CLASIFICACIÓN DE SEÑALES CAPTADAS DESDE UNA TRANSMISIÓN STREAMING EN TIEMPO REAL

Por: Enzo Camargo, Max Carrasco, Miguel Ruiz, Jhon Vargas

Facultad de Ciencias, Universidad Nacional de Ingeniería, Lima, Perú

Lima, 23 de marzo de 2021

RESUMEN

El presente informe describe un proyecto para clasificar las señales de audio captadas desde un emisor en tiempo real. Para realizar el procesamiento de la señal se obtuvo un archivo de aproximadamente ocho horas de duración, el cual fue dividido en intervalos de 3 segundos de duración, permitiendo obtener las características de los anuncios(slogans) seleccionados. Una parte de estos archivos serán utilizados como data de entrenamiento del modelo seleccionado y la otra parte para su validación. Se han seleccionado 28 atributos inicialmente mediante el uso del espectrograma de la señal. Asimismo se hace el modelamiento de una red neuronal de clasificación para etiquetas multistate usando el algoritmo MLPClassifier y aplicando la función de activación de salida softmax en la última capa.

Palabras clave: slogans, red neuronal, dataset, modelo, MLPClassifier, softmax, normalizacion.

I. INTRODUCCIÓN

La radio está evolucionando en un ecosistema de medios digitales cambiante. El audio a demanda ha dado forma al panorama de grandes datos de audio no estructurados disponibles en línea. En este trabajo, se utiliza la extracción para mejorar la capacidad de descubrimiento y el enriquecimiento del contenido proporcionado. Se desarrolla una aplicación web para la producción y transmisión de radio en vivo. La aplicación ofrece una funcionalidad típica de transmisión y mezcla en vivo, mientras realiza anotaciones en tiempo real como un proceso en segundo plano al registrar las señales identificadas. Para las necesidades de una estación de radio típica, se entrena un modelo de clasificación de avisos comerciales. El modelo se basa en una arquitectura de red neuronal. Los resultados se consideran alentadores en cuanto a la aplicabilidad de la metodología propuesta.

II. EL PROBLEMA

El problema principal consiste en determinar e identificar los slogans que se emiten en cualquier momento durante la emisión de un programa de radio. Este problema se subdivide en los siguientes puntos:

1. El problema de determinar qué intervalo de tiempo es el necesario para poder identificar los slogans.
2. El etiquetado de los audios. La duración total es de 8 horas, por lo que este problema es uno de los más pesados, pues es manual.
3. La preparación del dataset, conlleva en preparar la data cruda en información que pueda ser leído por el algoritmo. El paso de audio a formato numérico, el balanceo de datos, la limpieza, la normalización para un mejor manejo.
4. Escoger el algoritmo de clasificación con el número de capas y neuronas adecuadas.

III. METODOLOGÍA DEL TRABAJO

El procedimiento de trabajo que hemos propuesto para resolver el problema está dividido en 3 pasos. En la Fig 1., resumimos los pasos:

1. Creación de Dataset: Consiste en la generación del audio localmente, segmentación y etiquetado. También está asociado en la conversión de la señal de audio a información numérica que sea legible por el modelo, generación de atributos y la preparación de los datos.
2. Modelamiento y Entrenamiento: Está relacionado al modelo de clasificación multi-state, cantidad de capas e iteraciones. También consiste en encontrar el mejor accuracy y el menor error de entrenamiento
3. Evaluación del modelo: Se evaluará el modelo de manera local, luego se llevará a un servidor en la nube para ser evaluado en real time.

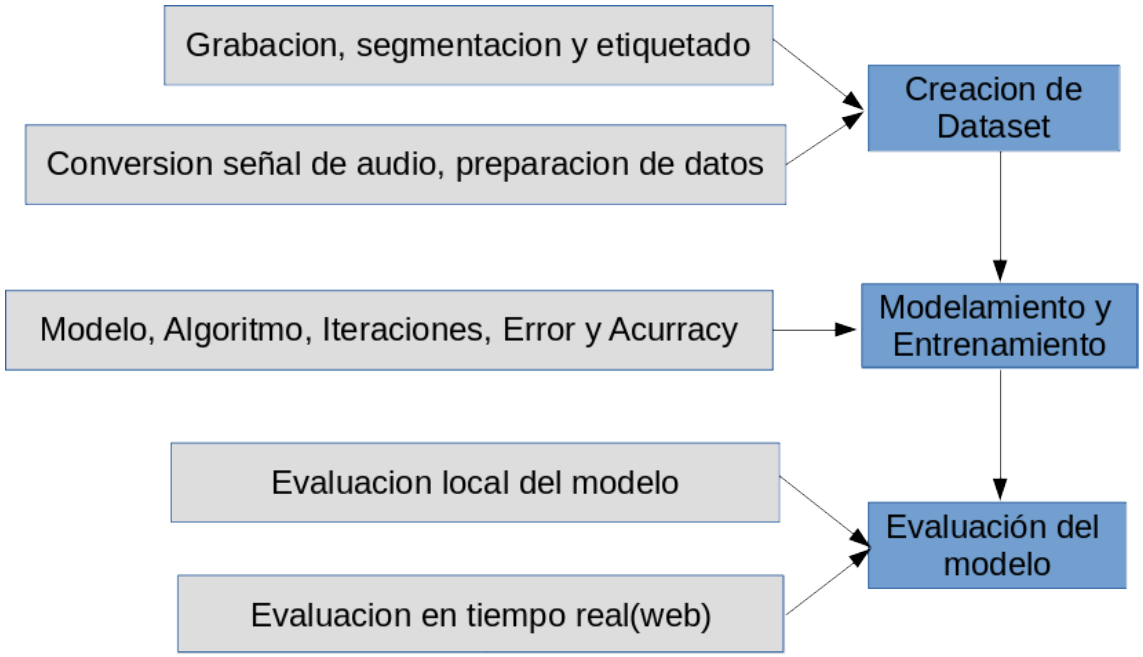


Fig 1.

IV. DESARROLLO DE LA SOLUCIÓN

Para el desarrollo de la solución hemos diseñado las siguientes etapas:

1. CREACIÓN DEL DATASET

El desarrollo de esta etapa tiene el siguiente flujo. En la Fig 2., podemos ver las siguientes partes:

- Entrada: El streaming RPP que usaremos cómo foco de recolección de datos
- Salida: 1ra salida sería el audio en crudo(Raw audio en mp3) y cómo 2da salida tenemos el dataset generado por las diversas herramientas de python.
- Herramientas: Se usó una Laptop ASUS 8G RAM, OS Linux/Debian. Se usaron cómo herramientas al reproductor VLC, ffmpeg del sistema operativo, python con modulos librosa, audiosegment, sklearn, entre otros.

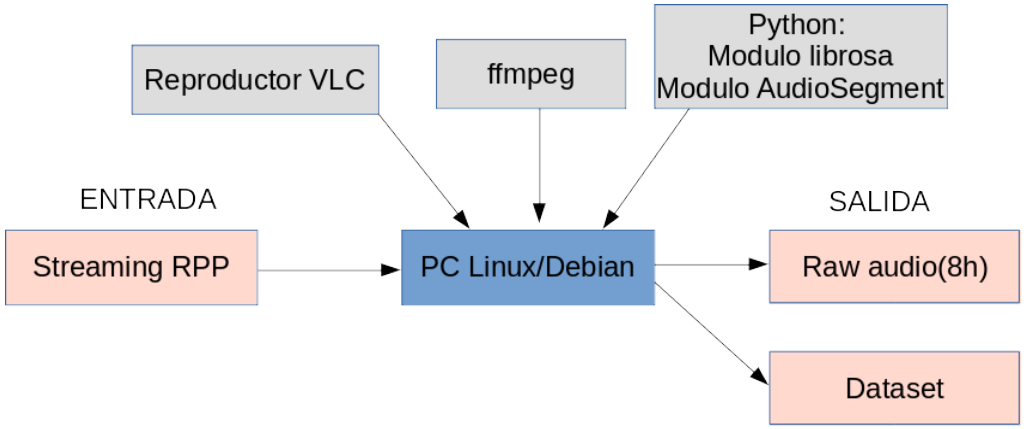


Fig 2.

Cómo primer paso se definen los slogans a analizar. En la siguiente tabla Tab 1. Mostramos los 10 slogans:

SLOGAN ID	MARCA RELACIONADA	ETIQUETA
slogan1	Aceite beltran	1
slogan2	Ajinomoto	2
slogan3	Donofrio	3
slogan4	Metro	4
slogan5	Verisure	5
slogan6	Tiyapuy	6
slogan7	Totus	7
slogan8	Pantene	8
slogan9	Upc	9
slogan10	Olspace	10
otros	Otros	0

Tab 1.

La fuente de streaming utilizada: https://18493.live.streamtheworld.com/RADIO_RPP.mp3

A partir del link proporcionado se obtendrá un archivo de audio de 8 horas de duración usando VLC cómo reproductor streaming y grabación de audio.

Para la segmentación, se utilizó los siguientes criterios de partición:

- La programación streaming de RPP tiene una estructura repetitiva de 10min(entrevistas) y 5min(anuncios).
- Se reconocen duración de slogans de 10seg, 15seg, 20seg, 30seg.
- Sí los anuncios están reproducidos de manera continua.

Según lo anterior, cómo el streaming tiene una estructura repetitiva, primero se partieron por grupos de 15min de audio. Luego de cada grupo de 15min, se decidió usar particiones(ventanas) de 3seg tal que puedan cubrir todos los tipos de slogans y que también se genere una cantidad adecuada de datos en cada grupo de 15min.

Luego se usó comandos ffmpeg para poder particionar los audios en 3seg:

- ffmpeg -i audio002_000.mp3 -f segment -segment_time 3 -c copy audio002_000_%03d_otros.mp3
- ffmpeg -i audio002_001.mp3 -f segment -segment_time 3 -c copy audio002_001_%03d_otros.mp3
- ffmpeg -i audio002_002.mp3 -f segment -segment_time 3 -c copy audio002_002_%03d_otros.mp3

Para la conversión, primero pasamos el audio en mp3 a formato wav usando el módulo AudioSegment. Este formato es el que utiliza el módulo librosa en python que lo utilizamos para poder generar espectrogramas a partir de las señales de audio. En la Fig 3., podemos ver el código utilizado:

```
############################################################
################### Preparacion de Datos  ##################

# Crear expectogramas a partir de los audios.
# 1. Convertimos los audios de mp3 a wav.
# 2. Convertir los audios.wav en png para poder crear los atributos.

# Definimos el formato de la imagen
cmap = plt.get_cmap('inferno')
plt.figure(figsize=(8,8))

# Creamos la lista de slogans
slogan = 'slogan1 slogan2 slogan3 slogan3 slogan4 slogan5 slogan6 slogan7 slogan8 slogan9 otros'.split()

# ./TestData es la ruta donde esta todos los audios etiquetados Audio00x_xyz_sloganw.
path = r'./RawData'
extension = '.mp3'


for root, dirs_list, files_list in os.walk(path):
    for file_name in files_list:
        if os.path.splitext(file_name)[-1] == extension:
            file_name_path = os.path.join(root, file_name)
            name=os.path.splitext(file_name)[0]
            sound = AudioSegment.from_mp3(file_name_path)
            sound.export(f'{file_name_path[:-3]}.wav', format="wav")
            audio = f'{file_name_path[:-3]}.wav'
            y, sr = librosa.load(audio, mono=True, duration=5)
            plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, cmap=cmap, sides='default', mode='default', sc
            plt.axis('off');
            plt.savefig(f'{file_name_path[:-3]}png')
            plt.clf()

```

Fig 3.

Los espectrogramas nos servirán para poder extraer las siguientes características de la señal y de este modelo la podemos usar para crear los atributos del dataset. En la Fig 4., podemos ver un ejemplo de un espectrograma de una señal de audio.

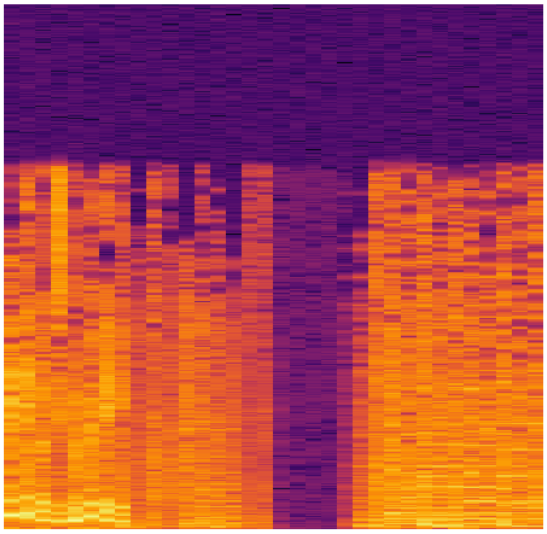


Fig 4.

Para crear los atributos, usamos el módulo librosa en python mediante el cual pudimos crear los siguientes atributos:

- Coeficientes de frecuencia Cepstral MEL (MFCC)
- Centroide espectral
- Tasa de cruce por cero
- Frecuencias cromáticas
- Caída espectral.

En la Fig 5., podemos ver el código de esta creación de atributos usando el módulo librosa:

```
44
45 # Extraemos las caracteristicas desde el Espectograma que se convertiran en atributos del dataset:
46 # 1. Mel-frequency cepstral coefficients (MFCC)
47 # 2. Spectral Centroid
48 # 3. Zero Crossing Rate
49 # 4. Chroma Frequencies
50 # 5. Spectral Roll-off.
51 extension = '.wav'
52 for root, dirs_list, files_list in os.walk(path):
53     for file_name in files_list:
54         if os.path.splitext(file_name)[-1] == extension:
55             file_name_path = os.path.join(root, file_name)
56             audio = file_name_path
57             y, sr = librosa.load(audio, mono=True, duration=30)
58             rmse = librosa.feature.rms(y=y)
59             chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
60             spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
61             spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
62             rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
63             zcr = librosa.feature.zero_crossing_rate(y)
64             mfcc = librosa.feature.mfcc(y=y, sr=sr)
65             p_slogan = identificar_parametros(file_name)
66             to_append = f'{file_name_path} {np.mean(chroma_stft)} {np.mean(rmse)} {np.mean(spec_cent)} {np.mean
67             for e in mfcc:
68                 to_append += f' {np.mean(e)}'
69             to_append += f' {p_slogan}'
70             file = open('dataset.csv', 'a', newline='')
71             with file:
72                 writer = csv.writer(file)
73                 writer.writerow(to_append.split())
74
75 # Cargamos y leemos los datos en la variable pandas data.
76 # Contiene las etiquetas y atributos.
77 data = pd.read_csv('dataset.csv')
78 data.head(10)
79
```

Fig 5.

Y con esto generamos el primer dataset (draft) que afinaremos conforme avancemos con los pasos de preparación y entendimiento de datos. En la Fig 6., vemos el comando “data.head(10)”, que nos muestra los diez primeras columnas del dataset.

	filename	chroma_stft	rmse	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1	mf
0	./RawData/audio001_006_125_otros..wav	0.387608	0.107008	1317.754772	1159.454356	2470.559479	0.095926	-241.768204	176.564
1	./RawData/audio001_006_166_otros..wav	0.356498	0.092106	1043.845881	881.089241	1884.407616	0.069344	-287.777222	179.974
2	./RawData/audio001_005_031_otros..wav	0.246486	0.179542	992.162413	791.586116	1866.659546	0.061218	-274.512939	158.679
3	./RawData/audio001_006_100_otros..wav	0.376072	0.078880	1288.575440	1003.262790	2220.256859	0.101771	-289.792877	177.994
4	./RawData/audio001_006_177_otros..wav	0.419918	0.071038	1766.992766	1637.920561	3540.959984	0.115303	-197.227325	143.355
5	./RawData/audio001_007_133_otros..wav	0.325838	0.126466	1550.898981	1528.945827	3450.403684	0.076562	-219.222626	138.284
6	./RawData/audio001_005_233_otros..wav	0.346125	0.164938	1896.240906	1642.376334	3531.779161	0.110685	-119.659988	131.901
7	./RawData/audio001_008_112_otros..wav	0.352584	0.080757	991.182967	795.214504	1747.027162	0.079491	-321.647705	185.630
8	./RawData/audio001_004_148_otros..wav	0.293553	0.100916	919.432505	918.736512	1707.633085	0.061224	-316.664307	163.455
9	./RawData/audio001_004_262_otros..wav	0.264183	0.185721	842.982547	664.467078	1384.601784	0.058426	-270.027100	191.468

10 rows × 28 columns

Fig 6.

Para la preparación y entendimiento de los datos, ejecutaremos los siguientes pasos:

1. Eliminacion de NULLs or NA

Primero validamos si existen datos NULL o NA. En la Fig 7., podemos ver el código usado. Para este fin usaremos los comandos “data.isnull().valuesany()” y “data.isna().sum().sum()”. Segun vemos no existen datos NULL o NA.

```
#print(data.describe())
# Verificamos si existen datos Null, NA en el dataset.
print("Datos Null y NA en el dataset")
print("============================ \n")
print(data.isnull().sum())
print("\n")
print("Datos Null en todo el dataset: ", data.isnull().values.any())
print("Datos NA en todo el dataset: ", data.isna().sum().sum())

```

```
Datos Null y NA en el dataset
============================

filename              0
chroma_stft           0
rmse                  0
spectral_centroid     0
spectral_bandwidth    0
rolloff               0
zero_crossing_rate    0
mfcc1                 0
mfcc2                 0
mfcc3                 0
mfcc4                 0
mfcc5                 0
mfcc6                 0
mfcc7                 0
mfcc8                 0
mfcc9                 0
mfcc10                0
mfcc11                0
mfcc12                0
mfcc13                0
mfcc14                0
mfcc15                0
mfcc16                0
mfcc17                0
mfcc18                0
mfcc19                0
mfcc20                0
label                 0
dtype: int64


Datos Null en todo el dataset:  False
Datos NA en todo el dataset:  0
```

Fig 7.

2. Eliminación de Outliers o Datos extremos

Para eliminar los datos extremos, analizaremos la asimetria de los datos. Para esto usaremos el codigo en python “data.skew()” donde podemos ver en la Fig 8. que algunos datos tienen alguno aplanamiento(skew>0 y skew>0).

```
data = pd.read_csv('dataset.csv')

# Para verificar la asimetria de los datos
print("Asimetria de los datos:\n", data.skew())

```

```
Asimetria de los datos:
 chroma_stft           0.733111
rmse                   0.256574
spectral_centroid      0.119376
spectral_bandwidth    -0.456112
rolloff               -0.056841
zero_crossing_rate     0.659702
mfcc1                  0.014606
mfcc2                  0.249412
mfcc3                 -0.211700
mfcc4                 -0.475015
mfcc5                  0.538091
mfcc6                 -0.127020
mfcc7                 -0.492690
mfcc8                  0.515021
mfcc9                 -0.294775
mfcc10                 0.137468
mfcc11                 0.377331
mfcc12                -0.152003
mfcc13                -0.180686
mfcc14                -0.171644
mfcc15                 0.124040
mfcc16                -0.518783
mfcc17                -0.166452
mfcc18                -0.009480
mfcc19                -0.297108
mfcc20                -0.733120
dtype: float64
```

Fig 8.

Está dispersion tambien lo podemos ver analizando los histogramas de cada columna(atributo) tal cómo se muestra en la Fig 9. Y Fig 10., podemos validar está informacion:

```
data = pd.read_csv('dataset.csv')

# Filtramos las columanas para poder analizar por grupos
columns = data.columns

numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
cols = [c for c in data.columns if data[c].dtype in numerics]

cols1 = ['chroma_stft', 'rmse', 'spectral_centroid', 'spectral_bandwidth', 'rolloff', 'zero_crossing_rate']
cols2 = ['mfcc1', 'mfcc2', 'mfcc3', 'mfcc4', 'mfcc5', 'mfcc6', 'mfcc7', 'mfcc8', 'mfcc9', 'mfcc10']
cols3 = ['mfcc11', 'mfcc12', 'mfcc13', 'mfcc14', 'mfcc15', 'mfcc16', 'mfcc17', 'mfcc18', 'mfcc19', 'mfcc20']

# Generamos 3 tipos de dataset para entender la dispercion que tienen los datos
data1 = data[cols1]
data2 = data[cols2]
data3 = data[cols3]

# Creamos los histogramas
ax1 = data1.hist(bins= 10, alpha=0.5)
ax1 = data2.hist(bins= 10, alpha=0.5)
ax1 = data3.hist(bins= 10, alpha=0.5)
plt.show()
```

Fig 9.

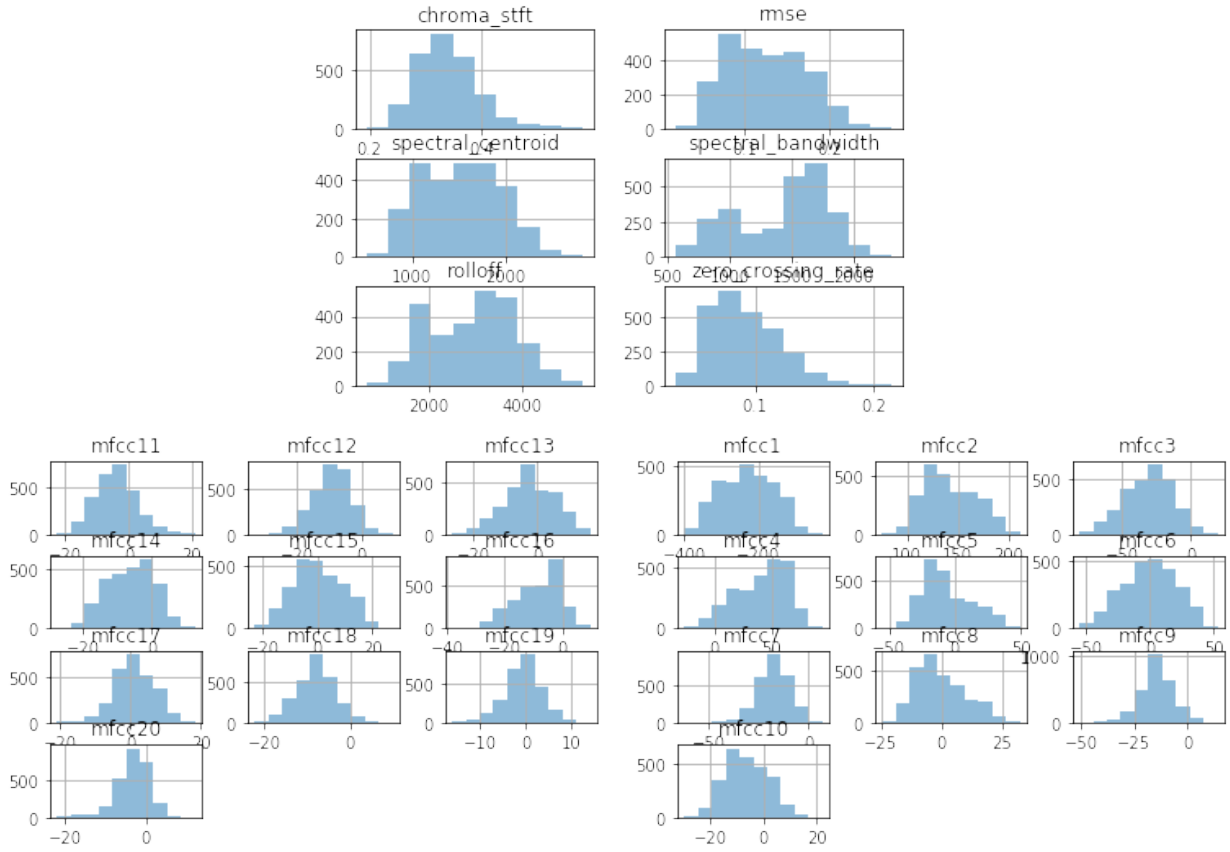


Fig 10.

Par poder eliminar los datos extremos usaremos los Quartiles con un IRQ = Q3-Q1. Dividimos la data en 4 cuartiles:

- Primer cuartil (Q1) = 25% de los datos.
- Segundo cuartil (Q2) = 50% de los datos
- Tercer cuartil (Q3) = 75 % de los datos.
- Cuarto cuartil(Q4) = 100% de los datos

Luego los valores extremos lo obtenemos con el siguente filtro(intervalo), el cual usaremos para remover los outlier con python tal cómo se muestran en la Fig 11.. En la Fig 12., vemos los outliers en boxplot antes de la remocion. En la Fig 13., vemos los datos sin los outliers.

Filtro: [Q1 - 1.5 * IQR, Q3 + 1.5 * IQR]

```
1
2  data = pd.read_csv('dataset.csv')
3
4  # Calculamos los quartiles y el IQR
5  Q0 = data.quantile(0.00)
6  Q1 = data.quantile(0.25)
7  Q2 = data.quantile(0.50)
8  Q3 = data.quantile(0.75)
9  Q4 = data.quantile(1.00)
10 print(Q0, data.quantile(0.05), Q1, Q2, Q3, data.quantile(0.95), Q4)
11 print("\n")
12
13 # Creamos el IRQ con el 3er y 1er quartil
14 IQR= Q3 - Q1
15
16 # Creamos el filtro que nos validara la existencia de valores extremos que este fuera del 1.5*IRQ en cada colum
17 filtro = ((data < (Q1 - 1.5 * IQR)) | (data > (Q3 + 1.5 * IQR))).any(axis=1)
18 outliers    = data[filtro]
19 data_new = data[~filtro]
20
21 # Validamos luego del filtro
22 ax1 = data_new.hist(bins= 20, alpha=0.5)
23 plt.show()
24
25 # Grafico de los datos extremos
26 plt.figure(figsize=(20,20))
27 outliers.boxplot()
28 plt.show()
29
30 # Grafica sin los datos extremos
31 plt.figure(figsize=(20,20))
32 data_new.boxplot()
33 plt.show()
34
35
```

Fig 11.

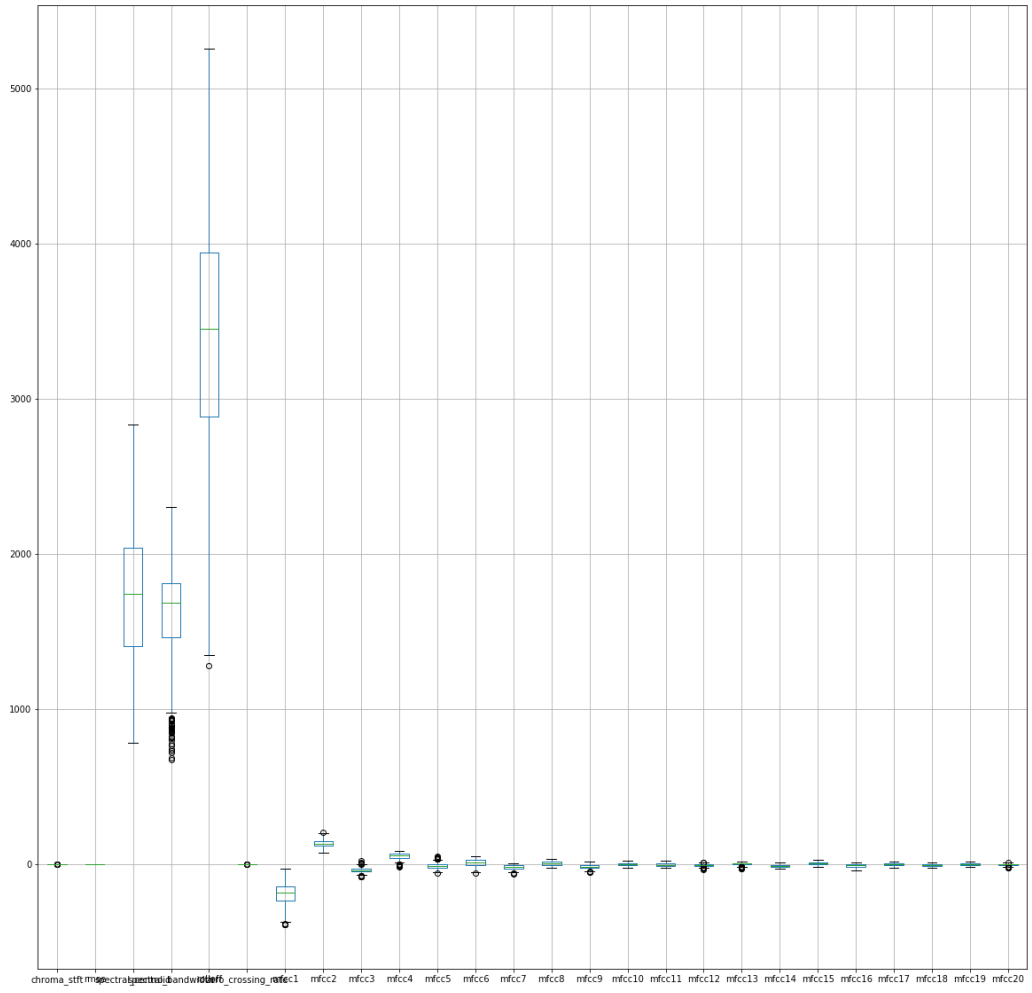

5000
4000
3000
2000
1000
0
chroma_stft
rmse
spectral_centroid
spectral_bandwidth
rolloff
zero_crossing_rate
mfcc1
mfcc2
mfcc3
mfcc4
mfcc5
mfcc6
mfcc7
mfcc8
mfcc9
mfcc10
mfcc11
mfcc12
mfcc13
mfcc14
mfcc15
mfcc16
mfcc17
mfcc18
mfcc19
mfcc20

Fig 12.

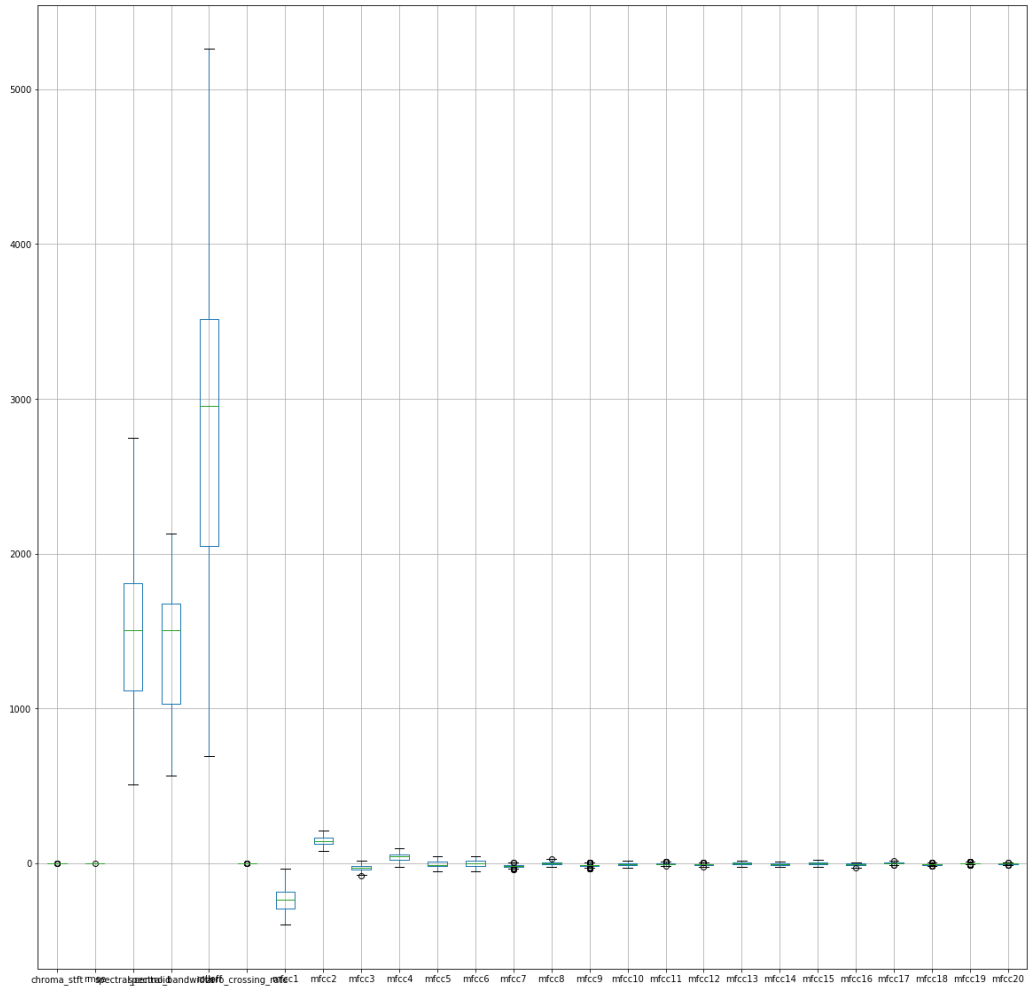


Fig 13.

Luego de está etapa tenemos una nueva data que la usaremos para balancear los datos mediante las clases.

3. Balanceo de datos mediante las clases

Primero balancear los datos, primero verificaremos sí las clases estan balanceadas. Luego usaremos la tecnica de Over-Sample que se encarga de rellenar en la cantidades minoritarias. En la Fig14., podemos ver el codigo usado en python. Y en la Fig 15., podemos ver los cambios antes y despues.

```
###### 3. Balanceo de datos mediante las clases #######
# Primero verificamos que una gran diferencia entre la clase "otros" y las otras clases slogan1,2,3,4,5...

data = pd.read_csv('dataset.csv')
otros, slogan1, slogan2, slogan3, slogan4, slogan5, slogan6, slogan7, slogan8, slogan9, slogan10 = data["clase"]

# Segun esto podemos verificar que existe un gran desbalance entre la clase "otros" y las otras clases
# Cantidad de filas por tipo de clase
print("Cantidad de filas por Clase: \n", data["clase"].value_counts())

#Creamos un filtro para cada clase
g0 = data.loc[data["clase"]=="otros"]
g1 = data.loc[data["clase"]=="slogan1"]
g2 = data.loc[data["clase"]=="slogan2"]
g3 = data.loc[data["clase"]=="slogan3"]
g4 = data.loc[data["clase"]=="slogan4"]
g5 = data.loc[data["clase"]=="slogan5"]
g6 = data.loc[data["clase"]=="slogan6"]
g7 = data.loc[data["clase"]=="slogan7"]
g8 = data.loc[data["clase"]=="slogan8"]
g9 = data.loc[data["clase"]=="slogan9"]
g10 = data.loc[data["clase"]=="slogan10"]

# Luego aplicamos la tecnica de Over-sample, que es utilizada para rellenar a las cantidades minoritarias
# que en este caso son las clases slogan1, slogan2, ... para que se iguale en cantidad con la clase otros.
g1_over = g1.sample(otros, replace=True)
g2_over = g2.sample(otros, replace=True)
g3_over = g3.sample(otros, replace=True)
g4_over = g4.sample(otros, replace=True)
g5_over = g5.sample(otros, replace=True)
g6_over = g6.sample(otros, replace=True)
g7_over = g7.sample(otros, replace=True)
g8_over = g8.sample(otros, replace=True)
g9_over = g9.sample(otros, replace=True)
g10_over = g10.sample(otros, replace=True)

data_balanced = pd.concat([g1_over, g2_over, g3_over, g4_over, g5_over, g6_over, g7_over, g8_over, g9_over, g10

# Volvemos a verificar el desbalance entre la clase "otros" y las otras clases, y confirmamos que ya no existe
# el desbalance.
# Cantidad de filas por tipo de clase luego del balanceo
print("Cantidad de filas por Clase: \n", data_balanced['clase'].value_counts())
print(data_balanced.shape)
data_balanced.head(10)
```

Fig 14.

```
Cantidad de filas por Clase:
 otros       2870
slogan5       32
slogan8       30
slogan9       23
slogan10      21
slogan4       20
slogan7       15
slogan6       11
slogan2        8
slogan1        6
slogan3        5
Name: clase, dtype: int64


Cantidad de filas por Clase:
 slogan4     2870
slogan6     2870
slogan2     2870
slogan7     2870
slogan5     2870
slogan3     2870
slogan1     2870
slogan9     2870
slogan8     2870
otros       2870
slogan10    2870
Name: clase, dtype: int64
(31570, 28)
```

Fig 15.

4. Eliminación de columnas innecesarias

Luego hacemos un drop a las columnunas innecesarias. En Fig 16., vemos el codigo usado.

```
###### 4. Eliminacion de columnas innesesarias #######
# Eliminamos las columanas que no utilizamos como la columna "filename" que es usada como nombre del archivo.
data = data.drop(['filename'],axis=1)
data.head(10)

```

	chroma_stft	rmse	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1	mfcc2	mfcc3	mfcc4	...	mfc
0	0.246486	0.179542	992.162413	791.586116	1866.659546	0.061218	-274.512939	158.679016	-40.953972	-4.987597	...	-18.612
1	0.325838	0.126466	1550.898981	1528.945827	3450.403684	0.076562	-219.222626	138.284988	-40.178246	76.730743	...	-5.340
2	0.346125	0.164938	1896.240906	1642.376334	3531.779161	0.110685	-119.659988	131.901947	-61.661648	82.906067	...	-17.611
3	0.312588	0.130916	975.252188	760.078493	1714.917755	0.072769	-267.866577	190.036835	-38.196693	0.476744	...	-17.666
4	0.364249	0.260357	1794.745267	2021.160286	4221.175509	0.081494	-63.521839	141.021225	-26.212214	73.127258	...	-8.537
5	0.315877	0.108854	1521.132893	1554.654395	2876.685706	0.086698	-241.809998	128.061890	-12.415193	30.308168	...	-2.868
6	0.318739	0.138269	1355.413032	1455.839847	2784.293241	0.061152	-210.396255	142.812653	-30.687391	34.047470	...	-5.921
7	0.259842	0.096981	1692.511085	1782.413156	3290.657363	0.102130	-243.032455	121.629257	-27.278633	47.174007	...	-11.309
8	0.311456	0.160802	1829.948126	1981.262729	3908.693677	0.087402	-157.531326	132.853424	-14.537085	49.044445	...	-10.712
9	0.284885	0.151329	1696.880725	1545.197061	3245.504406	0.094878	-223.183151	112.376274	-10.295667	67.448456	...	-9.507

10 rows × 27 columns

Fig 16.

5. Normalizacion y codificación de la clase

Normalizaremos los datos y encodficamos la clase para poder hacer las predicciones. En la Fig 17., para esto usaremos el siguiente codigo en python.

```
###### 5. Normalizacion #######

X = data.iloc[:, :-1]
print(np.array(data.iloc[:, :-1]))

minmax_scale = preprocessing.MinMaxScaler().fit(X)
X = minmax_scale.transform(np.array(data.iloc[:, :-1], dtype = float))

############ 6. Codificacion de la clase ##############

clase_list = data.iloc[:,-1]
encoder = LabelEncoder()
y = encoder.fit_transform(clase_list)
data["clase"] = y
print(y)

data.head(10)
```

Fig 17.

2. MODELAMIENTO Y ENTRENAMIENTO

Para el modelamiento, cómo es un escenario de clasificación multistate, usaremos una red neuronal artificial de 3 capas, que utilizara una función de activación de salida softmax para poder manejar estos casos de clasificación multistate. Tambien usaremos el metodo k-fold para dividir la data de training y test. En la Fig 18., vemos una descripcion grafica del metodo k-fold.

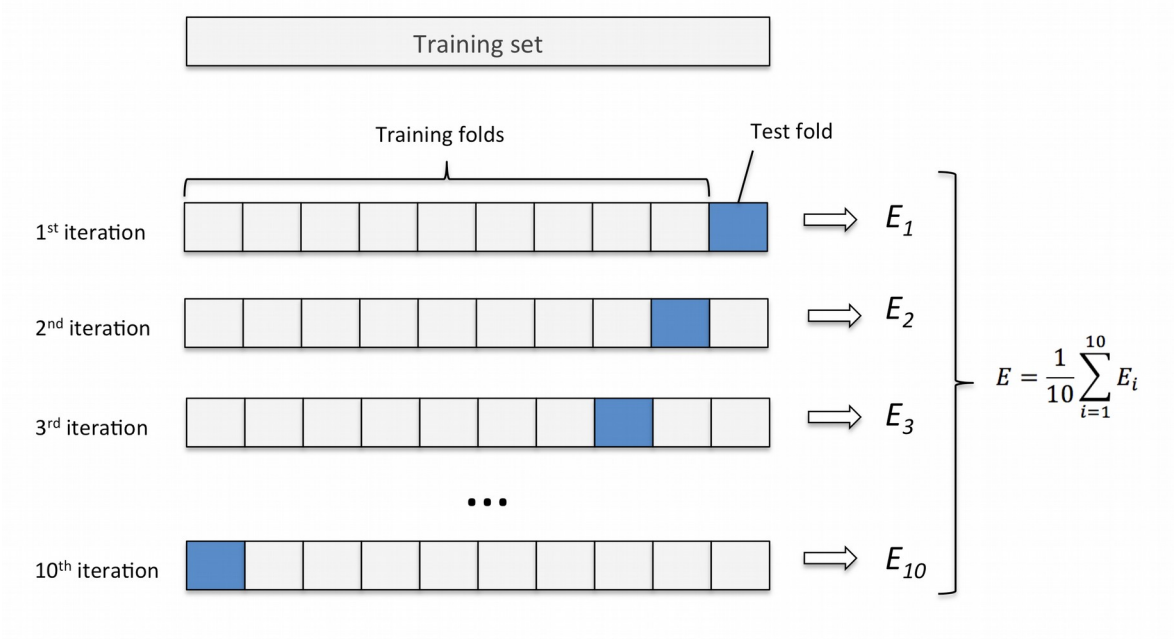


Fig 18.

Para trabajo la data de entrenamiento y test usaremos especificamente Stratified-KFold, que es una version mejorada de k-fold que considera la separacion del dataset en base a la misma cantidad de etiquetas. Para esto usaremos el siguiente codigo en python:

```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
```

En la Fig 19., vemos la implementacion del modelo usando la tecnica de k-fold.

```python

# Creacion del modelo con el algoritmo MLPClassifier ANN con las siguientes caracteristicas
# 1. Uso funcion de activacion softmax en la ultima capa(Esta configuracion esta en la logica del algoritmo)
# 2. Uso maximo de 300 iteracion
# 3. Uso de 3 capas
clf = MLPClassifier(random_state=1, max_iter=300)

# Creamos 5 iteraciones del algoritmo con Stratified K-Fold
fold_no = 1
for train_indices, test_indices in skf.split(X,y):
    clf.fit(X[train_indices], y[train_indices])

    # Predicciones
    y_test_prob = clf.predict_proba(X[test_indices])
    y_test_pred = clf.predict(X[test_indices])

    # Evaluacion del modelo
    score = clf.score(X[test_indices], y[test_indices])
    accuracy = accuracy_score(y[test_indices],y_test_pred)
    recall = recall_score(y[test_indices], y_test_pred, average=None)
    precision = precision_score(y[test_indices], y_test_pred, average=None)

    print('Fold',str(fold_no))
    print("Score: ", score)
    print("Acurracy: ", accuracy*100)
    print("Recall: ", recall)
    print("Precision: ", precision*100)

    print("Funcion de acctivacion en la ultima capa: ", clf.out_activation_)
    print("Perdida obtenida durante el entranamiento")
    print(clf.loss_)
    print(clf.best_loss_)
    print("Numero iteraciones")
    print(clf.n_iter_)
    print("Numero de capas")
    print(dim(clf.coefs_))

    cmatrix = confusion_matrix(y[test_indices], y_test_pred)
    plot_confusion_matrix(clf, X[test_indices], y[test_indices])
    plt.show()

    fold_no += 1
```

Fig 19.

En los resultados de cada iteracion se muestran en la Fig 20. Y Fig 21.

- Fold 1
- Score: 0.9968938070277616
- Acurracy: 99.68938070277616
- Recall: [0.96588486 1.   1.   1.   1.   1.
- 1.   1.   1.   1.   1.   ]
- Precision: [100.   99.57537155 99.78678038 99.36305732 99.78678038
- 99.36305732 100.   99.78723404 100.   99.15254237
- 99.78678038]
- Funcion de acctivacion en la ultima capa: softmax
- Perdida obtenida durante el entranamiento
- 0.004932725638218364
- 0.004932725638218364
- Numero iteraciones
- 198
- Numero de capas
- [2, 26, 100]

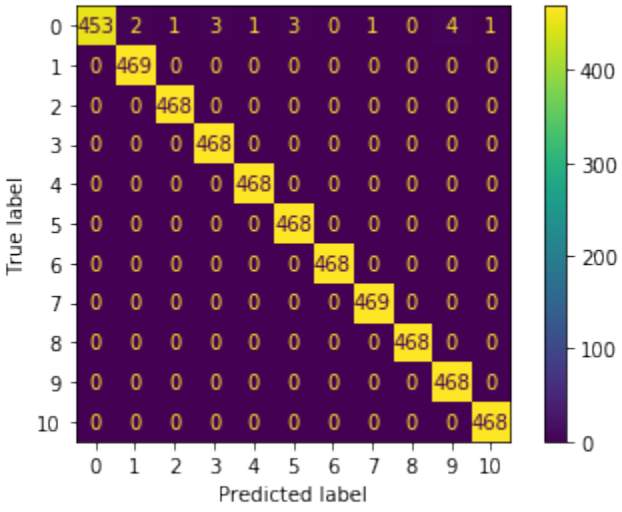


Fig 20.

Y la curve de error es la siguiente:

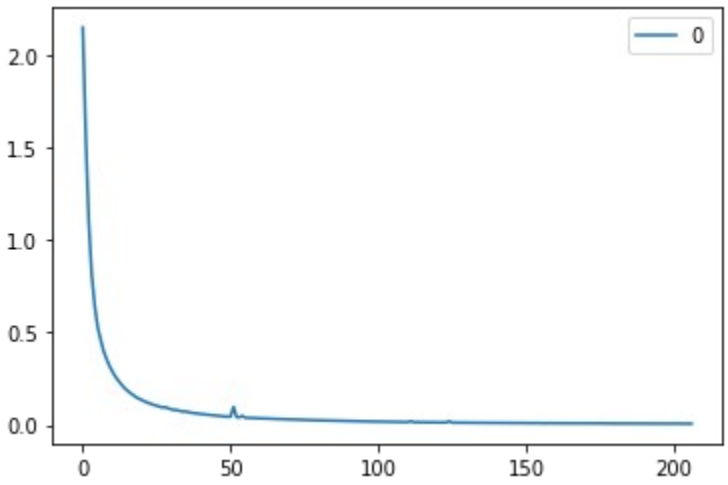


Fig 21.

Luego de las 5 iteraciones podemos concluir que se obtuvo una Acurracy promedio de 99.67%.

- Fold 1 Acurracy: 99.68938070277616%
- Fold 2 Acurracy: 99.68932038834951%
- Fold 3 Acurracy: 99.66990291262135%
- Fold 4 Acurracy: 99.8252427184466%
- Fold 5 Acurracy: 99.68932038834951%

3. EVALUACIÓN DEL MODELO

Para la evaluación del modelo usaremos el formato pickle el cual usaremos despues para evaluar el modelo luego de su entrenamiento. Con el comando “pickle.dump” podemos crear el modelo y con el comando “pickle.load” cargamos el modelo.

```
############################################################
################### Evaluacion del Modelo ###################

# Creacion del clasificador
pickle.dump(clf, open('classifier.pkl', 'wb'), protocol=4)

# Cargamos el modelo para su evaluacion
clf = pickle.load(open('classifier.pkl', 'rb'))
```

Fig 22.

Luego validamos con data real y tenemos el siguiente resultado:

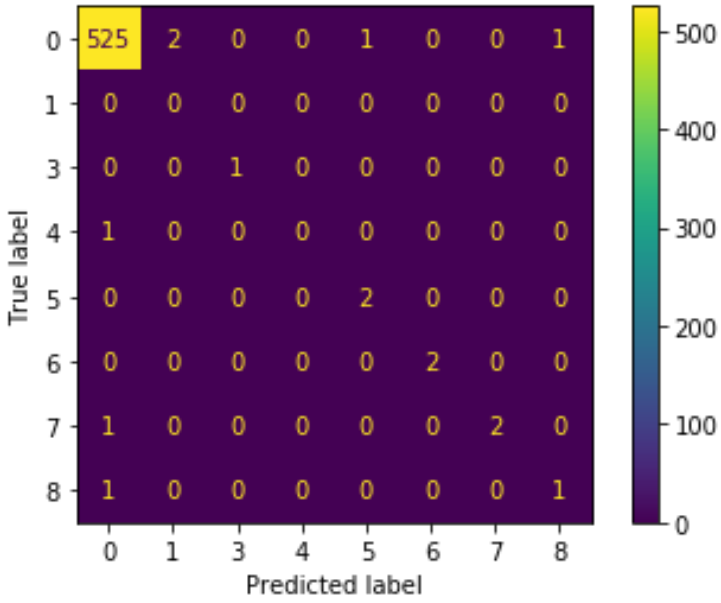
```
# Procesamiento de datos: Cargar la data del CSV, codificacion de etiquetas, división de datos en conjunto de e
data = pd.read_csv('dataeval.csv')

# Dropping unneccesary columns
data = data.drop(['filename'],axis=1)

#Encoding the Labels
genre_list = data.iloc[:, -1]
encoder = LabelEncoder()
y = encoder.fit_transform(genre_list)

###### 5. Normalizacion #######
X = data.iloc[:, :-1]
minmax_scale = preprocessing.MinMaxScaler().fit(X)
X = minmax_scale.transform(np.array(data.iloc[:, :-1], dtype = float))

print("Predic"clf.predict(X))

print("Evaluacion del Modelo con la data de prueba: ")
print("============================================ \n")
for i in range(0,len(X)):
    print('Prediction %s Probability: %.2f%%' %\
          (label[clf.predict(X)[i]], clf.predict_proba(X)[i].max()*100))
```

```
7 [1 0 0 0 0 0 0]
Evaluacion del Modelo con la data de prueba:
============================================

Prediction slogan1 Probability: 98.91%
Prediction otros Probability: 100.00%
Prediction otros Probability: 99.31%
Prediction otros Probability: 100.00%
Prediction otros Probability: 100.00%
Prediction otros Probability: 100.00%
Prediction otros Probability: 100.00%

<Figure size 576x576 with 0 Axes>
```

Fig 23.

V. RESULTADOS

Los resultados encontrados inicialmente, con la data original se encontró que la Clase otros (0) es la que se tenía más información haciendo que el modelo aprenda más este patrón:



Y durante la evaluación de slogans que son diferentes de otros, el modelo predijo erróneamente:

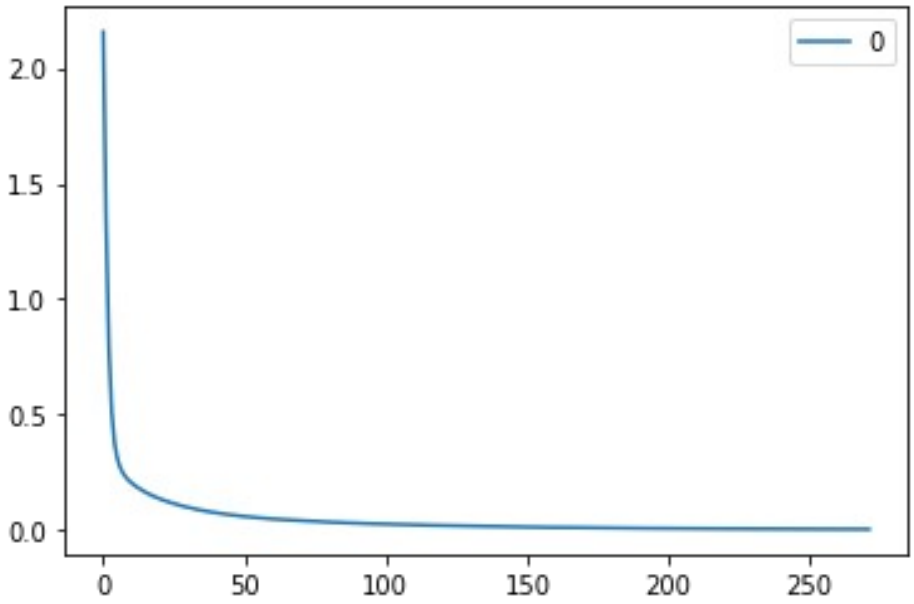
```
97
98  print("Evaluacion del Modelo con la data de prueba: ")
99  print("============================================ \n")
100 for i in range(0,len(X)):
101     print('Prediction %s Probability: %.2f%%' %\
102           (label[clf.predict(X)[i]], clf.predict_proba(X)[i].max()*100))
```

```
Evaluacion del Modelo con la data de prueba:
============================================

Prediction otros Probability: 98.53%
Prediction slogan7 Probability: 99.79%
Prediction otros Probability: 100.00%
Prediction otros Probability: 100.00%
Prediction otros Probability: 100.00%
Prediction otros Probability: 99.71%
Prediction otros Probability: 95.71%
Prediction otros Probability: 99.99%
Prediction otros Probability: 99.99%
Prediction otros Probability: 99.77%

<Figure size 576x576 with 0 Axes>
```

Sin embargo, se encontró que el número de iteraciones que se usó fue 272 menos del que se configuró (max_iter=300) y el accuracy era de 98.7%:

- Número iteraciones: 272
- Acurracy: 98.70370370370371%

Curva de perdida:



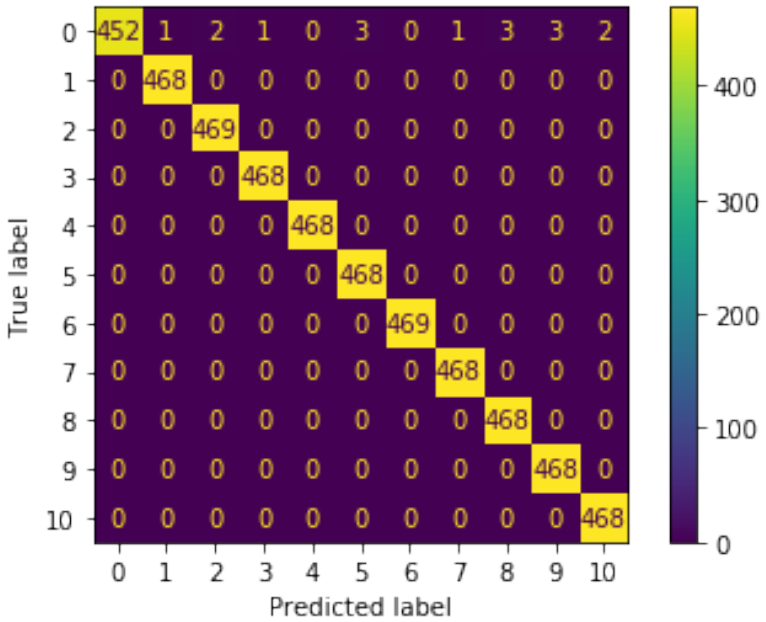
Luego despues de hacer el balanceo de datos, podemos ver que sí predijo correctamente:

```
100
101 print("Evaluacion del Modelo con la data de prueba: ")
102 print("============================================ \n")
103 for i in range(0,len(X)):
104     print('Prediction %s Probability: %.2f%%' %\
105           (label[clf.predict(X)[i]], clf.predict_proba(X)[i].max()*100))
```

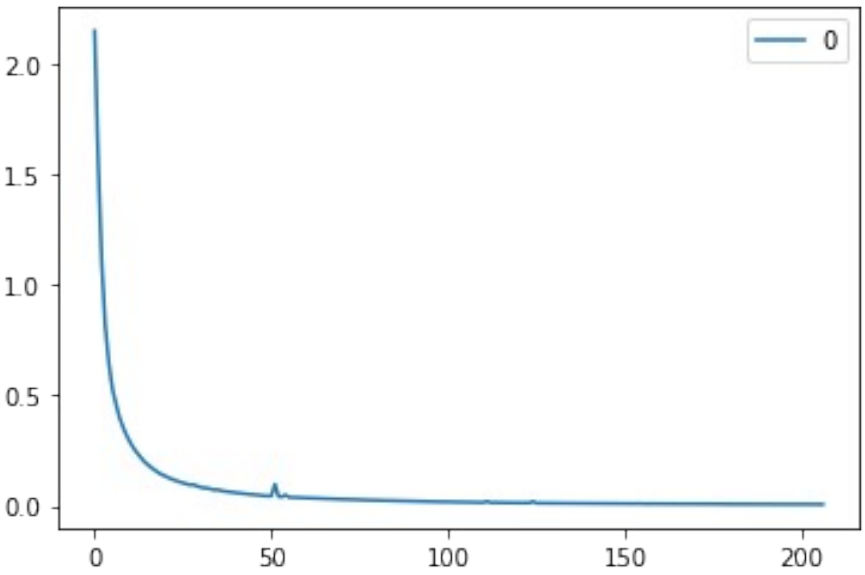
```
7 [1 0 0 0 0 0 0]
Evaluacion del Modelo con la data de prueba:
============================================

Prediction slogan1 Probability: 98.91%
Prediction otros Probability: 100.00%
```

Tambien se encontró que el número de iteraciones que se usó fue 207 menos del que se configuró (max_iter=300) y el accuracy era de 99.67%:

- Acurracy: 99.68932038834951
- Funcion de acctivacion en la ultima capa: softmax
- Perdida obtenida durante el entranamiento: 0.006835649966663543
- Numero iteraciones: 207
- Numero de capas: [2, 26, 100]



Curva de perdida:



VI. CONCLUSIONES

1. Cuando se trabaja con señales, es necesario transformar esos datos a infomacion que el modelo pueda leer. Por está razon es que se usaron librerias python cómo librosa, audiosegment, entre otros.
2. Para analisis de audio stream, se tienen que considerar la estructura que tiene dicho canal con respecto a su programación de entrevistas y anuncios. Esto ayuda a enfocarnos solo en lo mas importante.
3. Usando la tecnica de k-fold podemos validar que tan buena es nuestro data set debido que en cada iteracion se usan diferentes proporciones del dataset y se evalua en el modelo.
4. El desbalance de datos genera sesgo en el modelo. Para aprender necesitamos asignar la misma cantidad de columnas para cada etiqueta, haciendo que el proceso de aprendizaje no haya sesgo. Para que cada categoría tenga la misma probabilidad
5. Luego de un balanceo y limpieza de datos, vemos que el modelo realmente aprendió y pudemos comprobarlo mediante la evaluación del modelo.

VII. RECOMENDACIONES

1. Cuando se trabaja con audio, es mejor procesarlos en estado original, por que hacer cambios a otro formato genera perdida de informacion, lo cual puede afectar en la creacion del dataset y en los recursos de computo para su conversion.
2. Para el etiquetado se tuvo problemas con el tiempo, porque era manual, sin embargo el tiempo puede disminuir sí es que conocemos en que parte se encuentra la informacion mas importante y la podemos hacer con particiones segun el patron de entrevistas y anuncios.
3. Es importante tener en cuenta el entendimiento, preparacion y limpieza de datos al momento de crear el dataset y en lo posterior para entrenar el modelo, porque un mal dataset nos puede llevar a un sesgo que se traduce en una mala prediccion.
4. Al momento de evaluar el modelo es necesario considerar los datos de entrada tanto sí es de manera local o real-time, porque estos datos de entrada(rawdata) aun no estan preparados o listos para ser cargados al modelo debido a que necesitan tener el formato adecuado para el modelo lo pueda leer. Esto ocasiona que tenemos que tener en mente pasos previos cómo conversiones, uso computacional, tiempo de respuesta, etc.

VIII. REFERENCIAS

1. Web Radio Automation for Audio Stream Management in the Era of Big Data. Nikolaos Vryzas *, Nikolaos Tsipas and Charalampos Dimoulas. 11 April 2020.
2. Human Activity Recognition Using Smartphones.Erhan BÜLBÜL, Aydın ÇETİN, İbrahim Alper DOĞRU. ©2018 IEEE

Anexo 1: CODIGO FUENTE

```
#################################################################
###################### Librerias usadas  ########################
from sklearn.neural_network import MLPClassifier
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import librosa
from pydub import AudioSegment
import csv
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
import pickle
#################################################################

#################################################################
################### Preparacion de Datos  ###################

# Crear expectogramas a partir de los audios.
# 1. Convertimos los audios de mp3 a wav.
# 2. Convertir los audios.wav en png para poder crear los atributos.

# Definimos el formato de la imagen(color map)
cmap = plt.get_cmap('inferno')
plt.figure(figsize=(8,8))

# Creamos la lista de slogans
slogan = 'slogan1 slogan2 slogan3 slogan4 slogan5 slogan6 slogan7 slogan8 slogan9 sloga10
otros'.split()

# ./TestData es la ruta donde esta todos los audios etiquetados Audio00x_xyz_sloganw.
path = r'./Test'
extension = '.mp3'


for root, dirs_list, files_list in os.walk(path):
        for file_name in files_list:
        if os.path.splitext(file_name)[-1] == extension:
                file_name_path = os.path.join(root, file_name)
                name=os.path.splitext(file_name)[0]
                sound = AudioSegment.from_mp3(file_name_path)
                sound.export(f'{file_name_path[:-3]}.wav', format="wav")
                audio = f'{file_name_path[:-3]}.wav'
                y, sr = librosa.load(audio, mono=True, duration=5)
                plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, cmap=cmap, sides='default',
mode='default', scale='dB');
                plt.axis('off');
                plt.savefig(f'{file_name_path[:-3]}png')
#################################################################
```

```
#################################################################
################### Creacion del dataset  ###################

# Funcion para hacer el mapping de los archivos de audio con la clase
def identificar_parametros(filename):
        cadena1 = file_name
        if cadena1.find("otros")>=0 :
        return "otros"
        if cadena1.find("slogan1")>=0 :
        return "slogan1"
        if cadena1.find("slogan2")>=0 :
        return "slogan2"
        if cadena1.find("slogan3")>=0 :
        return "slogan3"
        if cadena1.find("slogan4")>=0 :
        return "slogan4"
        if cadena1.find("slogan5")>=0 :
        return "slogan5"
        if cadena1.find("slogan6")>=0 :
        return "slogan6"
        if cadena1.find("slogan7")>=0 :
        return "slogan7"
        if cadena1.find("slogan8")>=0 :
        return "slogan8"
        if cadena1.find("slogan9")>=0 :
        return "slogan9"
        if cadena1.find("sloganD")>=0 :
        return "slogan10"
        else:
        return 999

# Creamos la cabecera del data set
header = 'filename chroma_stft rmse spectral_centroid spectral_bandwidth rolloff
zero_crossing_rate'
for i in range(1, 21):
        header += f' mfcc{i}'
header += ' clase'
header = header.split()

# Creamos el archivo dataset.csv que tiene como primera columna la variale header
file = open('dataset.csv', 'w', newline='')
with file:
        writer = csv.writer(file)
        writer.writerow(header)

# Extraemos las caracteristicas desde el Espectograma que se convertiran en atributos del
dataset:
# 1. Mel-frequency cepstral coefficients (MFCC)
# 2. Spectral Centroid
# 3. Zero Crossing Rate
# 4. Chroma Frequencies
# 5. Spectral Roll-off.
path = r'./Test'
extension = '.wav'
for root, dirs_list, files_list in os.walk(path):
        for file_name in files_list:
        if os.path.splitext(file_name)[-1] == extension:
                file_name_path = os.path.join(root, file_name)
                audio = file_name_path
                y, sr = librosa.load(audio, mono=True, duration=30)
                rmse = librosa.feature.rms(y=y)
                chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
```

```
                spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
                spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
                rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
                zcr = librosa.feature.zero_crossing_rate(y)
                mfcc = librosa.feature.mfcc(y=y, sr=sr)
                p_slogan = identificar_parametros(file_name)
                to_append = f'{file_name_path} {np.mean(chroma_stft)} {np.mean(rmse)}
{np.mean(spec_cent)} {np.mean(spec_bw)} {np.mean(rolloff)} {np.mean(zcr)}'
                for e in mfcc:
                to_append += f' {np.mean(e)}'
                to_append += f' {p_slogan}'
                file = open('dataset.csv', 'a', newline='')
                with file:
                writer = csv.writer(file)
                writer.writerow(to_append.split())

# Cargamos y leemos los datos en la variable pandas data.
# Contiene las etiquetas y atributos.
data = pd.read_csv('dataset.csv')
data.head(10)
#############################################################

#########################################################################################################
##
############## Preparacion, Entendimiento, Limpieza y Normalizacion del dataset
##################

###### 1. NULLs or NA #######
# Verificamos si existen datos Null, NA en el dataset.
data = pd.read_csv('dataset.csv')
print("Datos Null y NA en el dataset")
print("============================== \n")
print(data.isnull().sum())
print("\n")
print("Datos Null en todo el dataset: ", data.isnull().values.any())
print("Datos NA en todo el dataset: ", data.isna().sum().sum())

###### 2. Outlayers or Datos extremos #######
# Verificamos si existen datos extremos en el dataset.
data = pd.read_csv('dataset.csv')

# Calculamos los quartiles y el IQR
Q0 = data.quantile(0.00)
Q1 = data.quantile(0.25)
Q2 = data.quantile(0.50)
Q3 = data.quantile(0.75)
Q4 = data.quantile(1.00)


# Creamos el IRQ con el 3er y 1er quartil
IQR= Q3 - Q1

# Creamos el filtro que nos validara la existencia de valores extremos que este fuera del 1.5*IRQ
en cada columna
filtro = ((data < (Q1 - 1.5 * IQR)) | (data > (Q3 + 1.5 * IQR))).any(axis=1)
outliers       = data[filtro]

# Sacamos los datos extemos y actualizamos el dataset
data_new = data[~filtro]

# Actualizamos dataset
data = data_new

###### 3. Balanceo de datos mediante las clases #######
```

```
# Primero verificamos que una gran diferencia entre la clase "otros" y las otras clases
slogan1,2,3,4,5...

# Buscamos los valores de cada clase en base a filas
otros, slogan1, slogan2, slogan3, slogan4, slogan5, slogan6, slogan7, slogan8, slogan9, slogan10
= data["clase"].value_counts()

# Segun esto podemos verificar que existe un gran desbalance entre la clase "otros" y las otras
clases
# Cantidad de filas por tipo de clase
print("Cantidad de filas por Clase: \n", data["clase"].value_counts())

#Creamos un filtro para cada clase
g0 = data.loc[data["clase"]=="otros"]
g1 = data.loc[data["clase"]=="slogan1"]
g2 = data.loc[data["clase"]=="slogan2"]
g3 = data.loc[data["clase"]=="slogan3"]
g4 = data.loc[data["clase"]=="slogan4"]
g5 = data.loc[data["clase"]=="slogan5"]
g6 = data.loc[data["clase"]=="slogan6"]
g7 = data.loc[data["clase"]=="slogan7"]
g8 = data.loc[data["clase"]=="slogan8"]
g9 = data.loc[data["clase"]=="slogan9"]
g10 = data.loc[data["clase"]=="slogan10"]

# Luego aplicamos la tecnica de Over-sample, que es utilizada para rellenar a las cantidades
minoritarias
# que en este caso son las clases slogan1, slogan2, ... para que se iguale en cantidad con la
clase otros.
g1_over = g1.sample(otros, replace=True)
g2_over = g2.sample(otros, replace=True)
g3_over = g3.sample(otros, replace=True)
g4_over = g4.sample(otros, replace=True)
g5_over = g5.sample(otros, replace=True)
g6_over = g6.sample(otros, replace=True)
g7_over = g7.sample(otros, replace=True)
g8_over = g8.sample(otros, replace=True)
g9_over = g9.sample(otros, replace=True)
g10_over = g10.sample(otros, replace=True)

# Balanceamos los datos
data_balanced = pd.concat([g1_over, g2_over, g3_over, g4_over, g5_over, g6_over, g7_over,
g8_over, g9_over, g10_over, g0], axis=0)

# Volvemos a verificar el desbalance entre la clase "otros" y las otras clases, y confirmamos que
ya no existe
# el desbalance.
# Cantidad de filas por tipo de clase luego del balanceo
print("\n")
print("Cantidad de filas por Clase: \n", data_balanced['clase'].value_counts())
print(data_balanced.shape)
data_balanced.head(10)

# Actualizamos dataset
data = data_balanced

###### 4. Eliminacion de columnas innesesarias #######
# Eliminamos las columanas que no utilizamos como la columna "filename" que es usada como nombre
del archivo.
data = data.drop(['filename'],axis=1)
data.head(10)

###### 5. Normalizacion #######

X = data.iloc[:, :-1]
```

```
print(np.array(data.iloc[:, :-1]))

minmax_scale = preprocessing.MinMaxScaler().fit(X)
X = minmax_scale.transform(np.array(data.iloc[:, :-1], dtype = float))

############ 6. Codificacion de la clase ##############


clase_list = data.iloc[:,-1]
encoder = LabelEncoder()
y = encoder.fit_transform(clase_list)
data["clase"] = y
print(y)

data.head(10)
#####################################################################


#############################################################
######### Creacion del Modelo y Entrenamiento ###############
#
# Para crear el modelo usaremos la tecnica Stratified K-fold (version mejorada de la tecnica K-
fold)
#para separar con la data de prueba y de entrenamiento

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)

def isnp(M):
    if isinstance(M, np.ndarray): return True
    return False

def dim(M):
        if isnp(M): M = list(M)

        if not type(M) == list:
        return []

        return [len(M)] + dim(M[0])


# Creacion del modelo con el algoritmo MLPClassifier ANN con las siguientes caracteristicas
# 1. Uso funcion de activacion softmax en la ultima capa(Esta configuracion esta en la logica del
algoritmo)
# 2. Uso maximo de 300 iteracion
# 3. Uso de 3 capas
clf = MLPClassifier(random_state=1, max_iter=300)

# Creamos 5 iteraciones del algoritmo con Stratified K-Fold
fold_no = 1
for train_indices, test_indices in skf.split(X,y):
        clf.fit(X[train_indices], y[train_indices])

        # Predicciones
        y_test_prob = clf.predict_proba(X[test_indices])
        y_test_pred = clf.predict(X[test_indices])

        # Evaluacion del modelo
        score = clf.score(X[test_indices], y[test_indices])
        accuracy = accuracy_score(y[test_indices],y_test_pred)
        recall = recall_score(y[test_indices], y_test_pred, average=None)
        precision = precision_score(y[test_indices], y_test_pred, average=None)

        print('Fold',str(fold_no))
```

```
        print("Score: ", score)
        print("Acurracy: ", accuracy*100)
        print("Recall: ", recall)
        print("Precision: ", precision*100)

        print("Funcion de acctivacion en la ultima capa: ", clf.out_activation_)
        print("Perdida obtenida durante el entranamiento")
        print(clf.loss_)
        print(clf.best_loss_)
        print("Numero iteraciones")
        print(clf.n_iter_)
        print("Numero de capas")
        print(dim(clf.coefs_))

        cmatrix = confusion_matrix(y[test_indices], y_test_pred)
        plot_confusion_matrix(clf, X[test_indices], y[test_indices])
        plt.show()

        fold_no += 1

#print(clf.loss_curve_)
pd.DataFrame(clf.loss_curve_).plot()
plt.show()
##############################################################################################


#################################################################
################### Evaluacion del Modelo ###################

# Creacion del clasificador
pickle.dump(clf, open('classifier.pkl', 'wb'), protocol=4)

# Cargamos el modelo para su evaluacion
clf = pickle.load(open('classifier.pkl', 'rb'))


label = {0: 'otros',
        1: 'slogan1',
        2: 'slogan2',
        3: 'slogan3',
        4: 'slogan4',
        5: 'slogan5',
        6: 'slogan6',
        7: 'slogan7',
        8: 'slogan8',
        9: 'slogan9',
        10: 'sloganD'}


cmap = plt.get_cmap('inferno')
plt.figure(figsize=(8,8))

slogan = 'slogan1 slogan2 slogan3 slogan3 slogan4 slogan5 slogan6 slogan7 slogan8 slogan9 sloganD
otros'.split()
path = r'./TestData'
extension = '.mp3'

for root, dirs_list, files_list in os.walk(path):
        for file_name in files_list:
        if os.path.splitext(file_name)[-1] == extension:
                file_name_path = os.path.join(root, file_name)
                name=os.path.splitext(file_name)[0]
                sound = AudioSegment.from_mp3(file_name_path)
```

```
                sound.export(f'{file_name_path[:-3]}.wav', format="wav")
                audio = f'{file_name_path[:-3]}.wav'
                y, sr = librosa.load(audio, mono=True, duration=5)
                plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, cmap=cmap, sides='default',
mode='default', scale='dB');
                plt.axis('off');
                plt.savefig(f'{file_name_path[:-3]}png')
                plt.clf()

header = 'filename chroma_stft rmse spectral_centroid spectral_bandwidth rolloff
zero_crossing_rate'
for i in range(1, 21):
        header += f' mfcc{i}'
header += ' clase'
header = header.split()

# Extraer caracteristicas desde el Espectograma: Mel-frequency cepstral coefficients (MFCC),
Spectral Centroid, Zero Crossing Rate, Chroma Frequencies, y Spectral Roll-off.
file = open('dataeval.csv', 'w', newline='')
with file:
        writer = csv.writer(file)
        writer.writerow(header)



extension = '.wav'
for root, dirs_list, files_list in os.walk(path):
        for file_name in files_list:
        if os.path.splitext(file_name)[-1] == extension:
                file_name_path = os.path.join(root, file_name)
                audio = file_name_path
                y, sr = librosa.load(audio, mono=True, duration=30)
                rmse = librosa.feature.rms(y=y)
                chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
                spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
                spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
                rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
                zcr = librosa.feature.zero_crossing_rate(y)
                mfcc = librosa.feature.mfcc(y=y, sr=sr)
                # to_append = f'{filename} {np.mean(chroma_stft)} {np.mean(rmse)}
{np.mean(spec_cent)} {np.mean(spec_bw)} {np.mean(rolloff)} {np.mean(zcr)}'
                p_slogan = identificar_parametros(file_name)
                to_append = f'{file_name_path} {np.mean(chroma_stft)} {np.mean(rmse)}
{np.mean(spec_cent)} {np.mean(spec_bw)} {np.mean(rolloff)} {np.mean(zcr)}'
                for e in mfcc:
                to_append += f' {np.mean(e)}'
                to_append += f' {p_slogan}'
                file = open('dataeval.csv', 'a', newline='')
                with file:
                writer = csv.writer(file)
                writer.writerow(to_append.split())

# Procesamiento de datos: Cargar la data del CSV, codificacion de etiquetas, división de datos en
conjunto de entrenamiento y prueba.
data = pd.read_csv('dataeval.csv')

# Dropping unneccesary columns
data = data.drop(['filename'],axis=1)

#Encoding the Labels
genre_list = data.iloc[:, -1]
encoder = LabelEncoder()
y = encoder.fit_transform(genre_list)

###### 5. Normalizacion #######
```

```
X = data.iloc[:, :-1]
minmax_scale = preprocessing.MinMaxScaler().fit(X)
X = minmax_scale.transform(np.array(data.iloc[:, :-1], dtype = float))

print("Predic"clf.predict(X))

print("Evaluacion del Modelo con la data de prueba: ")
print("========================================== \n")
for i in range(0,len(X)):
       print('Prediction %s Probability: %.2f%%' %\
       (label[clf.predict(X)[i]], clf.predict_proba(X)[i].max()*100))

################################################################################
```

Anexo 2: INDICACIONES

EXAMEN FINAL

MCC607 APRENDIZAJE AUTOMÁTICO Y MINERÍA DE DATOS

- Desarrolle el siguiente examen de forma grupal.
- Entregue: un archivo pdf con el desarrollo del procedimiento y un archivo comprimido con los resultados de cada paso, de tal forma que se demuestre que ha construido un procedimiento.

1. OBJETIVO

El objetivo de este examen es integrar los conceptos y procedimientos desarrollados en el curso mediante la construcción de un procedimiento para la clasificación de señales de captadas desde una transmisión de streaming en tiempo real.

La fuente de los datos es el streaming de RPP:
https://18493.live.streamtheworld.com/RADIO_RPP.mp3

Coloque este url en un explorador para escuchar el audio en tiempo real.

El procedimiento se elabora con la finalidad de identificar slogans de marcas en transmisiones de audio.

El grupo de trabajo implementa los procedimientos de adquisición de datos, almacenamiento, segmentación de las señales, etiquetados de señales, preparación del dataset, preparación de datos, entrenamiento del modelo y visualización de resultados.

Observar que el procedimiento debe de ser implementado para trabajar en tiempo real, pero las

etapas asociada al entrenamiento se deben de realizar fuera de línea.

2. PROCEDIMIENTO DE TRABAJO.

1. ADQUISICIÓN DE DATOS

Use la transmisión streaming proporcionada, por un tiempo de al menos 8 horas. Puede hacer uso de otro medio de transmisión que se originen en el país.

2. SEGMENTACIÓN DE LAS SEÑALES

Segmente las señales en bloques de k segundos.

Donde k debe ser una decisión del grupo de trabajo, justifique su decisión, en el sentido de que si este tiempo es necesario y suficiente para encontrar los patrones.

Explique con claridad cómo ha logrado identificar este parámetro.

Con fines de etiquetado almacene estos archivos en una carpeta.

3. ETIQUETADO DE LAS SEÑALES (entrenamiento)

Identifique al menos 10 slogan de marcas en las señales de audio, adicione una etiqueta denominada OTROS. El slogan debe corresponder a un sonido de tal forma que no sea posible “extraer el audio”

Eslogan	Descripción
Eslogan 1	Describa el eslogan

Otros	

Proceda a etiquetar las señales de audio manualmente, observe que este paso es crítico en el desarrollo del proyecto, dado que no se puede automatizar y que requiere la participación de personas con cierto nivel de experiencia. Explique cómo ha hecho para el proceso de etiquetado.

Cambie el nombre del archivo de audio con la etiqueta asignada.

4. PREPARACIÓN DEL DATASET (entrenamiento)

Desde cada segmento extraiga:

1. Al menos 15 atributos, generados por estadísticas de la señal.

2. Características en el dominio de la frecuencia, mediante el procedimiento FFT.

Diga cuales, presente las fórmulas, presente ejemplos de los resultados.

Slogan	Histogramas de las variables significativas
Slogan 1	Histograma 1, histgrama 2
Slogan 2	Histograma 3, histogramas 4

5. PREPARACIÓN DE DATOS

Aplique procedimiento de:

- Eliminación de valores extremos
- Eliminación de valores null
- Normalización de datos
- “Numerización” en caso de que sea necesario (encoder)

Dado que estas características no aportan suficiente información para que individualmente logren la diferenciación, no es necesario aplicar procedimientos de selección de atributos.

Presente una lista de los atributos que mejor explican la clase.

6. ENTRENAMIENTO DEL MODELO (entrenamiento)

Aplique un procedimiento de k-fold para la partición de los datos de prueba y entrenamiento (concepto no desarrollado en el curso).

Aplique procedimientos de sobre muestreo en caso de que sea necesario.

Entrena una red neuronal de 3 capas.

Determine la cantidad óptima de neuronas de la capa intermedia, mediante un procedimiento experimental.

7. CONSULTA DEL MODELO

Prepare un procedimiento para la consulta del modelo en tiempo real.
Prepare un web service para este propósito.

8. VISUALIZACIÓN DE LOS RESULTADOS

Con la finalidad de verificar que su procedimiento se encuentra operativo prepare una **página web** para que en tiempo real permita:

- Escuchar el audio de la radio.
- Presentar una línea de tiempo de cada uno de los segmentos.
- Asignar cada segmento a una categoría en forma automática.
- Presentar las estadísticas del tiempo en que cada eslogan se ha transmitido hasta el momento (veces y tiempo).

Enviar el link al email del profesor, al momento de enviar el examen final.

3. **ENTREGAR UN INFORME EN FORMATO DE PAPER.**

- TITULO, AUTOR Y FILIACIÓN
  - Título.
  - Apellidos, nombres
  - Fecha
  - Resumen
  - Introducción
-
- EL PROBLEMA
  - Especifique el problema a resolver, use sus propios términos para exponer el problema a resolver.
-
- METODOLOGÍA DE TRABAJO.
  - Exponga el procedimiento que desarrollará para resolver el problema, haga uso de los pasos planteados en este documento, al cual adicionará pasos adicionales no indicados. (observar que la metodología solo expone de qué manera resolverá el problema, pero no lo resuelve).
  - Use diagramas, explique conceptos, fórmulas y conceptos necesarios para entender el procedimiento.
-
- DESARROLLO DE LA SOLUCIÓN
  - Exponga cada uno de los pasos del proceso de solución, explique cada uno de estos pasos con ejemplos numéricos
  - Explique con claridad cada procedimiento, diga qué significa cada parámetro y cómo lo ha obtenido.
  - Acompañe cada paso con el código fuente que lo ejecuta.
-
- RESULTADOS
  - Resultados obtenidos
-
- CONCLUSIONES
  - Extraiga conclusiones significativas, presente cada conclusión

haciendo uso de indicadores numéricos o ratios. Una conclusión no es una opinión ni una apreciación, es la manifestación de un resultado (por ejemplo, aprendizajes en el proceso de etiquetado, tiempos de cálculo, identificación de los audios, etc.).

-
- RECOMENDACIONES
    - Presente al menos 3 recomendaciones para trabajos futuros, desarrolle estas recomendaciones para verificar su viabilidad.
-
- REFERENCIAS USADAS
    - Bibliografía leída, incluya solo la bibliografía que ha referenciado desde su informe.
-
- CÓDIGO FUENTE
    - Asegure que el código sea identado y que pueda ser leído por humanos (espaciado simple, letra Currier New de tamaño 8 a una sola columna)
    - Tengo cuidado en el tamaño de los tabs.
    - Encuadre cada función o método en un “cuadro”.
    - Hacer que el programa principal se denomine mainProcess

4. ## RECOMENDACIONES:

1. Inicie cada sección en una nueva hoja.
2. Coloque una sección de metodología, donde explica cada uno de los pasos a desarrollar.
3. Numere todas sus tablas, figuras y procedimientos.
4. Incluya este documento como parte de su informe.
5. Concluya con respecto a cada paso de la metodología planteada.
6. Incluya el url de la página con los resultados.
7. Nombre de examen: mcc607_ae2_g#_apellido1.apellido2.apellido3.apellido4.pdf.

Evite entregar adjuntos, todos los resultados deben estar incluidos en el informe.