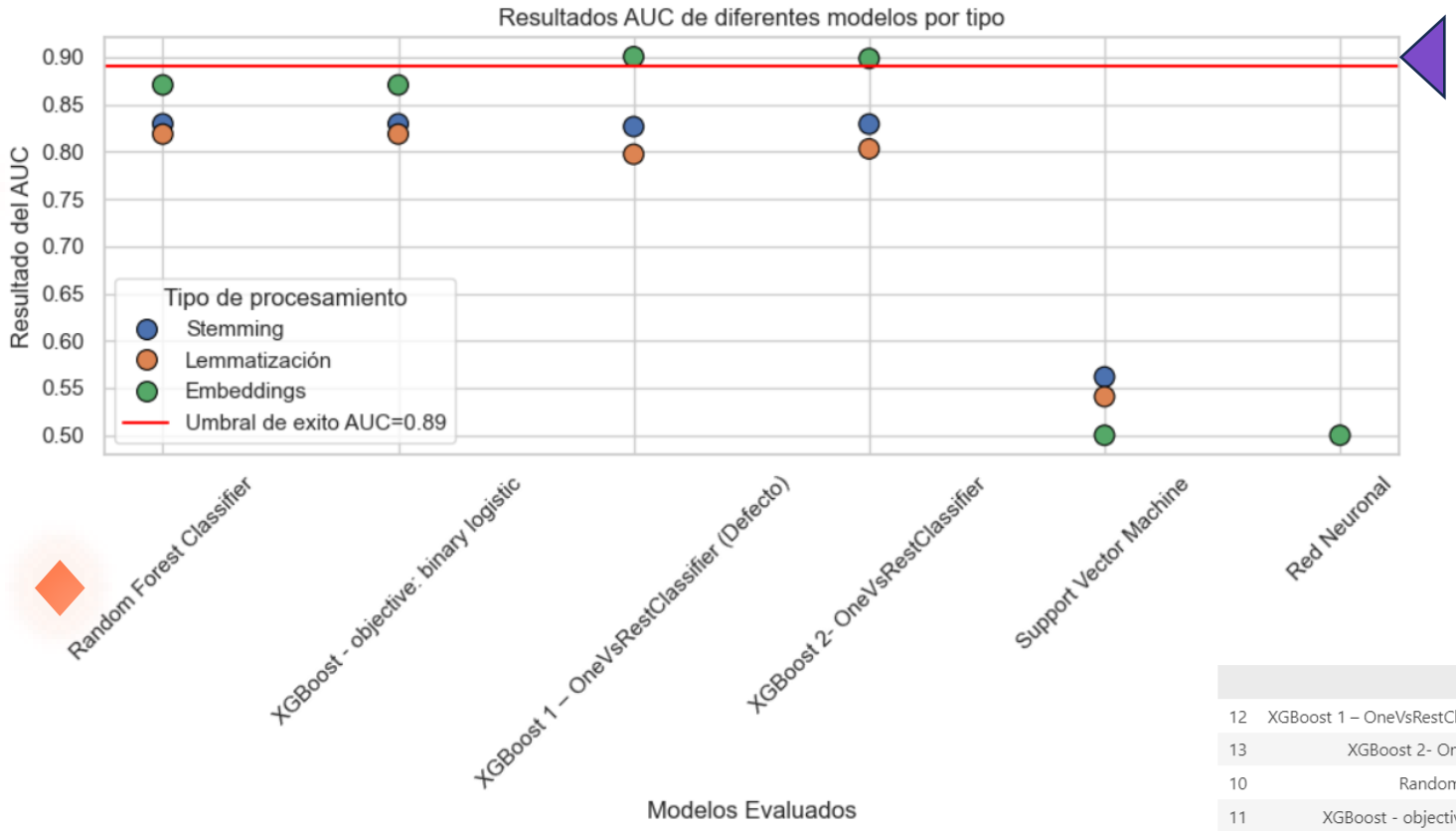
A continuación, se presenta un resumen de los modelos diseñados:



Nombre del Modelo	Parametros
Random Forrest Classifier	n_jobs=-1, n_estimators=150, max_depth=15
XGBoost - (objective: binary logistic)	objective='binary:logistic'
XGBoost - OneVsRestClassifier	Parametros por defecto: n_estimators=100,learning_rate=0.1, max_depth=3
XGBoost -(OneVsRestClassifier)	n_jobs=-1, n_estimators=100,learning_rate=0.3, max_depth=5
Support Vector Machine	Parametros por defecto
Red Neuronal	Parametros por defecto

6. Evaluación





Como se observa los modelos XGBoost vectorizados con el embedding de Tensorflow, son los que presentan el mejor desempeño sobre el umbral de éxito.

	Modelo	AUC	Tipo
12	XGBoost 1 – OneVsRestClassifier (Defecto)	0.9005	Embeddings
13	XGBoost 2- OneVsRestClassifier	0.8986	Embeddings
10	Random Forest Classifier	0.8706	Embeddings
11	XGBoost - objective: binary logistic	0.8706	Embeddings
0	Random Forest Classifier	0.8295	Stemming
1	XGBoost - objective: binary logistic	0.8295	Stemming
3	XGBoost 2- OneVsRestClassifier	0.8292	Stemming
2	XGBoost 1 – OneVsRestClassifier (Defecto)	0.8265	Stemming
5	Random Forest Classifier	0.8186	Lemmatización
6	XGBoost - objective: binary logistic	0.8186	Lemmatización
8	XGBoost 2- OneVsRestClassifier	0.8030	Lemmatización
7	XGBoost 1 – OneVsRestClassifier (Defecto)	0.7973	Lemmatización
4	Support Vector Machine	0.5618	Stemming
9	Support Vector Machine	0.5409	Lemmatización
14	Red Neuronal	0.5000	Embeddings
15	Support Vector Machine	0.5000	Embeddings

Gracias