CIMAT

Centro de Investigación en Matemáticas, A.C.

ANÁLISIS DE AUDIO PARA EXTRACCIÓN DE CARACTERÍSTICAS, SEGMENTACIÓN, CLASIFICACIÓN Y PREDICCIÓN

T E S I S

Que para obtener el grado de

Maestro en Ciencias

con Especialidad en

Computación y Matemáticas Industriales

Presenta

Omar Alberto Peña Olivares

Directores de Tesis:

Dr. Julio César Estrada Rico

Dr. Pablo Dávalos de la Peña

Autorización de la versión final

Guanajuato, Gto., 19 de diciembre de 2019

A mis padres, mi hermana y Lilia

Agradecimientos

Quiero agradecer a mi asesor, el Dr. Julio César Estrada Rico, por aceptar este tema de tesis, su tiempo, ideas, y retroalimentación. A mi coasesor Pablo Dávalos, por sus enseñanzas, por hacer de mi experiencia en Kueski algo increíble, por todo lo que aprendí, los seminarios, los comentarios de los Data Scientist, y sobre todo por proponer este tema de tesis.

A mi familia por acompañarme, por todo lo que me han dado, por estar al pendiente de mi. A mis padres y mi hermana, muchas gracias. A Lilia por estar todo este tiempo conmigo, por aguantarme y levantarme cuando me sentía profundamente agotado. Muchas gracias cariño.

A los amigos de Felipe, Marco Noguéz y Alejandro Frías, por este tiempo que vivimos juntos, por nuestras aventuras en Guanajuato, por esas noches de estudio, juegos y cervezas. Al buen Emilio, que llegó a nuestra pequeña comunidad en forma de piso de departamento. Aprendí mucho de ustedes.

A todos mis compañeros de generación,a los curvisarios, por estas tardes de basketball. A todos mis profesores, aprendí mucho y los admiro demasiado. A toda la comunidad Cimat-Demat por estos dos increíbles años.

Quiero agradecer a CONACyT por apoyarme económicamente para que puediera realizar mis estudios de maestría en el CIMAT.

Índice general

Agradecimientos IV

Prefacio X

1. Introducción 1

1.1. Audio 1

1.1.1. Características físicas del sonido 2

1.1.2. Aplicaciones del anális de audio 3

1.2. Representación digital del audio 6

1.2.1. Muestreo 6

1.2.2. Cuantización del audio 7

1.2.3. Códecs 8

1.2.4. Contenedores 9

2. Extracción de características 11

2.1. El espectograma 12

2.1.1. Transformada de Fourier 12

2.1.2. Audio framming 14

2.1.3. Windowing 14

2.2. Coeficientes de Mel 16

2.2.1. Escala de Mel 16

2.2.2. Bancos de filtros de Mel 17

2.2.3. Coeficientes cepstrales en las frecuencias de Mel 19

2.2.4. Deltas 20
2.2.5. Filtro de Savitzky–Golay 20

3. Técnicas de aprendizaje máquina 23
3.1. Birch 23
3.2. Redes Neuronales Convolucionales 25
3.3. XGBoost 28
3.4. Deep Features Synthesis 30
3.5. Bayesian Tunning and Bandits 33
3.6. Shap Values 35

4. Segmentación y clasificación 41
4.1. Clasificación de género 42
4.1.1. Clasificación de género basada en redes neuronales convolucionales 43
4.1.2. Experimentos y resultados 45
4.2. Segmentación basada en clustering 47
4.2.1. Experimentos y resultados 48
4.3. Audio a texto 50
4.3.1. Experimentos y resultados 50
4.4. Clasificación con base en texto 51
4.4.1. Term frequency – Inverse document frequency 51
4.4.2. Experimentos y resultados 52

5. Modelo de predicción de pago 55
5.1. Deep Feature Synthesis 59
5.2. Extracción de características de series de tiempo 62
5.3. Características espectrales 63
5.4. Selección de características 64
5.5. Modelo 66
5.6. Resultados 68

5.7. Interpretación 70

6. Conclusiones y trabajo a futuro 74

6.1. Conclusiones 74

6.1.1. Extracción de características 74

6.1.2. Clasificación de género 74

6.1.3. Segmentación de voces 75

6.1.4. Clasificador cliente-cobrador 75

6.1.5. Modelo de predicción de pago 75

6.2. Trabajo a futuro 76

Índice de figuras

1.1. Señal muestreada 7
1.2. Señal cuantizada en 11 niveles 8

2.1. Señal de audio 11
2.2. Hamming window 15
2.3. Espectograma de una señal de audio 16
2.4. Frecuencia vs Escala de MEL 17
2.5. Filtros de MEL 18
2.6. Banco de filtros de Mel escalados 19
2.7. Coeficientes cepstrales en las frecuencias de mel 19
2.8. Delta y Delta-Delta Coeficientes Cepstrales 20

3.1. Nodo en el árbol CF 24
3.2. Capa de entrada de red neuronal convolucional 26
3.3. Kernel en red neuronal convolucional 26
3.4. Padding en red neuronal convolucional 27
3.5. Ejemplo de shap values para una instancia 39
3.6. Importancia de características usando shap values 39
3.7. Media de los valores absolutos de los shap values 40

4.1. Arquitectura Red Neuronal 44
4.2. Matriz de confusión modelo de género 46
4.3. Características para segmentación basada en clustering 47
4.4. Proceso para extraer características de los audios 49

4.5. Proceso para segmentar las voces 49
4.6. Ejemplo de texto de cliente 51
4.7. Ejemplo de texto de cobrador 51
4.8. Shap Values para un cobrador 53
4.9. Diccionario de 3-grams 53
4.10. Proceso para clasificación de cliente y cobrador 54

5.1. Información filtrada de cada audio 55
5.2. Campo is_disbursed 56
5.3. Campo due_date 57
5.4. Campo paid_date 57
5.5. Campo paid_delayed_time 58
5.6. Múltiples llamadas debidas al mismo préstamo 58
5.7. Datos de las llamadas filtradas 59
5.8. Series de tiempo de los audios 60
5.9. Tabla con nombres de archivos 60
5.10. Tabla con loan id 61
5.11. Relaciones entre tablas 62
5.12. Características de deep feature synthesis 63
5.13. Características espectrales 64
5.14. Algunas características seleccionadas 66
5.15. Matriz de confusión del modelo 68
5.16. Curva ROC 69
5.17. Importancia de las características 70
5.18. Shap Values para modelo de comportamiento de pago 71
5.19. Efecto de la media del coeficiente 5 de Mel 71
5.20. Efecto de la varianza del coeficiente 12 de Mel 72
5.21. Promedio del coeficiente 5 de mel a través del tiempo 72
5.22. Curva de calibración del modelo 73

6.1. Pipeline de modelo final para un trabajo a futuro 77

Prefacio

Kueski es una Institución financiera en la ciudad de Guadalajara México, cuyo principal servicio es dar microprestamos a corto plazo. Gran parte del éxito de Kueski, se debe a su área de ciencia de datos, donde entre otras muchas otras cosas, crean modelos basados en datos para la prevención de fraude, aceptación de clientes, etc.

Diversas fuentes proveen de datos a Kueski para la creación de sus modelo, pero una fuente que hasta el momento tiene un nulo uso es el audio.

Kueski realiza diariamente llamadas de distinto tipo a sus clientes, una de las razones por las que se llama es la cobranza. Todos los audios son almacenados en una base de datos, pero no son utilizados para ningún modelo ni para extraer características de ningún tipo.

En este trabajo, se explora la extracción de información de las llamadas almacenadas por Kueski, y la creación de modelos que sirvan para clasificar o predecir algunas variables de interés. En general queremos responder la pregunta de: ¿Existe informacion relevante en los audios de cobranza que ayude a predecir el comportamiento de pago de los clientes?

Para responder est pregunta, se planetan varias técnicas para procesar el audio, segmentar, clasificar y extracción de características que puedan servir para la creación de variables predictoras.

El trabajo se divide de la siguiente manera:

Capítulo 1 - Introducción: Se explica la naturaleza del sonido, sus propiedades físicas, la representación digital del audio en una computadora, aplicaciones del análisis de audio, algunos conceptos importantes como la velocidad de muestreo, códecs más importantes y extensiones de almacenamiento.

Capítulo 2 - Extracción de características: Se abordan técnicas de procesamiento de audio para extraer información relevante del audio, específicamente para audios con voces. Algunos tipos de caracterísitcas son mostradas y su procedimiento de extracción.

Capítulo 3 - Técnicas de aprendizaje máquina: Técnicas supervisadas, no supervisadas, algoritmo de extracción de características, optimización de hiperparámetros, y métodos de intepretabilidad usados en este trabajo.

Capítulo 4 - Segmentación y clasificación: Se abordan las técnicas utilizadas en este trabajo para clasificación de género, segmentación de voces, y clasificación de cliente-cobrador. Se muestran los experimentos y sus resultados.

Capítulo 5 - Modelo de predicción de pago: Aquí se comienza con la etapa del filtrado de los audios de cobranza, la creación de la variable de respuesta para el comportamiento de pago, la extracción de características, selección de las mismas, el modelo, resultados, su interpretación y análisis final.

Capítulo 6 - Conclusiones y trabajo a futuro: Por último se concluye cada una de las etapas del trabajo, y se mencionan posibles ideas para trabajos a futuro.

A continuación se enuncian los objetivos del presente trabajo.

Objetivo general

- Extraer información de los audios que ayude a predecir el comportamiento de pago de los clientes, además de extraer variables predictorar que puedan servir a otros modelos de Kueski

Objetivos específicos

- Segmentar las voces de los audios
- Clasificar el género de las personas en los audios
- Clasificar los roles de las voces de los audios en cliente y cobrador
- Extraer caracteristicas del audio que puedan ser utilizadas por otros modelos existentes en Kueski para fines de cobranza
- Constuir un modelo para el comportamiento de pago en base al audio

Capítulo 1

Introducción

La ciencia de datos está cada vez más presente en diversos ramos de las actividades humanas. Muchas empresas se están volcando a utilizar este tipo de herramientas para optimizar y mejorar sus procesos, crear mejores productos o servicios, entre otros.

Parte de esta revolución es la gran cantidad de datos que a diario se registran. Estos datos pueden venir de diversas fuentes. Una fuente rica de información es el audio. El audio por su naturaleza, es más difícil de modelar. En este capítulo, se comenzará analizando la naturaleza del audio, sus características más importantes, aplicaciones actuales, su forma de representarse en sistemas digitales, y las técnicas básicas para su procesamiento.

1.1. Audio

Antes de analizar qué es el audio, se explicará qué es el sonido. El sonido consta de la propagación de ondas mecánicas a través de algún medio. Por lo general, el sonido lo percibimos a través del aire, como ondas comprimidas del mismo que viajan a una velocidad promedio de 331 m/s.

El audio es una representación analógica o digital del sonido. Podemos pensar en el sonido como una señal unidimensional temporal. Se comenzará primero explicando

un poco más sobre el sonido y sus características físicas.

1.1.1. Características físicas del sonido

El sonido se desplaza en el aire a una velocidad de 331.5 m/s cuando la tempetatura es de 0°C, una presión de una atmósfera y humedad relativa de 0 %. La velocidad del sonido a cualquier temperatura puede calcularse:

$V_s = V_0 + \beta T$

Donde:

V_0 es 331 m/s

β 0.606 $m/(S°C)$

T temperatura en grados centígrados.

El sonido queda caracterizado por su potencia acústica y su espectro de frecuencias.

Potencia. La potencia de un sonido depende de la magnitud de sus ondas. La unidad de potencia en el Sistemas Internacional es el Vatio(W). La percepción que los seres humanos tenemos de la potencia es lo que llamamos volumen, expresado en decibelios(dB).

$Potencia = 10log_{10}(\frac{W_1}{W_0})$

Donde:

$W_0 = 10^{-12} W/m^2$

El ser humano en promedio escucha sonidos entre 0 dB a 130 dB, donde este último valor equivale al comienzo del umbral del dolor.

Frecuencia Se dise que una señal es periodica, si para cierta cantidad T, $f(t) = f(t + T)$. A la cantidad T le llamamos periodo. Es decir, una señal es periodica si repite un patrón de cierto tamaño. La frecuencia de una señal periódica es la cantidad de veces que se repite dicho patrón en un tiempo dado. Por lo general, medimos la frecuencia en repeticiones por segundo. Una repetición por segundo se le llamada

Hertz. Así por ejemplo, una señal con frecuencia de 10 Hertz(HZ), se repite un mismo patrón 10 veces durante un segundo.

Como el sonido está compuesto por ondas, estás contienen diferentes frecuencias. El ser humano puede escuchar frecuencias entre los 20 y 20,000 HZ. En música, la nota **la** sobre el **do** central, tiene una frecuencia de 440 HZ.

Las voces de hombre y mujer, en promedio, tienen diferentes frecuencias a través de sus edades [2].

Tabla 1.1: Frecuencias en voces humanas

	Hombres	Mujeres
20-29	120	227
30-39	112	214
40-49	107	214
50-59	118	214
60-69	112	209
70-79	132	206
80-89	146	197

1.1.2. Aplicaciones del anális de audio

Algunos ejemplos de áreas de aplicación y de investigación actual c son [1]:

Speaker verification La idea es identificar a cierta persona a través de su voz. Normalmente se tiene en una base de datos la voz de la persona a verificar o voz objetivo, y un conjunto de voces extra. Si la voz de la persona que estamos probando, se parece más al promedio de las voces del conjunto extra que de la voz objetivo, se decide que no pasa la prueba de verificación.

Speaker clasification

Existen diferentes formas de clasificación, algunos ejemplos son: la edad, el género, personas.

Speaker segmentation

Separar voces de música, ruido, y diferentes sonidos es otra aplicación del audio. Separar diferentes voces incluso cuando éstas se sobrelapan es un problema muy difícil, conocido como el problema del cocktail.

Speaker detection

Encontrar uno o más hablantes específicos en un audio, a esto se le llama speaker dectection. Dependiendo del problema, se puede usar uno o alguna combinación de las anteriores técnicas.

Speaker tracking

Seguir a uno o varios hablantes a través de una señal de audio, a esto se refiere el speaker tracking. Es muy similar al speaker detection.

Audio a texto (Speech to text)

Una de las aplicaciones más populares, es convertir un audio a texto. Actualmente existen diversas aplicaciones que hacen esto en diferentes idiomas, siendo algunas de las más famosas la API de Google y Amazon translate.

Análisis de sentimiento

Se trata de categorizar los sentimientos que tenía la persona que da el mensaje, por lo general las clases son: positivo, negativo o neutral. Lo más común es primero convertir el audio a texto, y después utilizar algún algoritmo de análisis de sentimiento sobre el texto.

Speech synthesis

Es la otra cara de audio a texto; en este caso, se da un texto y la idea es convertir a un audio, que parezca lo más natural posible a una voz humana. En 2018, DeepMind publicó un modelo llamado Wavenet que mejora el estado del arte de hasta ese entonces.

Animal call detection

También existen sistemas que se encargan de clasificar los sonidos de ciertos animales, para determinar de a especie pertenece, como en [11], donde se clasifican especies de aves dada su llamada utilizando extracción de caracterísicas con apredizaje no supervisado

Estas son solo algunas de las recientes aplicaciones que se han estado estudiando en el procesamiento de audio. A continuación se estudiará cómo se representa el sonido en un sistema computacional para su procesamiento.

1.2. Representación digital del audio

Una señal, es una función $f(x_1, x_2, x_3, ..x_n)$ que mapea un punto $(x_1, x_2, x_3, ..., x_n) \in D(f)$ en el dominio $D(f)$, a un punto $f(x_1, x_2, x_3, ..., x_n) \in R(f)$ en el rango $R(f)$:

$$f : D(f) \to R(f)$$

$$(x_1, x_2, x_3, ..., x_n) \mapsto f(x_1, x_2, x_3, ..., x_n)$$

Una señal dependiente del tiempo, es una función $f(t)$ que mapea cualquier instante de tiempo en su dominio $D(f)$ en un punto en su rango $R(f)$, de tal manera que describe un fenómeno físico [8].

$$f : D(f) \to R(f)$$

$$t \mapsto f(t)$$

Una señal estacionaria es aquella cuyos parámetros estadísticos no cambian con el paso del tiempo. En cambio, si los parámetros cambian, se le denomina señal no estacionaria.

Para procesar una señal de audio, primero debemos representarla de alguna manera en una computadora, veremos a continuación de manera general cómo se logra tal representación.

1.2.1. Muestreo

El muestreo es el proceso por el cual se discretiza una señal, tomando muestras en diferentes instantes de tiempo. Existen diferentes técnicas de muestreo en el audio, pero la más sencilla es el muestreo periódico. En esta técnica, las muestras se toman con el mismo espaciado temporal T llamado el intervalo de muestreo. A la cantidad $F_s = \frac{1}{T}$ se le llama tasa de muestreo o velocidad de muestreo.

Teorema del muestreo. Una señal $f(t)$ que no contiene frecuencias superiores a f_c está completamente determinada por sus ordenadas en una serie de puntos espaciados por $\frac{1}{2f_c}$.

El teorema del muestreo indica que para poder muestrear una señal sin pérdida de información, debemos hacerlo a una frecuencia de al menos el doble de la mayor frecuencia de la señal original. En la figura 1.1 se muestra una señal y los diferentes puntos donde se toma una muestra.

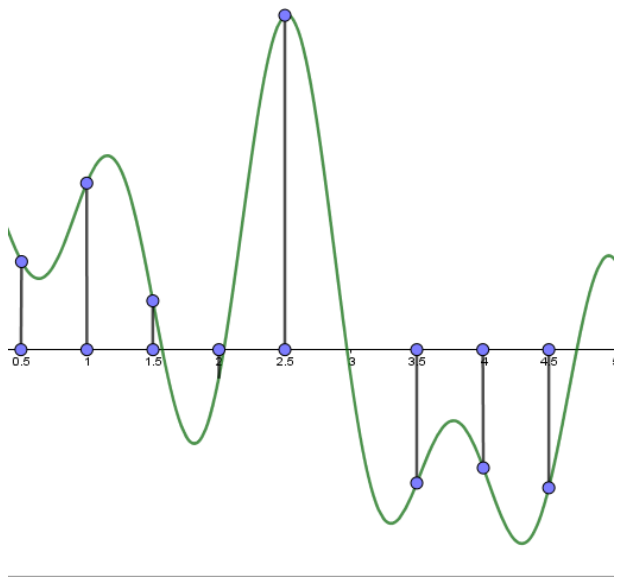

Figura 1.1: Señal muestreada

1.2.2. Cuantización del audio

Una vez discretizado el dominio (tiempo), se necesita discretizar la amplitud de la señal. Esta discretización depende de la cantidad de bytes que se utilizarán para representar la señal. Valores típicos son 8, 16, 32 bits. Por ejemplo, una señal de 32 bits podrá representar valores entre -2147483648 y 2147483647. Aunque existen otros formatos para representar el rango de valores que más adelante se explicará.

En general, los diferentes valores de una señal de audio, se normalizan en un rango entre -1 y 1. De tal manera, una señal de audio en un sistema computacional es un arreglo de valores entre -1 y 1, donde cada elemento representa un valor de la señal de audio en un instante t donde fue muestreado. En la figura 1.2, se muestra la señal original, y como queda después de cuantizarla en 11 niveles.

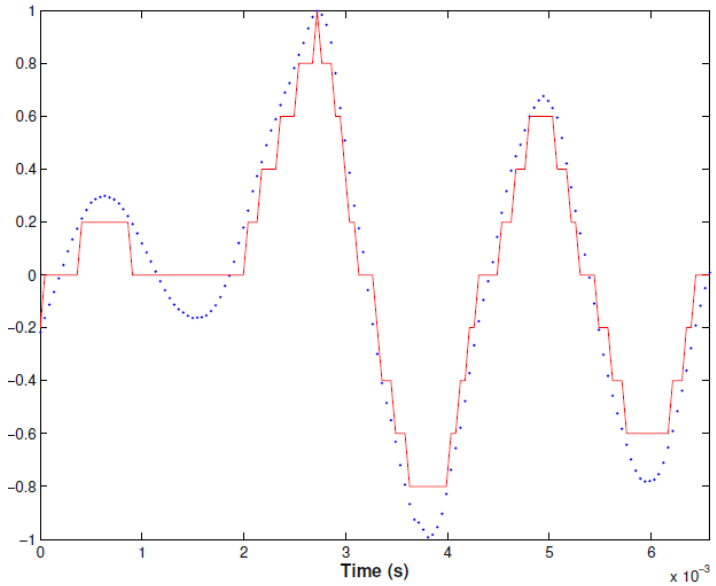

Figura 1.2: Señal cuantizada en 11 niveles

A continuación se mostrarán algunos de los tipos de codificación más comunes en el audio digital.

1.2.3. Códecs

En general, existen dos clases de codificación de audio, comprimidos y no comprimidos. Dentro de los no comprimidos, el más popular es el PCM.

PCM Por sus siglas en inglés Pulse Code Modulation, como se explicó en la sección anterior, una codificación lineal de los diferentes valores cuantizados. Existe PCM de 8, 16, 24,32 y 64 bits. Cada uno de estos puede estar en una representación con signo o sin signo y en formato little endian o big endian. Por ejemplo, un formato PCM 16 con signo tendría $2^{16} = 65536$ niveles distintos, los cuales se representarían

en un rango de $[-32768, 32767]$ y se almacenarían en 4 bytes con valores flotantes de $[-1, 1]$. La forma de almanecar cada uno de los bytes dependerá si está en formato little endian o big endian.

Existe otro formato sin compresión llamado IEEE float, que básicamente es un formato de 4 bytes, donde se codifican los niveles en valores en el rango $[-1, 1]$ según el estandar de flotantes IEEE.

Dentro de los formatos comprimidos, existen a su vez dos clases: con pérdida y sin pérdida. Cada tipo de códec utiliza diferentes algoritmos para comprimir las señales de audio. Algunos de estos códecs son:

Sin pérdida FLAC, MPEG-4,LPAC,MLP.

Con pérdida AAC, AC3,MP1, MP2, MP3, WMA.

Algunos de estos algoritmos utilizan ténicas como la transformada de Wavelet para la compresión con périda, o predicción lineal para compresión sin pérdida.

1.2.4. Contenedores

Un contenedor, es un formato en el cual se organiza el almacenamiento de diferentes fuentes de datos, como audio, video, metadatos, etc. Ejemplos son: AVI, MPG,MOV,ASF.

Los contenedores de audio dan la extensión al archivo que almanena la información digital. Existe un archivo de audio o contendor llamado WAV. En este trabajo de tesis, los audios son almancenados de origen en este formato, con un códec de audio float IEEE. En la siguiente tabla se muestra la definición de este formato.

Tabla 1.2: Definición de formato WAV float-IEEE

Campo	Longitud	Contenido
ckId	4	RIFF"
ckSize	4	$4 + 26 + 12 + (8 + M * N_c * N_s + (0 \text{ ó } 1))$
WAVEID	4	"WAVE"
ckID	4	"fmt "
cksize	4	18
wFormatTag	2	1
nChannels	2	N_c
nSamplesPerSec	4	F
nAvgBytesPerSec	4	$F * M * N_c$
nBlockAlign	2	$M * N_c$
wBitsPerSample	2	$8 * M$
cbSize	2	0
ckID	4	"fact"
cksize	4	4
dwSampleLength	4	$N_c * N_s$
ckID	4	"data"
cksize	4	$M * N_c * N_s$
datos	$M * N_c * N_s$	Los datos
pad byte	0 ó 1	Un byte si $M * N_c * N_s$ es impar

Donde:

N_s número de muestras,

N_c número de canales,

F velocidad de muestreo,

M cantidad de bytes por muestra

Capítulo 2

Extracción de características

Una señal de audio digital, es entonces, una serie de tiempo que por lo general se estandariza entre valores -1 y 1. Una señal típica de audio podemos verla en la figura 2.1.

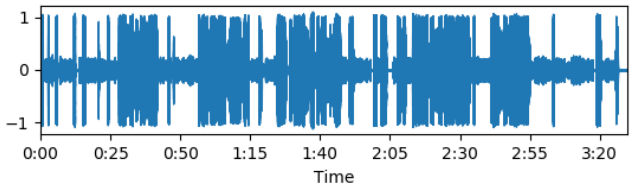

Figura 2.1: Señal de audio

En este capítulo, se aboradará el problema de extraer caracerísticas de una señal de audio y convertir de una serie de tiempo a una representación apta para un modelo de machine learning.

Existen diferentes técnicas de extracción de características de audio, en este trabajo de tesis se abordarán los llamados coeficientes ceptrales en las frecuencias de MEL, **MFCC** por sus siglas en inglés(mel frequency cepstral coefficients), que en las siguientes secciones se explicará el proceso para el cálculo de estos coeficientes.

2.1. El espectograma

Como primer paso hacia el cálculo de los MFCC, se debe calcular el espectograma de la señal de audio. El espectograma es básicamente la aplicación de la transformada de Fourier sobre la señal de audio. El resultado es una descomposición de la señal de audio en sus diferentes frecuencias. Se explicará esto con más detalle en las siguientes subsecciones.

2.1.1. Transformada de Fourier

La transformada de Fourier puede entenderse como la generalización del desarrollo en series de Fourier de una función para el caso no periódico. Una función que cumple con las condiciones de Dirichlet tiene un desarrollo en series de Fourier

Condiciones de Dirichlet. Sea una función $f(t)$ en un intervalo $-T \leq t \leq T$. Las condiciones de Dirichlet para esta función son:

1. $f(t)$ es absolutamente integrable en el intervalo $[-T, T]$
2. $f(t)$ es periódica con periodo $2T$
3. $f(t)$ es al menos de clase C^1 en el intervalo $[-T, T]$

Una función $f(t)$ que cumple con las condiciones de Dirichlet tiene una representación en series de Fourier de la forma:

$$f(t) = \sum_{n=-\infty}^{\infty} C_n e^{i\frac{n\pi t}{T}} \tag{2.1}$$

Donde:

$$C_n = \frac{1}{2T} \int_{-T}^{T} f(t) e^{i\frac{n\pi t}{T}} dt \tag{2.2}$$

Se suele hacer un cambio de variable para normalizar el periodo:

$$\hat{t} = \frac{\pi}{T} t \tag{2.3}$$

Reescribiedo (2.1) y (2.2)

$$f(t) = \sum_{n=-\infty}^{\infty} C_n e^{in\hat{t}} \tag{2.4}$$

$$C_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(t) e^{in\hat{t}} dt \tag{2.5}$$

Si tomamos el límite $T \to \infty$ y cambiamos la variable discreta n por la variable continua ω

$$f(t) = \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega \tag{2.6}$$

$$F(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \tag{2.7}$$

El par $f(t), F(\omega)$ es el par de transformada de Fourier.

Con la transformada de Fourier, dada una señal en el tiempo $f(t)$,podemos descomponerla en sus diferentes frecuencias. El único problema es que la señal de interés es una señal discreta. Existe para estos casos la versión de la transformada de Fourier discreta.

Dado un conjunto finito de muestras, queremos mapearlo a un conjunto finito de frecuencias. Sean N muestras, podemos escribir los instantes de cada muestra como $n = \{0, 1, ..., N-1\}$.

Usaremos también N como la resolución en las frecuencias que obtendremos. Por el teorema del muestreo, podremos recuperar las frecuencias:

$$f_k = \frac{k}{NT}, k = \{0, 1, ..., N-1\} \tag{2.8}$$

Podemos escribir el par de transformadas de Fourier en su versión discreta:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}kn} \tag{2.9}$$

$$X_n = \frac{1}{N} \sum_{n=0}^{N-1} X_k e^{\frac{2\pi i}{N}kn} \tag{2.10}$$

Estos calculos tienen una complejidad computacional de $O(N^2)$, por lo que se han desarrollado técnicas para hacer más eficiente este cálculo. Existen diferentes variantes de algoritmos, que en general se llaman **La transformada Rápida de Fourier** que consiste básicamente en dividir el problema en $2 \times \frac{N}{2}$ problemas y seguir dividiendo hasta que de un lado del árbol solo queda 1 problema. Con esto se logra conseguir una complejidad de $O(NlogN)$

2.1.2. Audio framming

Una señal de audio es no estacionaria, pero si tomamos pequeños intervalos en la señal, podemos esperar que estos sí sean estacionarios. Por ello, vamos a calcular la transformada de Fourier sobre pequeños intervalos de la señal original. Al proceso de tomar estos intervalos se le conoce como framming, y a los intervalos frames.

Para ello se debe elegir un tamaño de ventana N y un tamaño de deslizamiento D, de tal manera que $D < N$ por lo que cada frame tendrá un pequeño solapamiento con el anterior y el siguiente. Usualmente para aplcaciones de voz, se toma como tamaño de ventana N entre 20 y 30 ms y un tamaño de solapamiento D de 10 ms.

2.1.3. Windowing

Queremos calcular la transformada de Fourier sobre cada uno de los frames. El problema es que la transformada de Fourier espera que tengamos un número de periodos enteros en cada frame, visto de otra manera, la transformada de Fourier ve a cada señal de forma circular, como si al final de la señal continuara con el principio. Desafortunadamente esto no es cierto, y si pudieramos “pegar” el final de cada frame con el principio, la mayor parte de las veces veríamos saltos, que conducen a

inducir frecuencias erroneas en la transformada de Fourier. Por ello, una forma de mitigar esto es filtrar la señal de tal manera, que los extremos de la señal se atenuen. Este proceso se llama windowing y consta de multiplicar cada frame por una ventana, que en general es la función Hamming, aunque otras alternativas también se usan.

En la figura 2.2 podemos ver la forma de la función Hamming.

$$w(n) = 0.54 - 0.46cos(\frac{2\pi n}{N-1}) \tag{2.11}$$

Figura 2.2: Hamming window

Sea ${}_lx_n$ la n-ésima muestra del l-esimo frame obtenido en el proceso de framming,

$${}_l\hat{x}_n = {}_lx_n w(n) \tag{2.12}$$

Ahora, se aplica la transformada de Fourier sobre cada frame ya filtrado:

$${}_lX_k = \sum_{n=0}^{N-1} {}_l\hat{x}_n e^{-i\frac{2\pi kn}{N}} \tag{2.13}$$

donde $k = \{0, 1, ..., N-1\}$ son los índices de las frecuencias que se explicó en la seccción 2.1.1.

El resultado es una matriz, donde las columnas representan los diferentes frames, y las filas las frecuencias. Esta matriz tiene entradas en $\mathbb{C}$. Para cuestión de extracción de características, tomamos el módulo de cada entrada:

$$|{}_lX_k| = \sqrt{Re({}_lX_k)^2 + Im({}_lX_k)^2} \tag{2.14}$$

En la figura 2.3, podemos ver el espectrograma de un audio. El eje x es el tiempo, el eje y la frecuencia, y el color denota la potencia. Así zonas negras significan la ausencia de cierta frecuencia en determinado tiempo, mientras colores mas cercanos al amarillo y blando significan potencias altas.

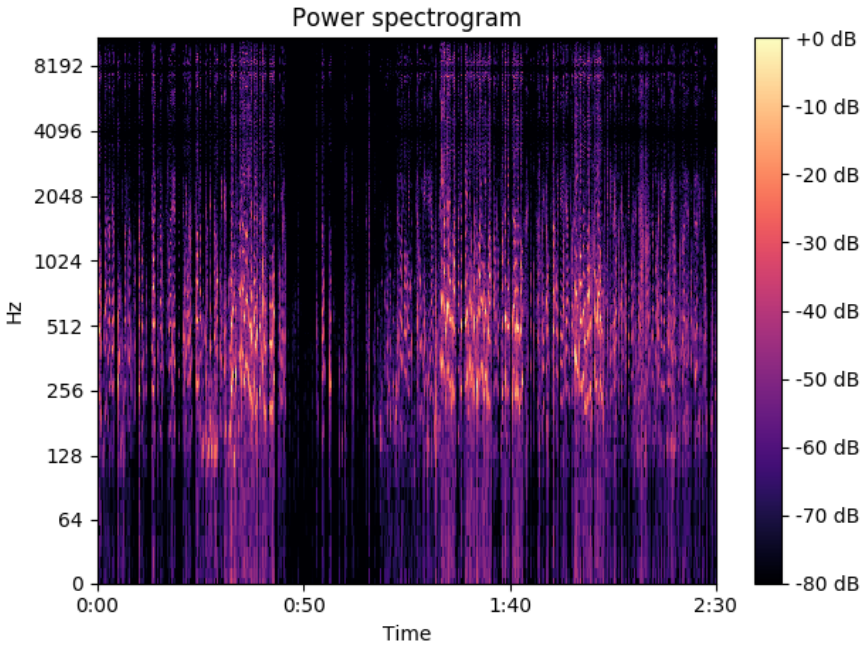

Figura 2.3: Espectograma de una señal de audio

2.2. Coeficientes de Mel

Es posible usar el espectograma como una caracterización del audio para un modelo de machine learning. En el caso de este trabajo, el audio es producido por la voz humana, y para ello existe un procedimiento especial para caracterizar este tipo de sonido, el cual enfatiza las características de las voces y elimina otro tipo de posibles ruidos.

2.2.1. Escala de Mel

Nuestro oído humano no percibe de manera lineal las frecuencias del sonido. Por ejemplo, la diferencia entre un sonido de 100 HZ y uno de 200 HZ, nos parecerá mas

grande que entre un sonido de 1100hz y otro de 120 0HZ. Aunque la percepción humana dirá que la diferencia en las frecuencias es mucho menor entre los sonidos de 1100 y 1200 HZ, la diferencia en ambos casos es 100 HZ. La escala de Mel se desarrolló en los años 1940's, con experimentación en la percepción de las frecuencias en humanos. Se desarrolló una escala que linealiza la percepción humana de la siguiente manera:

$$m = 2595log_{10}(1 + \frac{f}{700}) \tag{2.15}$$

$$f = 700(10^{m/2595} - 1) \tag{2.16}$$

En esta escala, 1000 HZ equivalen a 1000 Mels, la relación no linean entre mels y hz podemos vera en la figura 2.4

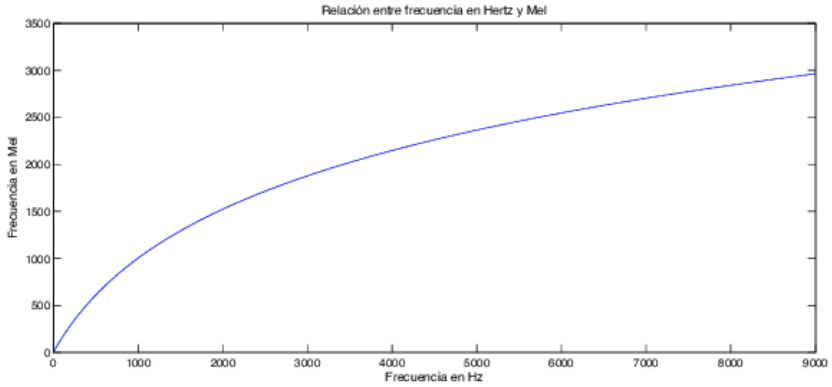

Figura 2.4: Frecuencia vs Escala de MEL

La idea es convertir las N frecuencias de nuestro espectrograma, a unas cuantas m bandas que estén equiespaciadas según nuestra percepción.

2.2.2. Bancos de filtros de Mel

En la práctica, se toman alrededor de 24 bandas, separadas por 100 Mels cada una. Para cada banda se toma un filtro triangular con altura máxima de 1, justo en la mitad de su banda como se aprecia en la figura 2.5.

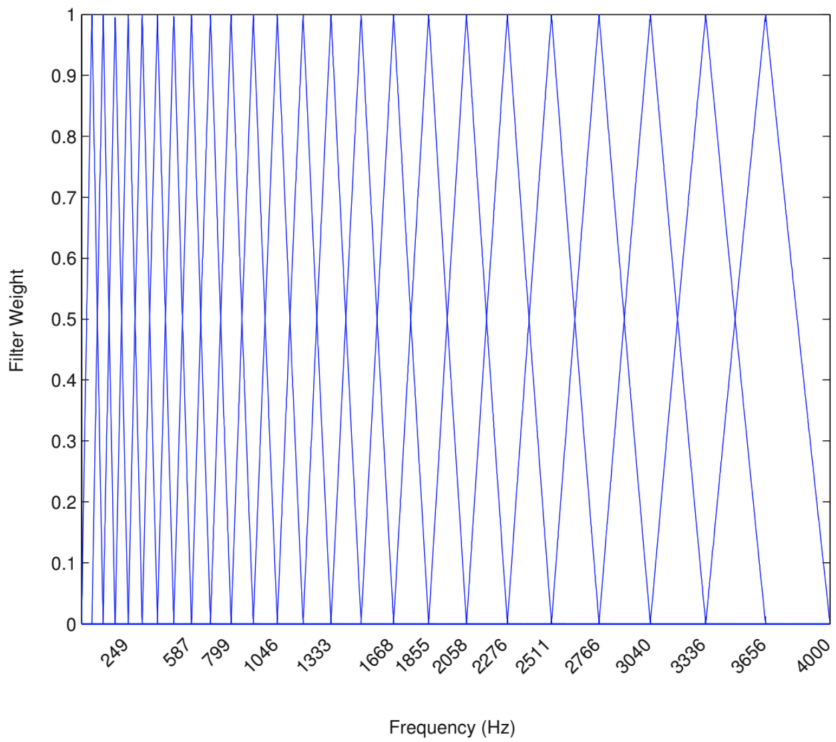

Figura 2.5: Filtros de MEL

$$|_l\hat{X}_m| = M_{m,k}|_lX_k| \tag{2.17}$$

$M_{m,k}$ es una matriz con tantas filas como filtros de MEL, y tantas columnas como columnas tenga el espectograma. Cada fila representa un filtro de MEL.

Además de que la frecuencia es percibida de manera no lineal, también sucede lo mismo con la intensidad de los sonidos. Por lo tanto, un último paso es necesario, linealizar en amplitud:

$$_lC_m = log(|_l\hat{X}_m|)^2 \tag{2.18}$$

A este resultado se le llama **banco de filtros de mel escalados**(figura 2.6), algunos modelos de machine learnig usan este resultado para caracterizar el sonido. En este trabajo, estaremos usando estos resultados junto con otra caracterización que resulta de aplicar un últmo proceso.

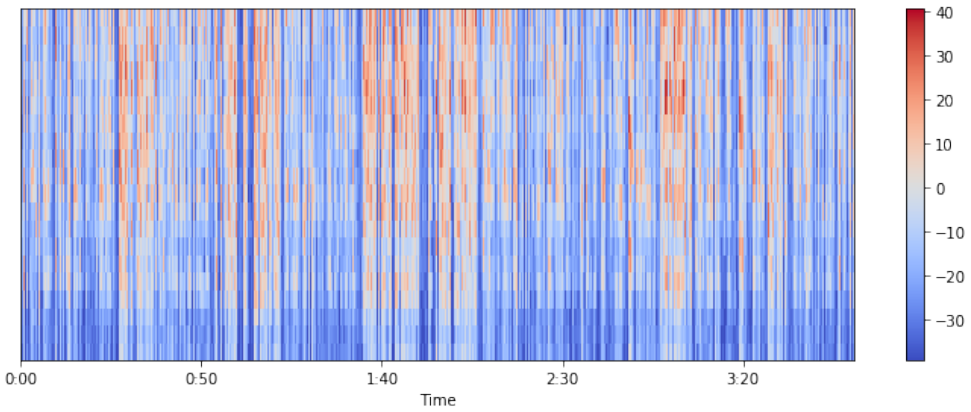

Figura 2.6: Banco de filtros de Mel escalados

2.2.3. Coeficientes cepstrales en las frecuencias de Mel

Transformada de coseno discreta inversa

Cómo último paso, podemos calcular la trasformar de Fourier inversa sobre los bancos de filtros de mel escalados, pero el resultados estaría en el dominio complejo. Podemos calcular entonces la transformada de Fourier de coseno inversa, la cual, nos devuelve resultados en el dominio real. En la figura 2.7 se muestran los MFCC de un audio.

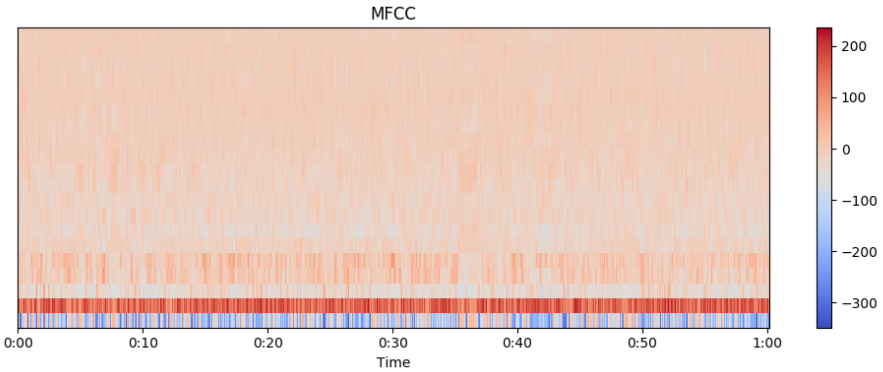

Figura 2.7: Coeficientes cepstrales en las frecuencias de mel

$$ {}_lC_d \sum_{m=0}^{M-1} a_{ml} C_m cos(\frac{\pi(2d+1)m}{2M}) \tag{2.19} $$

$$\begin{cases} \frac{1}{N} & k = 0 \\ \frac{2}{N} & k \geq 0 \end{cases} \tag{2.20}$$

2.2.4. Deltas

Para extraer más información sobre los coeficientes de MEL, se calculan la primera y segunda derivadas, llamadas, deltas de orden uno y dos, o Delta y Delta-Delta Coeficientes Ceptrales respectivamente. Para calcular estas derivadas se utiliza el método del filtro de Savitsky-Golay. [9]

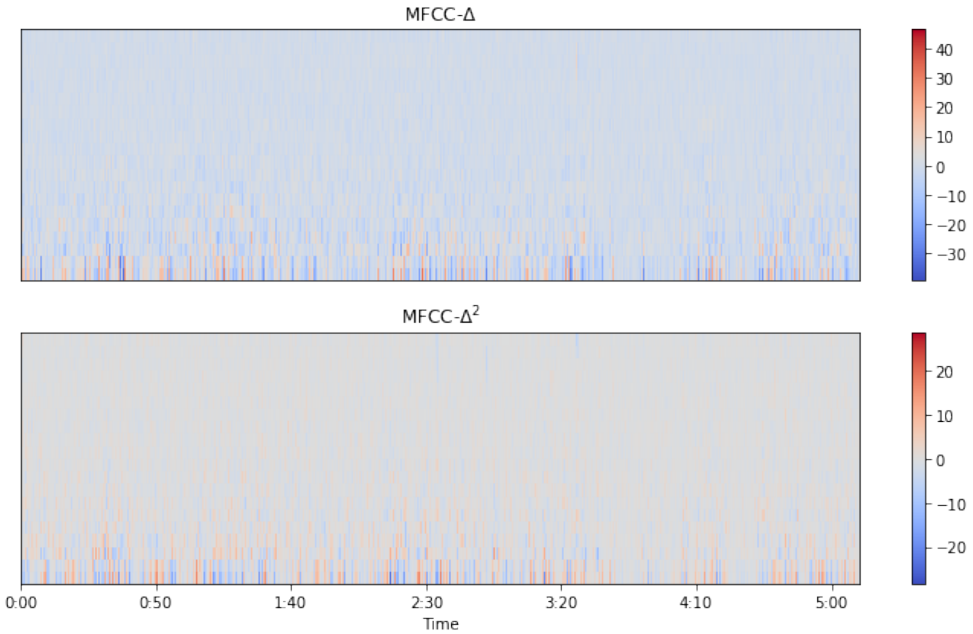

Figura 2.8: Delta y Delta-Delta Coeficientes Cepstrales

2.2.5. Filtro de Savitzky–Golay

Las deltas de los coeficientes de Mel, como se mencionó en la sección anterior, son calculados con un algoritmo llamado Savitzky-Golay. Antes de entrar a sus detalles, recordemos que podemos pensar en un filtro de una señal f, como una función que mapea cada valor f_i a otro g_i. Uno de los tipos de filtros mas sencillos en los que podemos pensar son del tipo:

$$g_i = \sum_{n=-n_L}^{n_R} c_n f_{i+n} \tag{2.21}$$

Donde n_L es el número de puntos a la izquierda que se usarán para calcular el valor de g_i y n_R los valores a la derecha. Si hacemos $n_L = n_R$ y $c_n = \frac{1}{n_R+n_L+1}$, obtenemos un filtro que reemplaza cada valor de la señal por su promedio en una vecindad simétrica de radio n_R

Savitzky-Golay en cambio, en cada punto f_i ajusta un polinomio típicamente de grado 2 ó 4 con mínimos cuadrados en todos los puntos $n_L + n_R + 1$ de la ventana, y hace g_i como el valor del polinomio en el punto i. Ningún otro valor del polinomio es usado. Para calcular g_{i+1} se vuelve a hacer una regresión con mínimos cudrados y se hace g_{i+1} como el valor del polinomio resultante en la posición $i + 1$.

Este proceso puede ser laborioso, pero existe una manera de calcular coeficiente c_n, de tal manera que Savitzky-Golay se convierte en un filtro del tipo de la ecuación (2.21)

Veamos como se calculan estos coeficientes c_n, supongamos que queremos obtener el valor g_0, por lo tanto, debemos ajustar un polinomio de grado M en el punto i, este polinomio será: $a_0 + a_1 i + a_2 i^2 + ... + a_M i^M$. El valor g_0 es el valor del polinomio cuando i = 0, es decir $g_0 = a_0$ De forma matricial:

$$A_{ij} = i^j, i = -n_L ... n_R, j = 0...M \tag{2.22}$$

Mínimos cuadrados:

$$A^T A a = A^T f \tag{2.23}$$

Además:

$$\{A^T A\}_{ij} = \sum_{k=-n_L}^{n_R} A_{ki} A_{kj} = \sum_{k=-n_L}^{n_R} k^{i+j} \tag{2.24}$$

$$\{A^T f\}_j = \sum_{k=-n_L}^{n_R} A_{kj} f_k = \sum_{k=-n_L}^{n_R} k^j f_k \tag{2.25}$$

c_n es la componente a_0 cuando f se reemplaza por el vector unitario e_n

Para fines de suavizado se obtienen las componentes a_0, pero para fines de cálculo de la derivada de orden m se utiliza la componente a_m

En la siguiente tabla se muestran los coeficientes para calcular derivadas de grado 1 con diferentes tamaños de ventana y grados de polinomio.

Tabla 2.1: Coeficientes de Savitzky–Golay

Grado del polinomio	1 ó 2				3 ó 4		
Tamaño de ventana	3	5	7	9	5	7	9
-4				-4			86
-3			-3	-3		22	-142
-2		-2	-2	-2	1	-67	-193
-1	-1	-1	-1	-1	-8	-58	-126
0	0	0	0	0	0	0	0
1	1	1	1	1	8	58	126
2		2	2	2	-1	67	193
3			3	3		-22	142
4				4			-86

Capítulo 3

Técnicas de aprendizaje máquina

En este capítulo, se desarrollarán las técnicas de aprendizaje maquina utilizadas en el presente trabajo. Se incluyen técnicas de aprendizaje supervisado y no supervisado, optimización de hiperparámetros, métodos de interpretabilidad de modelos, y un algoritmo de construcción de características.

3.1. Birch

Birch es un método de aprendizaje máquina no supervisado, su nombre proviene de la abreviación de **Balanced Iterative Reducing and Clustering using Hierarchies**[12]. Su funcionamiento básico es el siguiente.

Un Clustering Feature, abreviado CF, es una tripleta:

$$CF = (N, \vec{LS}, \vec{SS}) \tag{3.1}$$

Donde dado un conjunto de puntos en $\mathbb{R}^d$, N es la cardinalidad del conjunto,

$$\vec{LS} = \sum_{i=0}^{N} \vec{X}_i \tag{3.2}$$

$$\vec{SS} = \sum_{i=0}^{N} \vec{X}_i \circ \vec{X}_i \tag{3.3}$$

Donde $\circ$ es el producto Hadamard y $\vec{X}_i, \vec{LS}, \vec{SS} \in \mathbb{R}^d$

Cada CF tiene las siguientes medidas:

Centroide:

$$\vec{C} = \frac{\vec{LS}}{N} \tag{3.4}$$

Radio:

$$R = \sqrt{\frac{|\vec{LS}|^2 - 2\vec{LS} \cdot \vec{C}}{N} + |\vec{C}|^2} \tag{3.5}$$

Distancia entre clústers $CF_1 = [N_1, \vec{LS}_1, \vec{SS}_1]$ y $CF_1 = [N_2, \vec{LS}_2, \vec{SS}_2]$:

$$D = \sqrt{\frac{N_1 \cdot \vec{SS}_2 + N_2 \cdot \vec{SS}_1 - 2\vec{LS}_1 \cdot \vec{LS}_2}{N_1 \cdot N_2}} \tag{3.6}$$

El algoritmo hace uso de un árbol, donde cada nodo contiene una lista de pares $[CF_i, child_i]$ donde CF_i es un Clustering feature, y $child_i$ es un puntero a su nodo hijo. El tamaño de las listas de los nodos está limitada por un hipermarámetro B llamado *Branching Factor*.

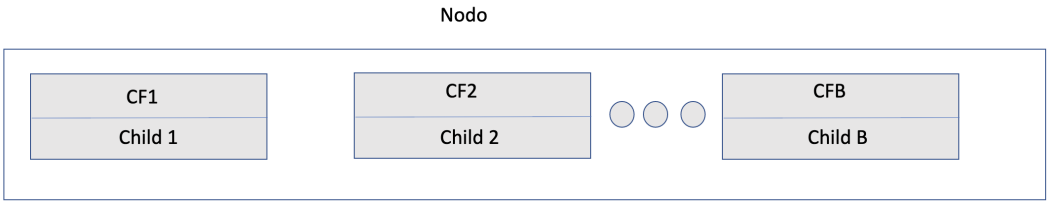

Figura 3.1: Nodo en el árbol CF

Se itera sobre todos los puntos del conjunto de datos, y se inserta uno a la vez en el árbol. Cuando se inserta un nuevo dato en el árbol, este comienza por la raíz y se desciende siempre buscando al CF más cercano según la distancia (3.6). Una vez llegado a un nodo hoja, se trata de unir este punto al CF, si el nuevo CF tiene un radio menor que un hiperparámetro T llamado *Threshold*, entonces el CF absorve el nuevo punto, en caso contrario, se crea un nuevo CF en el nodo, este CF solo contendrá el nuevo punto. Si al crear este nuevo CF, el tamaño en la lista del nodo excede el valor B, entonces el nodo se parte en dos, y se distribuyen todos los CF en los nuevos dos nodos. La distribución se hace tomando los dos CF mas alejados, e insertándolos uno en cada uno de los nuevos nodos, y el resto de CF se reparten según estén más cerca de estos CF según (3.6). Cuando se parte un nodo en dos, es necesario crear una nueva entrada en la lista del nodo padre, la creación de la nueva entrada puede provocar que este nodo también se parta y así sucesivamente. Cuando el nodo raíz se parte, el árbol gana un nivel en profundidad.

Una vez que se construye el árbol, se utiliza un método de clustering como K-means sobre las hojas, tomando los centroides de estos como puntos. El resultado de este cluster es el resultado de Birch.

3.2. Redes Neuronales Convolucionales

Las redes neuronales convolusionales, históricamente nacen para la clasificación de imágenes, y es hasta el momento, la aplicación mas popular de ellas. Desde hace algunos años, se ha comenzado a explorar este tipo de arquitecturas para aplicaciones en audio. La idea es tomar el espectograma, los coeficientes de Mel, los filtros escalados de Mel, o cualesquiera otras características en forma de matriz, y tratarlo como imágenes [7].

Para una red neuronal convolucional, conviene pensar en la entrada como una matriz de datos en lugar de un vector. Por ejemplo, si se tiene una imagen de 28x28, en

una red neuronal tradicional, la entrada es un vector de 784, en una red convolucional, en cambio, la entrada es una matriz de 28 x 28.

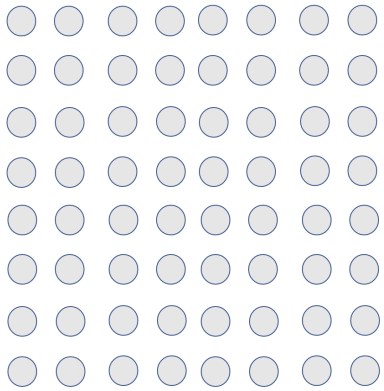

Figura 3.2: Capa de entrada de red neuronal convolucional

En estas redes, no todas las neuronas de una capa se conectan con la siguiente, en lugar de ello, se define un tamaño de kernel, el cual es una submatriz, de tal manera que solo un grupo de neuronas conectan con alguna neurona en la siguiente capa. En la figura 3.3, podemos ver un kernel de tamaño 2x2 en una capa. Las 4 neuronas resaltadas,conectan con una neurona en la siguiente capa.

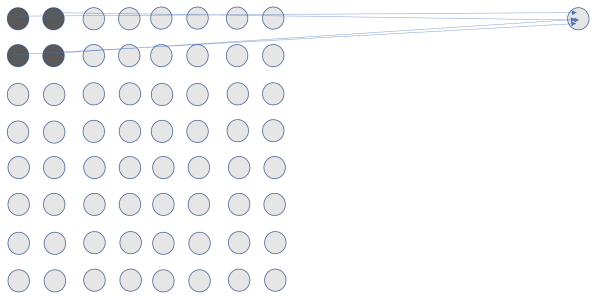

Figura 3.3: Kernel en red neuronal convolucional

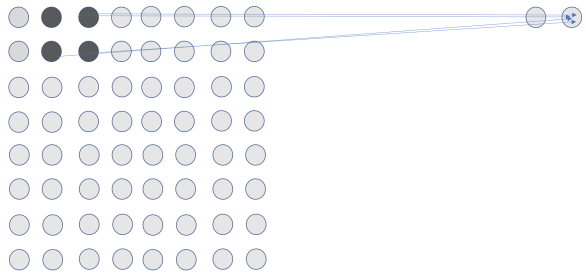

Figura 3.4: Padding en red neuronal convolucional

En la figura 3.4, podemos ver que el kernel se desplaza una posición, esta distancia es hiperparámetro de la arquitectura de la red y puede variar en distintos valores. A este tamaño de paso en el kernel se le suele llamar **padding**.

En nuestro ejemplo con un kernel de tamaño 2x2 y padding de 1. Cada neurona en la siguiente capa oculta, está conectada con 4 neuronas en la capa anterior. Cada una de estas 4 neuronas tendrá un peso en su conexión, y la neurona tendrá un bias. Otra gran diferencia con las redes neuronales totalmente conectadas, es que estos 4 pesos y el bías, serán iguales para todas las neuronas en la capa oculta, es decir, solo se calcularán 2x2 pesos y un bías.

El resultado en la capa oculta se le llama **feature map**. Se suele utilizar diferentes kernels en una capa para generar diferentes feartures maps.

Otra característca en las redes convolucionales, son las capas de pooling. En estas capas, se toma de nuevo una submatriz de la capa de neuronas, pero esta vez se aplica una operación que resume a todas las neuronas, y el resultado se deposita en la siguiente neurona de la siguiente capa. Por ejemplo, una operación común es el max-pooling, donde se toma el conjunto de neuronas dado por el tamaño de kernel, se calcula el máximo valor de todas ellas, y se pasa a la siguiente capa.

Por último, se suele convertir las últimas capas de nuevo a un vector, y construir a partir de esa capa una red neuronal totalmente conectada. Se espera que las capas

convolucionales se encarguen de extraer las caracteríticas importantes del problema, y las últimas capas totalmente conectadas hagan la clasificación.

3.3. XGBoost

XGBoost,es una técnica de aprendizaje máquina supervisado,que ha sido muy popular en los últimos años dada la velocidad de entrenamiento y los buenos resultados que obtiene. XGBoost,es una técnica de ensamble de árboles, donde el resultado final es el concenso de todos los árbolesde decisión que crea [13].

Sea un conjunto de datos con n instancias, y m características, un modelo de ensamble de árboles que usa K modelos para predecir el valor de la instancia x_i es:

$$\hat{y}_i = \phi(X_i) = \sum_{k=1}^{K} f_k(X_i) \tag{3.7}$$

Donde f_k es un árbol de regresión.

La función a minizar en el entrenamiento del XGBoost es:

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{3.8}$$

Con:

$$\Omega(f) = \psi T + \frac{1}{2}\lambda ||\omega||^2 \tag{3.9}$$

Donde l es una función de costo que mide la diferencia entre la predicción $\hat{y}_i$ y el valor real y_i. El término Ω penaliza la complejidad del modelo. T es el número de hojas en los árboles.

Esta función contiene funciones como parámetros y no puede optimizarse por métodos tradicionales. En lugar de ello, el entrenamiento se hace de manera aditiva, agregando un árbol en cada iteración. Así, en la iteración t, sea $\hat{y}_i{}^{(}t)$ la predicción de

la i-ésima instancia en la iteración t, se necesita agregar un árbol f_t para minimizar:

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(X_i)) + \Omega(f_t) \tag{3.10}$$

De manera voraz se va añadiendo el árbol que mejor minimiza la función de costo.

Se puede aproximar(3.10) utilizando el teorema de taylor por un modelo cuadrático:

$$L^{(t)} \approx \sum_{i=1}^{n} [l(y_i, \hat{y_i}^{(t-1)}) + g_i f_t(X_i) + \frac{1}{2} h_i f_t^2(X_i)] + \Omega(f_t) \tag{3.11}$$

donde $g_i = \partial_{\hat{y}} l(y_i, \hat{y}^{t-1})$ y $h_i = \partial_{\hat{y}}^2 l(y_i, \hat{y}^{t-1})$

Quitando los términos constantes:

$$\tilde{L}^{(t)} = \sum_{i=1}^{n} [g_i f_t(X_i) + \frac{1}{2} h_i f_t^2(X_i)] + \Omega(f_t) \tag{3.12}$$

Expandiendo $\Omega(f_t)$

$$\tilde{L}^{(t)} = \sum_{i=1}^{n} [g_i f_t(X_i) + \frac{1}{2} h_i f_t^2(X_i)] + \psi T + \frac{1}{2} \lambda \sum_{j=1}^{T} \omega_j^2 \tag{3.13}$$

Luego, en lugar de recorrer por instancias, recorremos por el conjunto I_j donde I_j son los índices de las instancias que van a dar a la hoha j

$$\sum_{j=1}^{T} [(\sum_{i \in I_j} g_i)\omega_j + \frac{1}{2}(\sum_{i \in I_j + \lambda})w_j^2] + \psi T \tag{3.14}$$

Así, los pesos óptimos son:

$$\omega_j = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{3.15}$$

Sustituyendo 3.15 en 3.14, podemos tener un valor score de la calidad del árbol:

$$-\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T \tag{3.16}$$

A continuación se explicarán brevemente los diferentes hiperparámetros de XGBoost.

- *eta*: Es análogo al tamaño de paso.
- *min_child_weight*: La suma mínima de todos los pesos de todas las observaciones en un hijo.
- *max_depth*: Máximo profunidad de los árboles.
- *max_leaf_nodes*: Número máximo de hojas en un árbol.
- *gamma*: Un nodo se parte solo cuando se logra una reducción de al menos *gamma* en la función de pérdida.
- *subsample*: Proporción de la muesta usado en cada árbol.
- *colsample_bytree*: Proporción de columnas(características) usadas en cada árbol.
- *alpha*: Regularización L1.
- *lambda*: Regularización L2.

3.4. Deep Features Synthesis

En el año 2015, se publicó el artículo: *Deep Feature Synthesis: Towards Automating Data Science Endeavors*. El artículo contiene un algoritmo para la creación de características dado un conjunto de tablas relacionadas [5]. La descripción del algoritmo es la siguiente:

Se define a una tabla de datos como una **entidad**. Un conjunto de datos está formado por K entidades. Sea entonces un conjunto de datos formado por las entidades

$E^1, E^2, ..., E^k$ y sin pérdida de generalidad, supongamos que cada entidad contiene J características(columnas en las tablas). Denotamos como $x^k_{i,j}$ como el valor de la i-ésima instancia en su característica j de la entidad k.

Definimos una de las entidades como la entidad objetivo (entity target). Sobre esta entidad se calcularán las diferentes características. Existen 3 tipos de características.

La entidad objetivo, puede estar relacionada de dos maneras con el resto de las tablas.

Relación hacia adelante: Se trata de cuando una instancia de una entidad está relacionada con una sola instancia de otra entidad. En el artículo original, se menciona de ejemplo la relación entre dos tablas, donde en una tabla se encuentran órdenes de compra, donde cada orden de compra está relacionada con un solo comprador en otra tabla de compradores.

Relación hacia atrás: Esta relación sucede cuando una instancia de una entidad k está relacionada con todas las intancias de otra entidad que tienen relación directa con la entidad k. En el mismo ejemplo de órdenes y compra, la relación desde compradores a órdenes de compra es una relación hacia atrás, ya que un comprador puede tener varias órdenes de compra.

El algoritmo deep feature synthesis, calcula tres tipos de características diferentes:

Características de la entidad*(efeat)*: Son características que se calculan utilizando elementos solamente de la misma tabla. Pueden ser de la forma:

$$x'_{i.j} = efeat(x_{i,j}) \tag{3.17}$$

Es decir, para su cálculo solo se necesita el valor de $x_{i,j}$, o también de la forma

$$x'_{i.j} = efeat(x_{1,j}, x_{2,j}, ..., x_{i,j}) \tag{3.18}$$

Como por ejemplo, el cálculo de percentiles.

Caractetísticas directas*(dfeat)*: Son características usando la relación directa. En estos casos, las características de una entidad pasan a ser caracteríticas de la entidad objetivo.

Características relacionales*(rfeat)*: Estas se calculan utilizando relaciones hacia atrás. Se colectan todas las instancias con las que se tiene una relación hacia atrás, y se aplica una función a este conjunto.

Algunas características de este tipo son, máximo, mínimo, promedio, varianza, curtosis, etc.

El algoritmo deep feature synthesis se muestra a continuación.

Algorithm 1 Deep Feature Synthesis

```
procedure MAKE_FEATURES(E^i, E^{1,...,M}, E_V)
    E_V = E_V ∪ E_i
    E_B= BACKWARD(E^i, E^{1,...,M})
    E_F= FOWARD(E^i, E^{1,...,M})
    for E^j ∈ E_B do
        MAKE_FEATURES(E^j, E^{1,...,M}, E_V)
        F^j = F^j ∪ RFEAT(E^i, E^j)
    for E^j ∈ E_F do
        if E_j ∈ E_F then
            Continue
        MAKE_FEATURES(E^j, E^{1,...,M}, E_V)
        F^i = F^i ∪ DFEAT(E^i, E^j)
```

El algoritmo recibe la entidad objetivo, el conjunto de todas las entidades, y un conjunto E_V(en la primera llamada vacío) con todas las entidades ya visitadas.

La primera línea de código muestra el prototipo de la función. La segunda línea hace una unión con el conjunto de ya visitados, y la entidad objetivo en la llamada. La tercera línea recolecta todas las entidades con las que la entidad objetivo tiene relación hacia atrás y los pone en el conjunto E_B. La línea cuatro hace lo mismo pero para las entidades con las que tiene relación hacia adelante y las pone en E_F. Después se itera sobre sobre E_B y para cada entidad se vuelve a llamar la función con parámetro la entidad objetivo la entidad en la que se lleva la iteración. Se tiene un conjunto con todas las relaciones hacia atrás que tiene la entidad E^j(entidad en la iteración actual) con la E_i(entidad objetivo en la llamada a la función). En la línea 8 se itera sobre E_F y si las entidades aun no son visitadas (linea 9) se invoca de manera recursiva el algoritmo al igual que en el ciclo anterior. También se guarda un conjunto con las relaciones directas entre la entidad objetivo y las iteradas.

No se muestra en el pseudocódigo, pero un parámetro de profundidad(depth) controla la profundidad de la recursión. Entre más grande el parámetro depth, mayor cantidad de características calculadas(si existen las relaciones necesarias).

3.5. Bayesian Tunning and Bandits

Algunos algoritmos de aprendizaje máquina, como redes neuronales, métodos de ensamble, máquinas de soporte vectorial, etc, tienen parámetros que no se modifican con el entrenamiento y son fijados de antemano. Estos parámetros se le conocen como hiperparámetros, y dependiendo de sus valores es muchas veces la calidad del modelo final. Ajustar estos valores es una tarea que no siempre es trivial, y que muchas veces depende de la experiencia humana. Existen diferentes técnicas para ajustar estos parámetros de manera automática. Una de ellas es usada en este trabajo y se basa en técnicas bayesianas y de conceptos de reinforcement learning como los bandits.

Grid search o random search, son técnicas populares para a optimización de hiperarámetros en modelos de aprendizaje máquina. Ambas técnicas exploran el espacio

de búsqueda de manera no informada. La optimización bayesiana, en cambio, hace uso de la información en cada iteración para aumentar la probabilidad de mejorar en las siguientes iteraciones [4].

Dado nuestro modelo M con hiperparámetros x, y una medida de calidad del modelo P, queremos hacer:

$$\arg\max_{x} P(M) \tag{3.19}$$

Existen dos aspectos importantes en la optimización bayesiana, primero, una función f llamada modelo subrogado que simulará a $P(M)$, y una funcion de adquisición α.

Algorithm 2 Optimización Bayesiana

```
procedure Bayesian_optimization
    for n=1,2,...,maxiter do
        Selecciona x_{t+1} al optimizar una función de adquisición α
        Consulta función objetivo para obtener y_{t+1}
        Actualiza el conocimiento D_{t+1} = {D_t, (x_{t+1}, y_{t+1})}
        Actualiza modelo estadístico
=0
```

Es común utilizar procesos gaussianos como modelo subrogado. Dado un conjunto de puntos en el espacio de hiperparámetros $x_1, x_2, .., x_k$ y sus respectivas evaluaciones en el modelo subrogado $f(x_1), f(x_2), ..., f(x_k)$, suponemos que este vector proviene de una distribución multivariable normal con un vector de medias particular y matriz de covarianza. Este vector de medias y matriz de covarianzas se eligen através de una función de medias y un kernel. Por lo general, la función de medias en una función constante tal que $\mu_0(x) = \mu$ y el kernel:

$$\Sigma_0(x, y) = \lambda e^{-||x-y||^2} \tag{3.20}$$

Donde λ es un parámetro escogido previamente.

Una vez que se tiene el vector y la matriz, entones el vector $[f(x_1), f(x_2), ..., f(x_k)]$ se modela:

$$[f(x_1), f(x_2), ..., f(x_k)] \sim Normal(\mu_0(x_{1:k}, \Sigma_0(x_{1:k}, x_{1:k}))) \tag{3.21}$$

Supongamos ahora que tenemos un modelo subrogado $f(x_{1:n})$ y observamos un nuevo punto $x_k = x_{n+1}$ y queremos inferir el valor $f(x_k)$. Vamos a calcular la distribución condicional de $f(x_k)$ dado $f(x_{1:n})$

$$f(x_k)|f(x_{1:n}) \sim Normal(\mu_n(x_k),^2 (x_k)) \tag{3.22}$$

Con

$$\mu_n(x_k) = \Sigma_0(x, x_{1:n})_0(x_{1:n}, x_{1:n})^{-1}(f(x_{1:n}) - \mu_0(x_{1:n})) + \mu_0(x) \tag{3.23}$$

$$\sigma^2(x) = \Sigma_0(x, x) - \Sigma(x, x_{1:n})\Sigma(x_{1:n}, x_{1:n})^{-1}\Sigma(x_{1:n}, x) \tag{3.24}$$

A esta distribución condicional se le conoce como la distribución de probabilidad a posteriori.

La función de adquisición, es una función que elige el próximo punto x_k de manera inteligente. Existen diferentes técnicas basadas en bandits de reinforcement learning, como Thompson Sampling.

3.6. Shap Values

No todos los modelos son igualmente interpretables. Modelos como regresión lineal, regresión logística, árboles de decisión, son fáciles de interpretar, es decir, podemos entender porqué clasifican una instancia tal cual lo hacen. Pero existen modelos que son como cajas negras, en los cuales es difícil entender que está sucediendo con el modelo al momento de clasificar una instancia. Ejemplos de estos modelos son las redes neuronales o los métodos de ensamble como los random forest.

Debido a ello, diferentes técnicas de interpretación han sido desarrolladas, y estas técnicas se han clasificado en grupos según su aproximación a resolver el problema. Uno de estos grupos son las técnicas locales, donde se trata de dar una explicación a la respuesta del modelo para una instancia en particular. Una de las técnicas más famosas es LIME(Local Interpretable Model-agnostic Explanations). En 2017, un artículo llamado *A Unified Approach to Interpreting Model Predictions* presenta la técnica de Shap Values, que es una mejora en LIME, y demuestrá ser la única en cumplir con tres características deseadas en un modelo de interpretabilidad.

Entender LIME es entender casi en su totalidad el funcionamiento de los shap values. Por lo tanto, se mostrará primero el desarrollo teórico de LIME, y por último, se explicará la aportación de los shap values.

Decimos que $g \in G$ es una explicación de un modelo, donde G es un conjunto de modelos potencialmente interpretables, como regresiones lineales. El dominio de g es $\{0,1\}^{d'}$. $\Omega(g)$ es una medida de complejidad de la explicación g. [6]

Sea $f : \mathbb{R}^d \to \mathbb{R}$ un modelo que se desea explicar. Sea $\pi_x(z)$ una medida de proximidad entre la instancia z y x, $L(f, g, \pi_x)$ una medida de cuan parecido es el modelo g a f en la vecindad de x definida por π_x. La explicación dada por LIME, se obtiene al minimizar:

$$L(f, g, \pi_x) + \Omega(x) \tag{3.25}$$

Dada una instancia x queremos explicar el modelo f bajo esta instancia. En LIME, existe el concepto de *Representación interpretable*, que es un vector $x' \in \{0,1\}^{d'}$ Es decir, dada una instancia, debemos definir una manera de convertirla a un vector binario. Esta transformación depende de la naturaleza de las instancias. Por ejemplo, en clasificación de texto, podemos pensar en la versión interpretable como un vector binario, donde los ceros indican la ausencia de una palabra, aunque el modelo original

puede estar usando una representación distinta como una matriz de TFIDF o algún embedding.

Una vez que se tienen las transformaciones $x = h(x')$, $x' = h^{-1}(x)$, se calcula la versión interpretable x', y se hace un muestreo aleatorio alrededor de x' de la siguiente manera. Se toma el vector binario x' y de manera aleatoria, se hacen 0 una cantidad también aleatoria de entradas que no eran 0. A las muestras obtenidas las denotaremos por z'_i. Una vez obtenidas estas muestras, se calcula su versión original, es decir $z_i = h(z'_i)$. Con las z_i se calcula $f(z_i)$ y entonces usando (3.18) se obtiene la explicación de LIME.

Un ejemplo de como puede lucir (3.18) es:

$$L(f, g, \pi_x) = \sum_{z,z' \in Z} \pi_x(z)(f(z) - g(z'))^2 \tag{3.26}$$

Donde g es un modelo de regresión lineal, $\pi_x(z) = e^{-\frac{D(x,z)}{2\sigma^2}}$ D alguna métrica de distancia como la distancia L2.

Los coeficientes de la regresión resultante, son la importancia de cada uno de las características interpretables.

Shap values hace uso de estos mismos principios, pero cambia el kernel por el Shapley Kernel[10].

$$\pi_x(z) = \frac{M - 1}{M|z'||z'|(M - |z'|)} \tag{3.27}$$

Donde M es la cantidad de características de las versiones interpretables, y $|z'|$ es la cantidad de entradas diferentes a cero en z'.

Este kernel se basa en los shapley values, que es una medida en teoría de juegos, que a grandes rasgos, mide en juegos colaborativos, que tanto aporta un individuo a una coalición.

Las tres propiedades que cumplen los shap values que ningún otro modelo de explicabilidad son las siguientes:

Precisión local

$$f(x) = g(x') = \phi_0 + \sum_{i=0}^{M} \phi_i x'_i \tag{3.28}$$

Es decir, el modelo original y su explicación son exactamente iguales en la instancia x y su versión interpretable x'

Missingness

$$x'_i = 0 \Longrightarrow \phi_i = 0 \tag{3.29}$$

Si una característica de la versión interpretable es cero, entonces su correspondiente valor en el coeficiente de la regresión es cero. Es decir, es una característica sin relevancia.

Consistencia

Sea $z'_i - i$ hacer $z_i = 0$. Si se tienen dos modelos f y:

$$f'(z) - f'(z - i) \geq f(z') - f(z - i), \forall z' \tag{3.30}$$

Entonces $\phi_i(f') \geq \phi_i(f)$. Es decir, si la iésima componente de la versión interpretable z' provoca mayores cambios en un modelo f' que en el modelo f para todas las z'. Entonces el coeficiente de la regresión ϕ_i es mas grande para f' que para f

En la siguiente imagen podemos ver un ejemplo de shap values para una instancia. Los valores en rojo muestran características que .empujan"la salida del modelo hacia valores mas negativos, mientras que valores en azul muestran las características que

llevan al modelo hacia valores mas grandes.

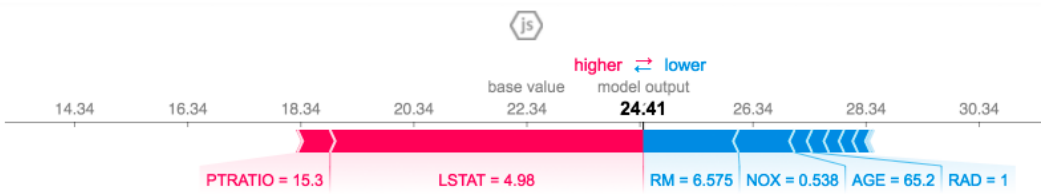

Figura 3.5: Ejemplo de shap values para una instancia

También puede visualizarse la importancia de las características, en la siguiente imagen, vemos de arriba hacia abajo el orden de la importancia de las características. En azul si los valores son bajos para la característica, y en rojo si son altos. De manera horizontal, podemos ver hacia que clase aportan las características. En el ejemplo, como característica más importante vemos LSTAT. Valores pequeños en LSTAT empujan la hacia la clase positiva, y valores grandes hacia la clase negativa.

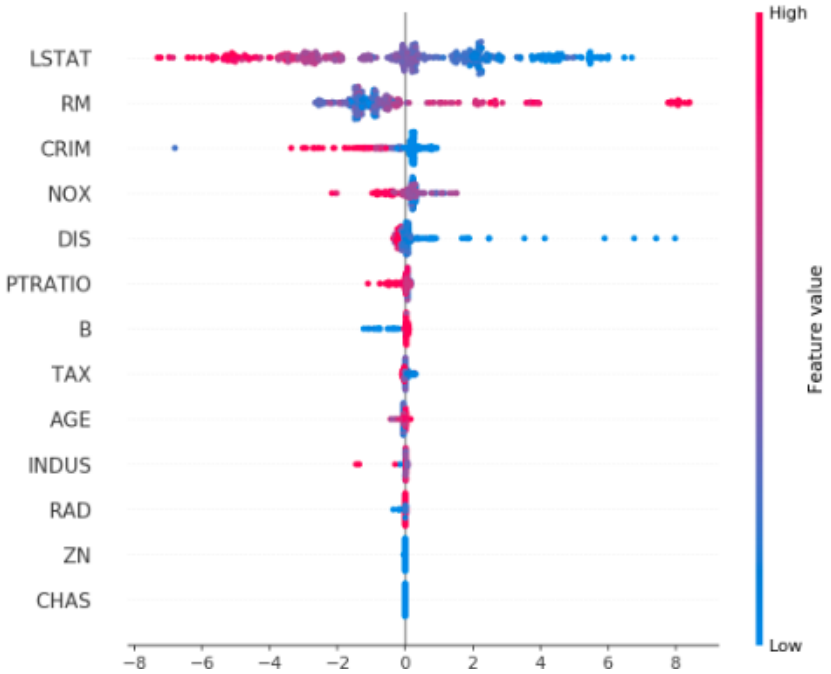

Figura 3.6: Importancia de características usando shap values

Calculando el valor medio de los valores absolutos para tener una idea mas general de la importancia de cada característica.

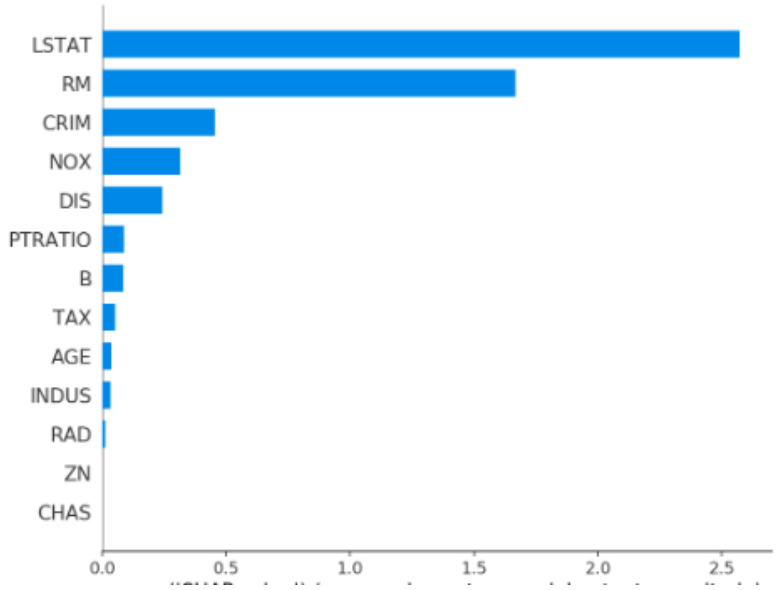

Figura 3.7: Media de los valores absolutos de los shap values

Capítulo 4

Segmentación y clasificación

En Kueski,al día, se realizan cientos de llamadas de cobranza, de confirmación de datos, de agradecimiento por pago, etc. Cada una de las llamadas es grabada y se almacena de manera que se puede identificar el día que se realizó, la persona de Kueski que realizó la llamada, el motivo de la llamada, a que cliente se llamó, y un identificador que relaciona con el préstamo.

Existe por tanto, una gran base de información que no es utilizada de ninguna manera, solo para mantener un historial de las llamadas. En este trabajo, se pretende explorar técnicas con las cuales sea posible extraer información útil, y automatizar este proceso de extracción lo más posible.

Como primer paso, se propuso dada una llamada, segmentarla en sus dos protagonistas. Partiendo del suspuesto que en una llamada, solo existen dos interlocutores:el cliente y el trabajador de Kueski. Una primera solución a este problema, fue entrenar un clasificador de género, de tal manera que en los casos en los que los interlocutores sean de diferente género, el clasificador puede separar las voces automáticamente.

4.1. Clasificación de género

Para lograr el clasificador de género, se utilizó la siguiente estrategia. Primero, se genero un dataset de audios de hombre y mujer. Para ello, se escucharon audios y aquellos en los que ambos hablantes fueran mujeres, se agregaron al conjunto de audios de mujer. Aquellos audios donde los dos interlocutores fueran hombres, se agregaron al conjunto de datos de hombres. Aquellos en los que hombre y mujeres hablaban en la llamada, fueron ignorados para esta etapa. Se consiguieron, 821 audios de mujeres y 661 de hombres.

Para la extracción de características en los audios, cada llamada fue preprocesada de tal manera que el primer paso fue eliminar los silencios, generando así audios mas cortos y continuos en sonido. Después se calcularon los bancos de filtros de mel escalados, es decir, los coeficientes de mel sin calcular la transfomada de coseno discreta. Se decidió utilizar estos descriptores, ya que dieron mejor resultado que los coeficientes de Mel. Se calculan 24 bancos de filtros, y una vez que se tiene la matriz de descriptores, se parten en 30 frames, es decir, se toman matrices de 24X30, donde cada matriz representa aproximadamente 0.6 segundos de audio.

En análisis de voz, se recomienda utilizar duraciones de audio de aproximadamente 0.6 segundos, no mucho mas corto, porque no alcanzaría a describirse los fonemas de la voz humana, y no mucho mas largo porque es mas probable que es una muestra entre dos voces diferentes[3].

Así pues, cada 0.6 segundos aproximadamente quedan representados por matrices de 24 X 30, es decir, 720 entradas. En total, se generaron 228413 matries del conjunto de audios de género, 119468 etiquetados como mujeres y 108945 como hombres.

4.1.1. Clasificación de género basada en redes neuronales convolucionales

En [3] se propone un clasificador de género basado en redes neuronales convolucionales, en este trabajo se toma una arquitectura similar pero mas simple con el fin de reducir el costo computacional, y el conjunto de entrenamiento es propio(las llamadas de Kueski).

Se utilizó una red neuronal con la siguiente arquitectura:

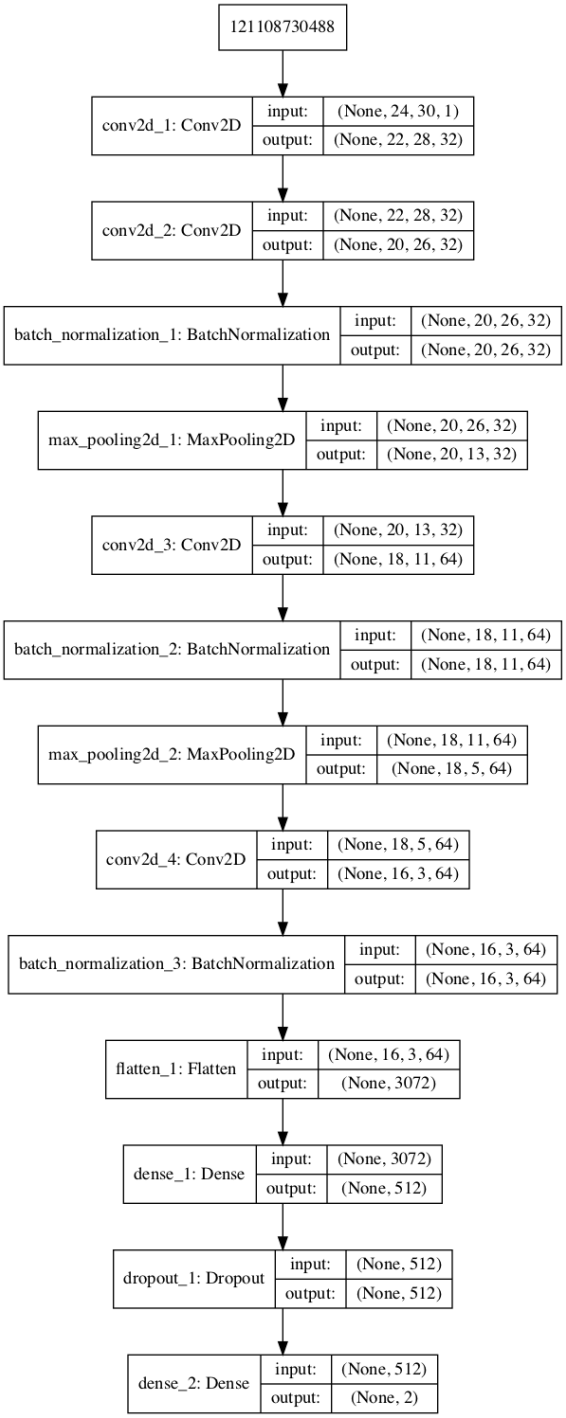

Figura 4.1: Arquitectura Red Neuronal

- Capa de convolución con 32 kernels de 3x3 activación RELU
- Capa de convolución con 32 kernels de 3x3 activación RELU

- Capa de normalización de lote(batch normalization)
- Max pooling 1x2
- Capa de convolución con 64 kernels de 3x3 activación RELU
- Capa de normalización de lote(batch normalization)
- Max pooling 1x2
- Capa de convolución con 64 kernels de 3x3 activación RELU
- Capa de normalización de lote(batch normalization)
- Capa de aplanamiento
- Capa densa de dos neuronas, activación softmax, dropout 0.5, regularización L2 0.01

En total 71458 parámetros entrenables.

Como algoritmo de entrenamiento se utilizó ADAM, cómo función de perdida *categorical cross entropy*, cómo medida de desempeño la exactitud y tamaño de lote de 1000.

Para la optimización sobre los hiperparámetros, tales como valor de regularización L2, dropout y batch size, se utilizó Talos, que hace una grid search sobre el espacio de los hiperparámetros.

4.1.2. Experimentos y resultados

Para evaluar la calidad del modelo de género, se tomo el conjunto de datos de audios C̈ommon Voiced̈e Kaggle. Este conjuto cuenta con miles de audios donde cada audio consta de una sola voz. El conjunto de datos cuenta con etiquetas de género. Todos los audios son en inglés. Se utilizaron 18249 audios de hombres y 18249 de mujeres para la evaluación.

Al momento de extraer las características de estos audios para el modelo de género, se obtuvieron 33965 instancias etiquetadas como voz de mujer, y 30368 como voz de hombre.

Las métricas fueron las siguientes:

- **Exactitud** : 0.89196
- **F1 Score**: 0.89165
- **Proporción de hombres bien clasificados** : 0.94168
- **Proporción de mujeres bien clasificados** : 0.84752

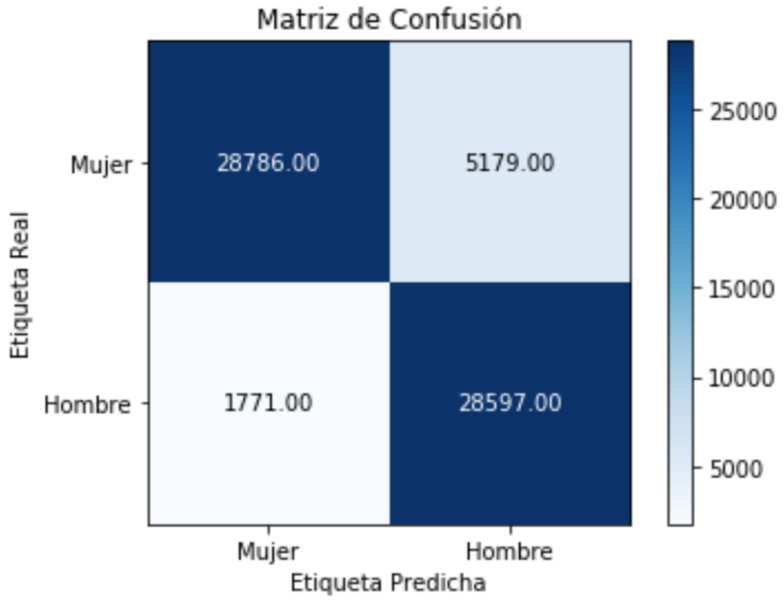

Figura 4.2: Matriz de confusión modelo de género

Estos resultados son sobre audios provenientes de fuentes diferentes a los audios con los que fue entrenado el modelo y con idioma diferente. Al momento de usar el modelo sobre audios de Kueski, la calidad de la predicción se eleva dado que, la fuente de los audios es la misma que los audios de entrenamiento, y además, sucede que muchas veces un audio contendrá la voz de un cobrador que fue utilizada para entrenar el modelo. Para fines de este trabajo, esto último no representa un problema,

al contrario, ayuda a mejorar la calidad de la segmentación de los nuevos audios que se van produciendo.

4.2. Segmentación basada en clustering

La segmentación anterior, solo funciona en el caso de que los interlocutores sean de géneros opuesto, si el clasificador detecta que ambos son hombres o ambos mujeres, entonces se procede a realizar la segmentación con una técnica diferente.

Se calculan los coeficientes de Mel usando 30 filtros y 30 frames. Además, se calculan los deltas de orden 1 y 2, quedando entonces tensores de 30X3X3. Cada uno de estos tensores, representa aproximadamente 0.6 segundos de audio.

Sin los deltas, solo se tendrían características estáticas de los fragmentos de los audios. Con los deltas en cambio, podemos obtener la variación de los mismos através del tiempo, obteniendo características dinámicas de los mismos.

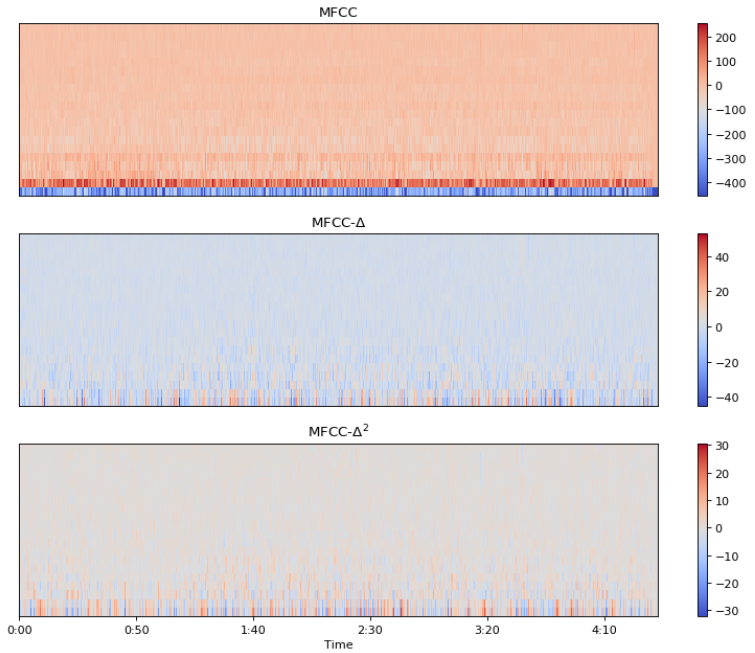

Figura 4.3: Características para segmentación basada en clustering

Una vez se tienen los tensores que caracterizan a los diferentes fragmentos del audio, se utiliza el algoritmo de Birch para hacer clustering utilizando dos clases. La idea es que los fragmentos de las voces de las personas queden en un cluster diferente. Se experimentó con diferentes técnicas de agrupación, y el que mostró mejores resultados fue Birch. Es difícil dar métricas para la calidad de la segmentación, ya que implicaría segmentar los audios de manera manual para comparar con el resultado del algoritmo, y la segmentación manual de los audios es demasiado tediosa, por lo que la calidad de la segmentación se basó en escuchar el resultado de audios procesados por las diferentes técnicas de agrupamiento y quedarse con el que a .ºido"fue el que obtuvo mejores resultados.

Para el algoritmo de Birch se utilizaron los siguientes hiperparémtros:

- Threshold = 0.5
- Branching Factor = 50

4.2.1. Experimentos y resultados

Es difícil dar una métrica de la calidad de la segmentación basada en clustering. Se eligió el método de agrupamiento que mejores resultados dió escuchando los resultados.

Se notó que entre más largo es un audio, mejor es la segmentación. Posiblemente se debe a que en audios de mayor duración, se tienen más muestras de las voces y las características que definen una voz de otra son más fáciles de separar. Por ejemplo, en un audio pequeño, al separar en dos clases, puede que al no tener muchas muestras que represtan a las voces, otras características sobresalgan y se separen con base en estas. Por ejemplo, en un audio donde en un comienzo hablan ambos locutores de forma calmada, y después elevan su tono de voz, la segmentación podría separar en una clase la parte tranquila de la conversación, y en la otra clase la parte agresiva.

Afortunadamente, hasta el momento se observaron llamadas que transcurren de forma tranquila y de suficiente duración para poder separar de manera correcta las voces.

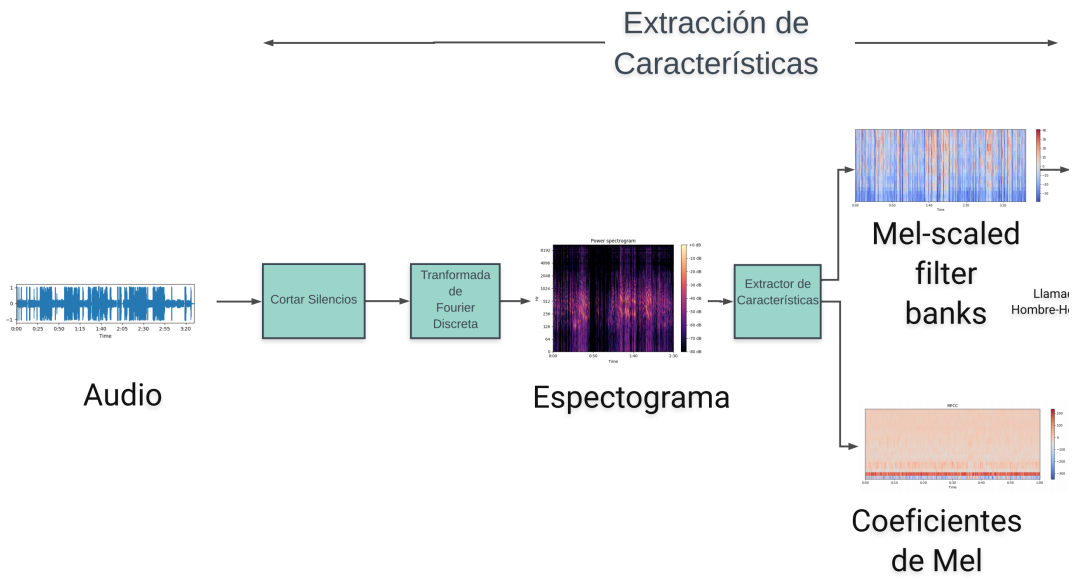

Figura 4.4: Proceso para extraer características de los audios

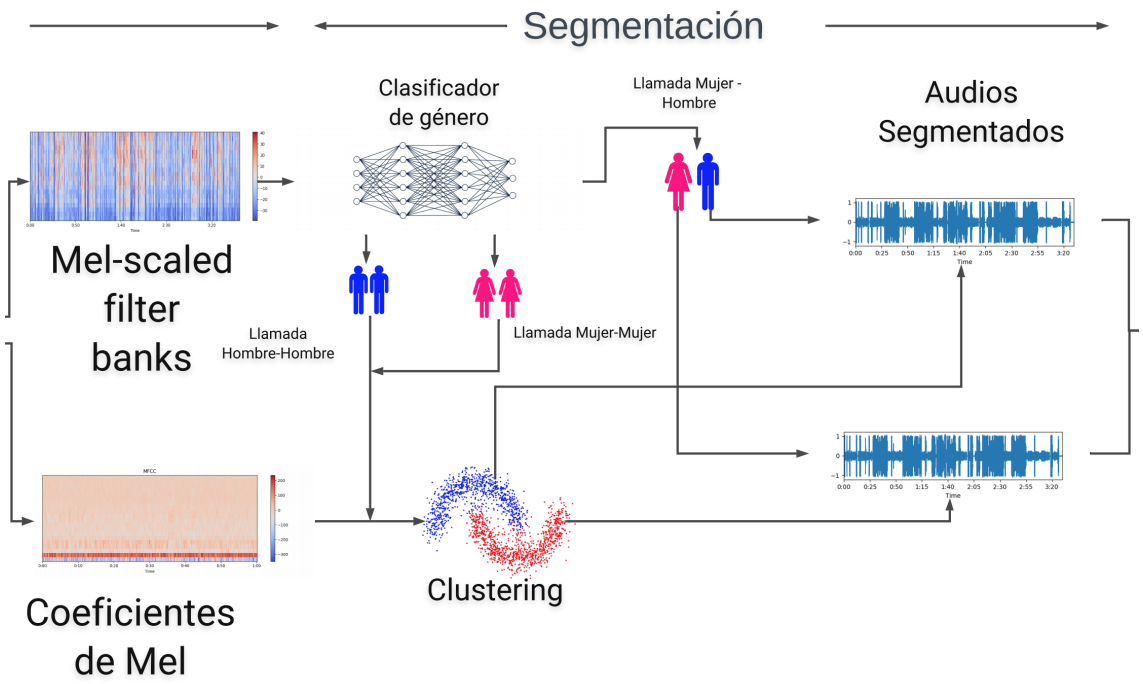

Figura 4.5: Proceso para segmentar las voces

4.3. Audio a texto

Una vez se tienen las voces segmentadas, la siguiente etapa es convertir estos audios a texto. Para ellos, se utilizó al API de Google *Speech to Text Google Cloud*. Esta API puede calibrarse para mejorar la calidad de la traducción a texto.

4.3.1. Experimentos y resultados

Para calibrar la API, se utilizó una lista de palabras y frases que de antemano se notó que utilizaban mucho los cobradores. Algunas de estas frases y palabras son:

- Kueski
- Círculo de crédito
- Buró de crédito
- La página de préstamos
- Intereses

Y algunas otras más. Con esto, cuando aparece una de estas palabras o frases, la API de Google aumenta en su certeza sobre esas frases, y mejora la traducción al no equivocarse en estas y dar una traducción diferente. También se proporcionó a la API la velocidad de muestreo de los audios y el tipo de iteracción como llamada telefónica e idioma español.

Pero antes de usar la API, se necesitó cambiar el códec de los audios, ya que la Google Cloud no soporta el códec IEEE. Por lo tanto, como primer paso, se utilizó la librería de FFMPEG para cambiar los códec de los audios de IEEE a PCM-16.

Para poder usar la API de Google, los audios deben ser menores a un minuto, por lo tanto, se cortaron los audios en bloques de 59 segundos a lo máximo, y se hicieron las llamadas a la API con estos bloques, una vez que se regresan los bloques de texto,

estos se unen en un solo archivo.

A continuación algunos ejemplos del resultado de traducir el audio a texto de un cliente y un cobrador.

```
Hola si Bien gracias ayer mandé un correo porque vi que la página decía que si no podía
hacer el pago de haber que mandar el correo y que ellos me iban a regresar una respuesta
pero como no me regresan respuesta de té apenas iba yo a marcar el número que aparece ahí
Mira la hora yo no soy tan me quedé sin dinero porque me enfermé no cuento con toda la
cantidad entonces no sé si me podrían dar a poder pagar si yo creo que si no yo tendría
```

Figura 4.6: Ejemplo de texto de cliente

```
Hola buenos días tengo el gusto con ******* hola *****  te llama ****** de kueski.com hola
no te quito mucho tiempo mi llamada es de servicio solamente para recordarte que el día de
hoy vence el plazo de tu préstamo Kueski que tienes con nosotros por 1214 Con 12 centavos
Me gustaría saber si podemos contar con su pago el día de hoy signo de Cómo realizar el
pago o algo referente a tu cuenta en lo que pueda apoyarte muy Así es igual yo te estoy
mandando un mensaje de texto con la información que necesitas para realizar el pago así Si
tengas alguna duda sobre tu cuenta o algo más en lo que pueda apoyarte te confirma
entonces a 1414 Con 12 centavos te agradezco alguna duda queja o sugerencia nos puedes
escribir a soporte@kueski.com que tengas un excelente día gracias
```

Figura 4.7: Ejemplo de texto de cobrador

4.4. Clasificación con base en texto

Con los textos, se entrenó un modelo para clasificar si el origen entre cliente o cobrador. El modelado del texto se describe a continuación.

4.4.1. Term frequency – Inverse document frequency

Una vez obtenidos los textos, se experimentó con la creación de un clasificador, donde la variable de respuesta sea el rol de la persona, es decir, cliente o cobrador. Se creó una bolsa de palabras, y se intentó con diferentes tamaños de enegramas. Los mejores resultados fueron al usar 3-gramas y usando un tamaño de bolsa de 100.

Luego se calculó el TFIDF de esta bolsa de palabras. TFIDF es una técnica para extraer la relevancia de una palabra o n-gram en una bolsa de palabras. TFIDF es el producto de dos medidas:

$$TF(t,d) = \frac{f(t,d)}{max\{f(t,d)\}\ t \in d} \tag{4.1}$$

TF mide la frecuencia de cada término en su documento y luego lo normaliza dividiendo entre la máxima frecuencia que aparece en el documento

$$IDF(t,D) = log\frac{|D|}{|\{d \in D : t \in D\}|} \tag{4.2}$$

Es el número de documentos en total, entre el número de documentos donde el término aparece. A esta fracción se calcula el logaritmo.

TFIDF es el producto de ambas medidas. De esta manera, se castiga a términos que son demasiado comunes como árticulos y conectores, ya que este tipo de términos tienen poco poder predictivo, ya que aparecen en ambas clases al ser comunes

Con la matriz obtenida de TFIDF, se separaron los datos en un conjunto de entranmiento y pruebas, y se entrenó un modelo XGBOOST, utilizando 80 textos de cobradores y 80 de clientes, y se evaluó en un conjunto de 20 textos de clientes y 20 de cobradores.

4.4.2. Experimentos y resultados

El resultado fue una precisión de 1. Entonces se aplicó al modelo la técnica de interpretabilidad de shap values. Se descubrió que todas las instancias clasificadas como cobradores, las características que en este caso son 3-grams, empujaban hacia la clase cobrador. Mientras que para las instancias cliente, ninguna característica tenía influencia en la clasificación.

Posiblemente 3-gramas funciona mejor que palabras individuales, ya que por ejemplo, la palabra Kueski puede ser mencionada tanto por cliente o cobrador, pero frases estilo: "llamo de Kueski", son solo dichas por cobradores.

Figura 4.8: Shap Values para un cobrador

Algunos 3-grams en el diccionario fueron:

```
'el motivo de': 32,
'motivo de mi': 58,
'de mi llamada': 17,
'mi llamada es': 56,
'es de servicio': 41,
'recordarte que hoy': 86,
'que hoy vence': 77,
'del día de': 23,
'si tienes alguna': 89,
'tienes alguna duda': 95,
'de pago algo': 18,
'pago algo más': 65,
'vence el plazo': 99,
'en verse reflejado': 39,
'círculo de crédito': 12,
'kueski com que': 48,
'com que tengas': 9,
'de servicio para': 21,
'servicio para recordarte': 87,
```

Figura 4.9: Diccionario de 3-grams

Por lo que el modelo aprendió que en ausencia de frases comunes se trata de un cliete, y entre más frases comunes, más probable es que se trate de un cobrador. Esto debido a que todos los cobradores tienen un guión prestablecido al momento de hacer las llamadas de cobranza.

Con estos pasos, se logra clasificar si la llamada segmentada proviene de un cliente o de un cobrador.

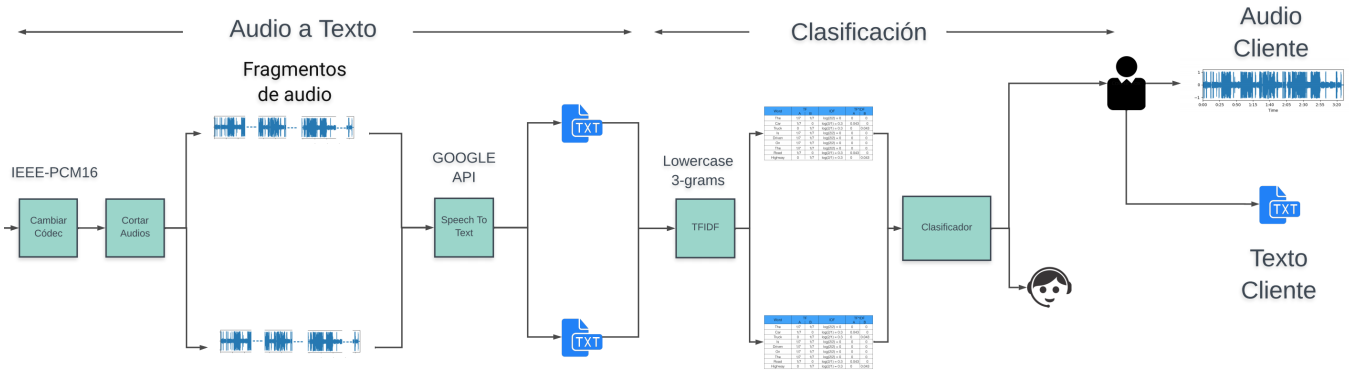

Figura 4.10: Proceso para clasificación de cliente y cobrador

Capítulo 5

Modelo de predicción de pago

En este capítulo, se explora la construcción del modelo de predicción del comportamiento de pago dado un conjunto de audios de cobranza.

Se tomaron los audios de todas las llamadas del mes de enero del 2019, que corresponden a 9952 llamadas en total. Para cada llamada se tiene un conjunto de información, como el número del teléfono a que se llamó, comentarios del cobrador, nombre del cobrador, fecha de la llamada, un identificador que relaciona la llamada con el préstamo en cuestión llamado loanid, etc.

Primero se filtró toda esta información y se generó una tabla con el nombre del archivo del audio, la fecha que se realizó la llamada, y el loanid.

	File Name	Date	Loan Id
0	MARCADOR035/2019_1/2/9/_ListaFP1_[redacted]..	02/01/19 09:05 AM	5c2c3c2dd80ce890ce3e041b
1	MARCADOR035/2019_1/2/9/_ListaFP1_[redacted]..	02/01/19 09:07 AM	5c2c2cb72fe742e356549cf7
2	MARCADOR035/2019_1/2/9/_ListaFP1_[redacted]..	02/01/19 09:08 AM	5c2c0aeb79e07795b97135bc
3	MARCADOR035/2019_1/2/9/_ListaFP1_[redacted]..	02/01/19 09:08 AM	5c2bfc42e3d70de5be6f2af3
4	MARCADOR035/2019_1/2/9/_ListaFP1_[redacted]...	02/01/19 09:08 AM	5c2be3d4b5b2377f14ce66e0

Figura 5.1: Información filtrada de cada audio

El siguiente paso es filtrar todas las llamadas con duración menor a un minuto.

El objetivo de esto es eliminar las llamadas no contestadas o a números equivocados. Se ha visto por experiencia que llamadas de este tipo duran menos de un minuto.

El objetivo siguiente fue determinar cuales llamadas eran de cobranza y filtrarlas. Esto se logró de la siguiente manera. Primero, se buscó en la base de datos de Kueski, un campo en una tabla llamado $is_disbursed$

	loan_id	is_disbursed
0	5be7077bcf6daf3128484dce	1
1	5be9c34ca13f33963890892d	1
2	5be9e1bc215ad217575e735b	1
3	5beb5169bd6a32e3d48939b0	1
4	5bec5b991737f8a74a6d5b74	1

Figura 5.2: Campo is_disbursed

Este campo tiene cero si Kueski no depositó el préstamo, y un uno si lo hizo.

Se eliminaron todas las llamadas donde Kueski no realizó el depósito. Existen diferentes razones por las que Kueski no realiza el depósito, puede que el cliente decida no continuar con el trámite, puede faltarle documentación, o también podría sospecharse de fraude y no realizarse el préstamo. Existen llamadas de confirmación de datos, que en su mayoría pertenecen a los audios con un cero en $is_disbursed$.

Una vez que se tienen llamadas donde Kueski sí hizo el préstamo, se buscó el campo due_date. Este campo contiene la fecha de vencimiento del préstamo.

	loan_id	due_date
0	5be7077bcf6daf3128484dce	2018-12-14
1	5be9c34ca13f33963890892d	2018-12-12
2	5be9e1bc215ad217575e735b	2018-12-12
3	5beb5169bd6a32e3d48939b0	2018-12-13
4	5bec5b991737f8a74a6d5b74	2018-12-14

Figura 5.3: Campo due_date

Se comparó la fecha de la llamada con la fecha de vencimiento, si la llamada fue realizada con anterioridad a la fecha de vencimiento, se elimina. Con esto, se quitan las llamadas de confirmación de datos a usuarios que sí se les prestó.

El siguiente paso fue buscar el campo *paid_date*. Este campo contiene la fecha en que se pagó, o NaN en caso de no haberse realizado el pago hasta este momento.

	loan_id	paid_date
0	5be7077bcf6daf3128484dce	None
1	5be9c34ca13f33963890892d	None
2	5be9e1bc215ad217575e735b	2019-01-07
3	5beb5169bd6a32e3d48939b0	None
4	5bec5b991737f8a74a6d5b74	2019-03-14

Figura 5.4: Campo paid_date

Si la llamada se realizó después de que el cliente pagó, se elimina. Existen casos donde los pagos de los clientes se reflejan hasta 48 horas después, por lo que se realizan llamadas de cobranza sobre pagos que ya se realizaron. También existen llamadas de agradecimiento por el pago.

Las llamadas restantes se consideran llamadas de cobranza. A continuación se

buscó el campo *paid_delayed*, este campo contiene NaN en caso de que no haya sido pagado el préstamo hasta este momento. Un número entero que significa el número de días que se pagó con retraso. Un cero si se pagó el día de vencimiento, o un número negativo si se pagó antes de la fecha de vencimiento.

	loan_id	paid_delayed_time
0	5be7077bcf6daf3128484dce	NaN
1	5be9c34ca13f33963890892d	NaN
2	5be9e1bc215ad217575e735b	26.0
3	5beb5169bd6a32e3d48939b0	NaN
4	5bec5b991737f8a74a6d5b74	90.0

Figura 5.5: Campo paid_delayed_time

La variable de respuesta se calculó con base en este campo de la siguiente manera:

$$y = \begin{cases} 0, & x = NaN \\ 0, & x \geq 30 \\ 1, & x \leq 30 \end{cases} \tag{5.1}$$

Es decir, cero si el cliente no ha pagado o pagó después de 30 días. 1 si pagó antes de los 30 días. Después de 30 días, las llamadas de cobranza son pasadas a una agencia externa.

Además se agrega una columna con el número de días de retraso que lleva el pago el día que se hizo la llamada (days late) y el número de veces que ya se había llamado a la persona por ese préstamo antes de esa llamada (num calls).

	File Name	Date	Loan_Id	paid_delayed_time	target	days late	num calls
143	MARCADOR035/2019_1/4/10/_[redacted]..	04/01/19 10:05 AM	5c095043b31eff8bf61fa4de	NaN	0	0.0	0
448	MARCADOR035/2019_1/8/11/_[redacted]_...	08/01/19 11:59 AM	5c095043b31eff8bf61fa4de	NaN	0	4.0	1

Figura 5.6: Múltiples llamadas debidas al mismo préstamo

Con la variable de respuesta calculada, se separan los audios en las dos categorías. La tabla final de los audios filtrados queda de la siguiente manera:

	File Name	Date	Loan_Id	paid_delayed_time	target	days late	num calls
430	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:39 AM	5c1a5cedd074c67804cd1df1	0.0	1	0.0	0
431	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:39 AM	5c0d440effbe344a939852e4	2.0	1	0.0	0
432	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:39 AM	5c1fd5186bd4a396f682ffeb	0.0	1	0.0	0
433	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:41 AM	5c1a6573acef646d52921f0c	NaN	0	0.0	0
434	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:41 AM	5c1811efea38933d3b357a18	1.0	1	1.0	1
435	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:42 AM	5c1a651660bdba3075f148e7	0.0	1	0.0	0
436	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:44 AM	5c18544b21e5ccc84dbf65f1	2.0	1	0.0	0
437	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:45 AM	5c1419f84be9ef95a2b2fb70	NaN	0	0.0	0
438	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:46 AM	5c12b0f444625238bd48ae46	NaN	0	1.0	0
439	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:46 AM	5c1051f26ce8b0241bba3393	0.0	1	0.0	0
440	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:47 AM	5c084e0334f47a13c000c036	NaN	0	4.0	0
441	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:47 AM	5c1407bac75885a8a905cf2e	NaN	0	5.0	0
442	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:47 AM	5c27a1491d8fd8ce788118bf	0.0	1	0.0	0
443	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:49 AM	5c198bd0ed4de4d7f19a8ab9	NaN	0	0.0	0
444	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:50 AM	5c1aceebc19471b17cd0a76f	13.0	1	0.0	0
445	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:52 AM	5c186d17dabce4539bcb48f7	1.0	1	1.0	0
446	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:52 AM	5c13c4066d52b804452ba58b	NaN	0	5.0	0
447	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:55 AM	5c0695f1842755290cf41cd5	7.0	1	5.0	0
448	MARCADOR035/2019_1/8/11/[redacted]	08/01/19 11:59 AM	5c095043b31eff8bf61fa4de	NaN	0	4.0	1

Figura 5.7: Datos de las llamadas filtradas

5.1. Deep Feature Synthesis

Una vez que se obtuvieron solo llamadas de cobranza, y se extrajo el audio de los clientes, se extrajeron características de los audios utilizando el algoritmo deep feature synthesis de la siguiente manera.

De cada audio se extrajo su serie de tiempo a su velocidad de muestreo original. Se creó una tabla nombrada *time_series* donde cada fila representa una muestra del audio. La columna ts es el valor del audio en la muestra, además existen las columnas *loan_id*, el nombre del archivo y la fecha.

	index	ts	loan_id	file_name	date
2000	2001	0.001	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2001	2002	0.000	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2002	2003	-0.000	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2003	2004	-0.000	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2004	2005	-0.001	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2005	2006	-0.003	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2006	2007	-0.004	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2007	2008	-0.004	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2008	2009	-0.003	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2009	2010	-0.000	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2010	2011	0.003	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2011	2012	0.005	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2012	2013	0.006	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2013	2014	0.005	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2014	2015	0.003	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2015	2016	0.001	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2016	2017	-0.001	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2017	2018	-0.003	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2018	2019	-0.004	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16
2019	2020	-0.005	5c180937c8f064c5a047b964	_[redacted]_20190116_093806.WMA	2019-01-16

Figura 5.8: Series de tiempo de los audios

Después se crean dos tablas nuevas, una donde cada fila es el nombre de cada audio, y una columna con su fecha. Esta tabla se nombra *files*

	file_name	first_time_series_time
_[redacted]_093806.WMA	_[redacted]_093806.WMA	2019-01-16
_[redacted]_101939.WMA	_[redacted]_101939.WMA	2019-01-31
_[redacted]_112059.WMA	_[redacted]_20190118_112059.WMA	2019-01-18
_[redacted]_105251.WMA	_[redacted]_20190121_105251.WMA	2019-01-21
_[redacted]_101321.WMA	_[redacted]_101321.WMA	2019-01-25
_[redacted]_090215.WMA	_[redacted]_090215.WMA	2019-01-28
_[redacted]_154937.WMA	_[redacted]_154937.WMA	2019-01-10
_[redacted]_111008.WMA	_[redacted]_111008.WMA	2019-01-29
_[redacted]_105225.WMA	_[redacted]_105225.WMA	2019-01-23
_[redacted]_110319.WMA	_[redacted]_20190114_110319.WMA	2019-01-14

Figura 5.9: Tabla con nombres de archivos

La otra tabla creada tiene como filas los diferentes *loan_id* y la fecha de la primera llamada con ese *loan_id* relacionado. A esta tabla se llama *loan_id*

	loan_id	first_time_series_time
5c180937c8f064c5a047b964	5c180937c8f064c5a047b964	2019-01-16
5c2e65e0eac6eb2fa2691c29	5c2e65e0eac6eb2fa2691c29	2019-01-31
5c1aa68d829ad4370543819c	5c1aa68d829ad4370543819c	2019-01-18
5c1e995bb669afc86a165d5b	5c1e995bb669afc86a165d5b	2019-01-21
5c2518e7f7ea7edf54ae38ba	5c2518e7f7ea7edf54ae38ba	2019-01-25
5c338f6c4ab3e0eb865b9443	5c338f6c4ab3e0eb865b9443	2019-01-28
5c07d7ef9b634b2cd38d194a	5c07d7ef9b634b2cd38d194a	2019-01-10
5c29165b5da1e0d6a2c9e0d4	5c29165b5da1e0d6a2c9e0d4	2019-01-29
5c20fe4d895da81f724827d9	5c20fe4d895da81f724827d9	2019-01-23
5c1518b26f5a560504145dd6	5c1518b26f5a560504145dd6	2019-01-14

Figura 5.10: Tabla con loan id

El objetivo de estas tablas es ejecutar el algoritmo de deep feature synthesis, utilizando los valores de la serie de tiempo, pero creando características agrupando estos valores por nombre de archivo y por *loan_id*.

Es necesario establecer las relaciones entre las tres tablas, así de la tabla *time_series* el campo *file_name* se relaciona con el campo del mismo nombre de la tabla *files*. El campo *loan_id* de la tabla *time_series* se relaciona con el campo homónimo de la tabla *loan_id*

Una vez teniendo las 3 tablas (entidades) y sus relaciones, se establece la tabla *files* como la entidad objetivo.

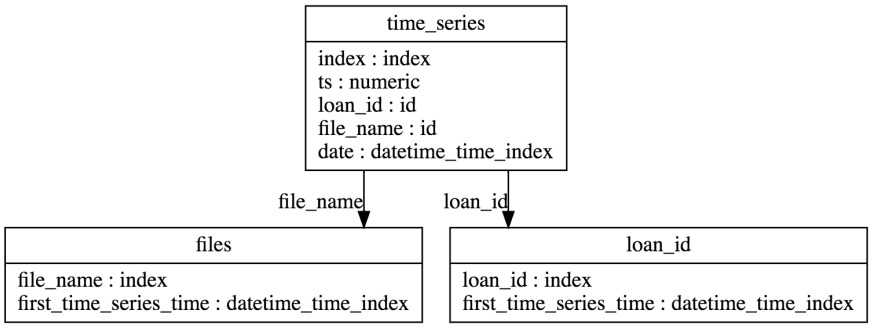

Figura 5.11: Relaciones entre tablas

5.2. Extracción de características de series de tiempo

Una vez construido el conjunto de entidades, sus relaciones y la entidad objetivo, se necesitan establecer el tipo de características que se calcularan utilizando deep feature synthesis. Por lo general, deep feature synthesis se utiliza con características de transformación y agregación. En este trabajo solo se utilizan de agregación. Características comunes de este tipo son: mínimo, máximo, promedio, cuantil, desviación estándar,etc. En este trabajo, se agregaron además otro tipo de características para serie de tiempo.

Energía absoluta

$$\sum_{i=1,..n} x_i^2 \tag{5.2}$$

Suma absoluta de cambios

$$\sum_{i=1,..n-1} |x_{i+1} - x_i| \tag{5.3}$$

Complejidad

$$\sqrt{\sum_{i=1,..n-1} (x_i - x_{i+1})^2} \tag{5.4}$$

Cambio promedio

$$\sum_{i=1,..n-1} x_{i+1} - x_i \tag{5.5}$$

Promedio de la segunda derivada centrada

$$\sum_{i=1,..n-1} (x_{i+2} - 2x_{i+1} + x_i) \tag{5.6}$$

NUM_UNIQUE(time_series.loan_id)	COUNT(time_series)	COMPLEXITY(time_series.ts)	ABS_ENERGY(time_series.ts)	ABS_SUM_CHANGES(time_series.ts)
1	1131520	154.770	36862.617	58333.188
1	3494400	99.392	52266.016	87460.281
1	1936896	96.493	47862.820	66986.797
1	411648	41.374	4850.860	12914.618
1	600064	80.782	21730.555	32066.398
1	3431936	103.522	59126.523	87866.211
1	1018368	109.406	39428.438	47182.133
1	1454080	119.956	70947.594	77462.195
1	1300992	89.292	44947.117	49176.172
1	600064	56.041	15873.293	19765.492

Figura 5.12: Características de deep feature synthesis

En total se crean 90 características. El algoritmo deep feature synthesis utiliza las relaciones en las entidades para crear características que solo dependen del audio en cuestión, y algunas otras son creadas utilizando todos los audios relacionados con el mismo *loan_id*

5.3. Características espectrales

Para cada audio se calcularon 20 coeficientes de MEL. Para cada coeficiente se calcularon las siguientes características: Media, varianza, energía absoluta(5.2), suma absoluta de cambios(5.3), complejidad(5.4), cambio promedio(5.5) y promedio de la segunda derivada centrada(5.6). Para cada audio, se tienen entonces 140 nuevas características.

MEL_MEAN_2	MEL_STD_2	MEL_ABS_ENERGY_2	...	MEL_ABSOLUTE_SUM_CHANGES_19	MEL_MEAN_ABS_CHANGE_19
129.568	64.437	246253892.795	...	35005.532	2.977
105.037	75.201	75663006.382	...	11381.557	2.511
122.047	73.766	87793872.663	...	12763.531	2.957
104.184	61.280	372194214.373	...	70807.672	2.779
148.488	63.024	222814001.912	...	27066.765	3.161

Figura 5.13: Características espectrales

5.4. Selección de características

Una vez que se tiene la matriz de características, se clasifican estas en constantes, reales y binarias. Constantes son aquellas características que son iguales para todas las instancias, binarias aquellas que solo contienen dos valores diferentes y reales aquellas que toman más de dos valores. Las características constantes son eliminadas. No se encontraron características binarias.

Para las características reales, se calcula un test de significancia para generar p-values para cada característica. Esto se logra de la siguiente manera. Se toma cada característica como un vector que tendrá tantas entradas como instancias en el conjunto de datos. Es decir, para la i-ésima característica tendremos el vector x_i Dividimos este vector en dos, donde un primer vector tendra las características donde la variable de respuesta es cero, y en el otro donde la variable de respuesta es uno, generando los vectores x_{i0} y x_{i1} respectivamente. Sea n_0 la dimensión de x_{i0} y n_1 de x_{i1} se calcula:

$$U_0 = n_0 n_1 + \frac{n_0(n_0+1)}{2} - R_0 \tag{5.7}$$

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1 \tag{5.8}$$

Donde R_0 es la suma de los rangos de las entradas del vector x_{i0} y R_1 del vector x_{i1}

El estadístico U de la prueba se toma como

$$U = max(U_0, U_1) \tag{5.9}$$

Después se calcula la desviación estándar y la media del estadístico:

$$\sigma = \sqrt{\frac{n_0 n_1 (n_0 + n_1 + 1)}{12}} \tag{5.10}$$

$$m = \frac{n_0 n_1}{2} \tag{5.11}$$

La aproximación a una normal es:

$$z = \frac{U - m}{\sigma} \tag{5.12}$$

$$p = 2(1 - F_z(|z|)) \tag{5.13}$$

Donde F_z es la función de distribución (acumulada) de la normal estándar.

Una vez que se tienen los p-values, se seleccionan solo las características relevantes usando el método de Benjamini Hochberg. Esto es, primero se ordenan los p-values de menor a mayor. Para cada p-value se calcula

$$\frac{\alpha r}{m} \tag{5.14}$$

Donde α es un nivel de significanca, r es el rango que le toca al p-value, y m la cantidad de p-values.

Solo se toman las características cuya cantidad en (5.14) sea menor que su p-value.

5.5. Modelo

Se tomaron las 90 características generadas con deep feature synthesis utilizando los audios como series de tiempo, y 140 características generadas por los coeficientes de MEL. Después de hacer selección de características con Mann Whitney, son seleccionadas 67 características.

File Name	days late	MEL_STD_11	MEL_STD_7	MEL_ABS_ENERGY_1	MEL_STD_14	MEL_MEAN_ABS_CHANGE_7	MEL_MEAN_ABS_CHANGE_11	MEL_ABS_ENERGY
23026.WMA	0.000	8.877	15.612	616457471.889	7.149	6.251	4.172	5409346.5
05251.WMA	0.000	9.113	18.284	328053340.583	6.789	6.773	4.100	3029310.3
21107.WMA	0.000	9.302	17.306	471706967.768	7.037	6.014	3.738	3329781.8
01421.WMA	0.000	7.150	14.491	437102862.251	6.262	4.728	3.038	4903407.0
20146.WMA	0.000	9.307	17.903	1100859884.622	7.335	6.455	4.324	6357158.3
02639.WMA	0.000	8.914	15.781	1109785895.915	6.688	6.006	4.085	9872306.5
14013.WMA	0.000	9.297	17.796	247035221.554	6.781	6.724	4.164	3232557.2
94430.WMA	0.000	9.819	17.488	361871838.065	7.282	6.251	4.068	5185236.2
01755.WMA	0.000	8.837	18.772	307533322.354	6.481	6.524	4.192	6183197.0
15414.WMA	0.000	10.972	19.010	670670597.656	7.849	5.704	3.813	5171082.6
15140.WMA	0.000	9.673	15.699	776450764.046	7.331	6.173	4.288	20414964.4
11437.WMA	0.000	10.845	18.068	346481184.952	7.553	6.555	4.269	3020582.3
24105.WMA	0.000	9.273	15.686	333930725.293	7.015	5.876	4.031	3547807.7
23720.WMA	0.000	8.063	14.606	535292881.090	6.404	5.417	3.610	3873863.2
94317.WMA	0.000	9.090	17.228	562199042.624	7.186	5.599	3.895	10301751.4
21609.WMA	0.000	8.839	17.647	454408240.086	6.602	6.579	4.092	2600240.9
85902.WMA	0.000	6.700	12.876	686020420.192	5.605	4.027	2.700	7413161.9
22601.WMA	0.000	9.432	17.217	269847186.157	6.882	6.314	4.084	4033674.6

Figura 5.14: Algunas características seleccionadas

Con las características anteriores, y variable de respuesta (5.1), se divide el conjunto de datos en entrenamiento y validación, tomando el 20 por ciento de los datos para validación.

Se entrenaron modelos XGBoost, definiendo el espacio de hiperparámetros como el siguiente:

learning_rate	[0.1,0.6]
n_estimators	[100,3000]
max_depth	[1,5]
subsample	[0.3,1]
colsample_bytree	[0.2,1]
gamma	[0,5]
scale_pos_weight	[0.3,0.9]

Los hiperparámetros fueron optimizados utilizando bayesian tunning and bandits. Se hicieron 1000 iteraciones en la búsqueda de estos hiperparámetros.

Dado que las clases están desbalanceadas, se utilizó el promedio del recall de ambas clases, es decir, la proporción de aciertos en cada clase como función de score para la medición del desempeño de los modelos, el recall se define cómo:

$$recall(clase\ n) = \frac{TP}{TP + FN} \tag{5.15}$$

Donde TP son los verdaderos positivos para la clase n y FN los falsos negativos para la clase n

Por último, se tomó el mejor modelo y se movió el punto de corte a 0.496 para mejorar las métricas.

5.6. Resultados

Con un conjunto de evaluación de 730 llamadas y 2919 llamadas de entrenamiento, las métricas obtenidas fueron las siguientes:

Promedio recall	0.5901
Recall clase Paid	0.6579
Recall clase No paid	0.5223
Exactitud	0.6288
F1 score	0.7356

Matriz de confusión:

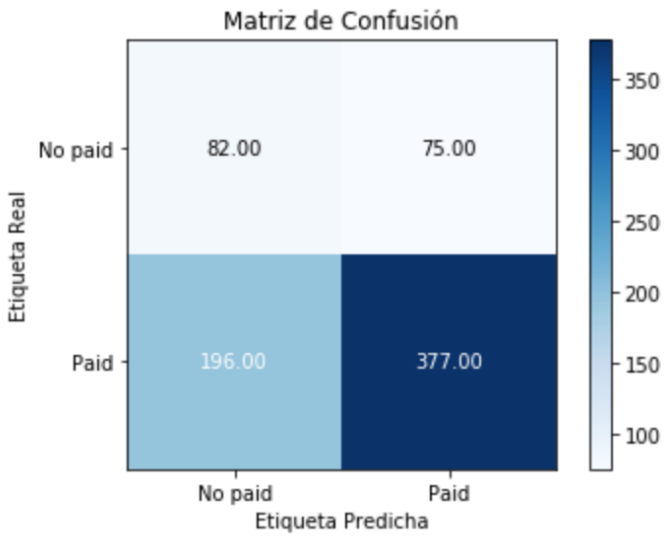

Figura 5.15: Matriz de confusión del modelo

Los hiperparámetros encontrados:

learning_rate	0.5753
n_estimators	1191
max_depth	5
subsample	0.3212
colsample_bytree	0.8056
gamma	5
scale_pos_weight	0.2407

El recall en cada clase es mayor que 0.5. Es decir dado que una llamada sea de un cliente que pagó, la probabilidad de clasificarlo correctamente es de 0.6579, y dada

una llamada de un cliente que no pagó, la probabilidad de clasificarlo correctamente es de 0.5233. Aunque ligeramente, la probabilidad en ambas clases de clasificar correctamente es mayor que de clasificarlo incorrectamente.

Se calculó la curva Roc:

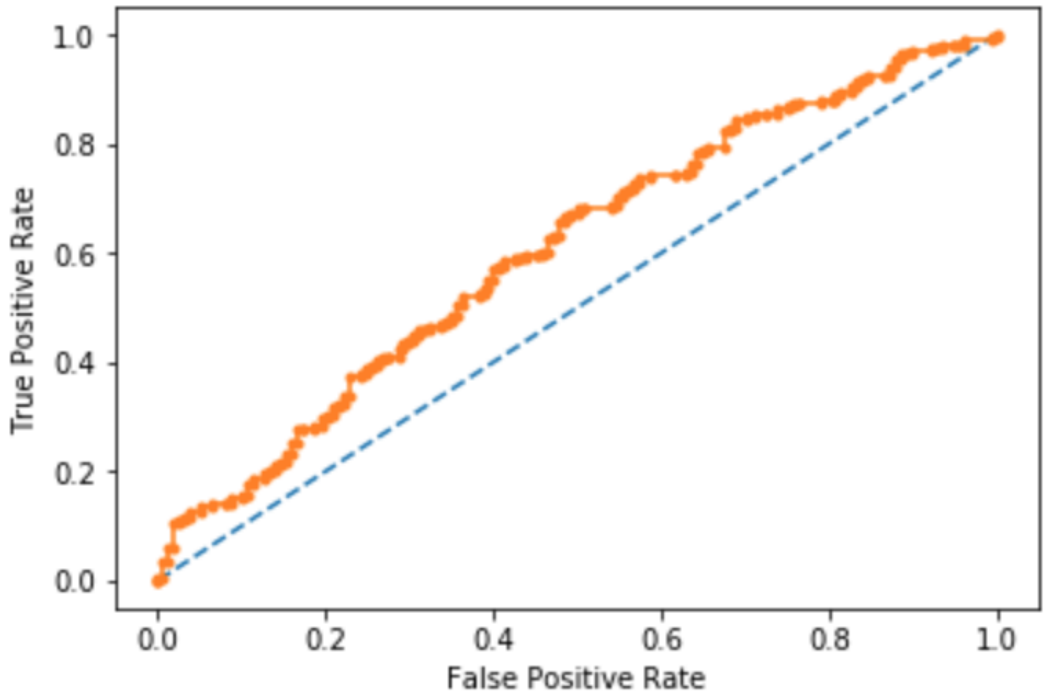

Figura 5.16: Curva ROC

El área bajo la curva es de **0.608**

5.7. Interpretación

Calculando el orden de importancia de las características en el modelo basado en los shap values:

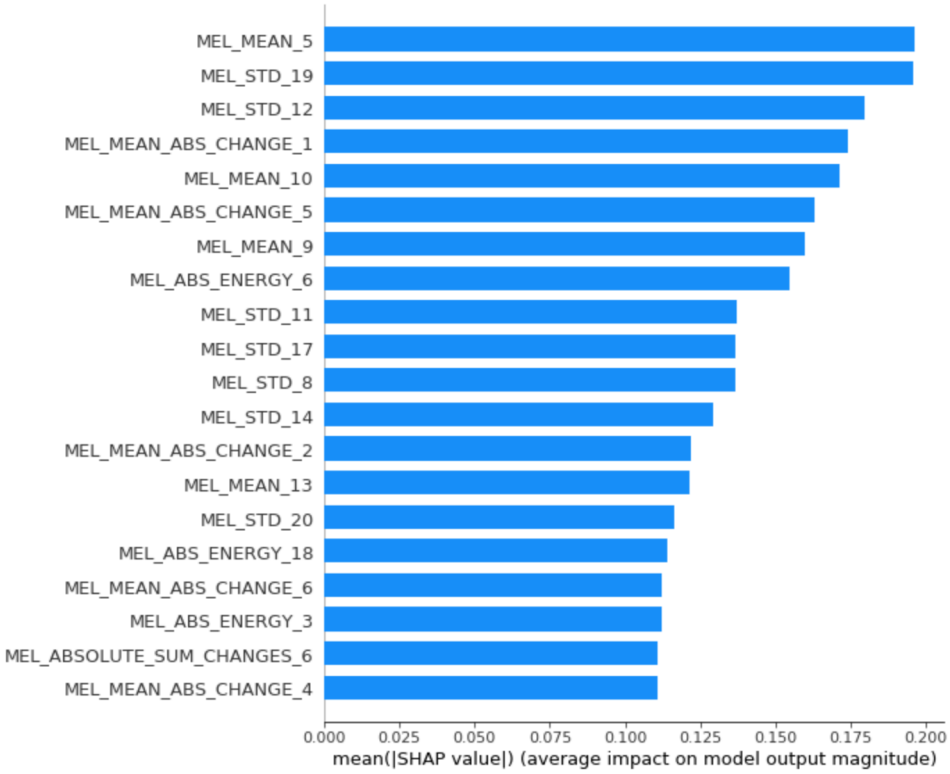

Figura 5.17: Importancia de las características

Las medias y varianzas de algunos coeficientes de MEL resultan ser las características más importantes. La más importante es la media del coeficiente de MEL 5. El coeficiente 5 de MEL equivale aproximadamente a las frecuencias entre 298 y 390 herz. Graficando el impacto de las características más importantes a la salida del modelo, a valores mas altos en el coeficiente 5 de MEL, la salida del modelo tiende a clasificar como 1(si pagó). Otra característica con mucha importancia, es la varianza del coeficiente 19, que va desde los 2757 a 3079 herz.

Las característias más importantes son las relacionadas con los coeficientes de MEL, dejando fuera las características de series de tiempo, y la información de días

de atraso al momento de las llamadas y el número de llamadas.

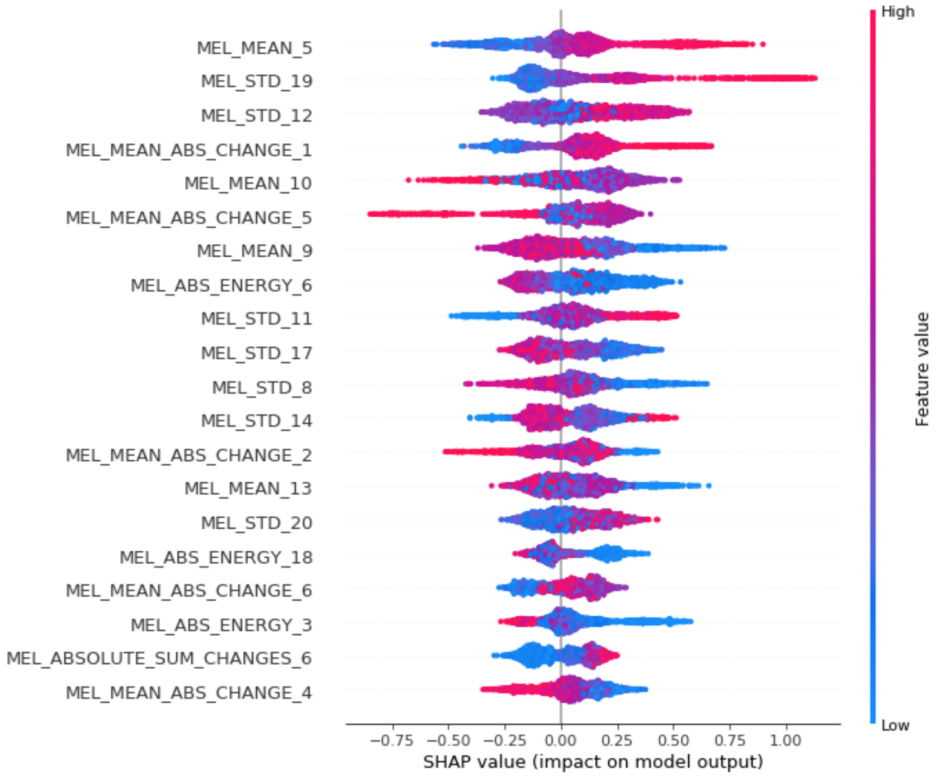

Figura 5.18: Shap Values para modelo de comportamiento de pago

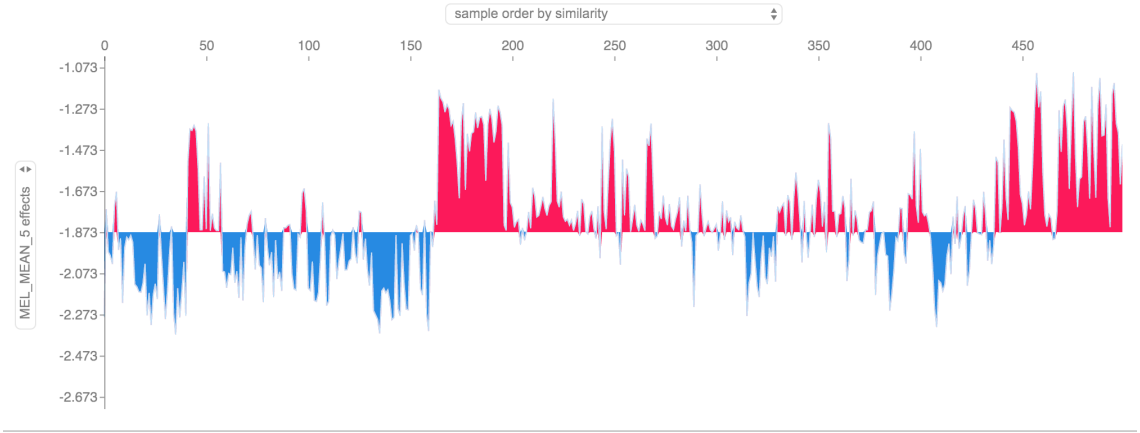

Figura 5.19: Efecto de la media del coeficiente 5 de Mel

Figura 5.20: Efecto de la varianza del coeficiente 12 de Mel

Para asegurarnos que el efecto del coeficiente 5 de MEL en la predicción no está relacionado a algún cambio en los dispositivos electrónicos mediante los cuales se hacen las llamadas de cobranza, se calculó el promedio del coeficiente 5 de MEL por día durante el transcurso de las llamadas de entrenamiento.

En la siguiente gráfica, en el eje x tenemos los días consecutivos donde se hicieron las llamadas, y en el eje y el promedio del coeficiente 5 de MEL de todas las llamadas realizadas ese día.

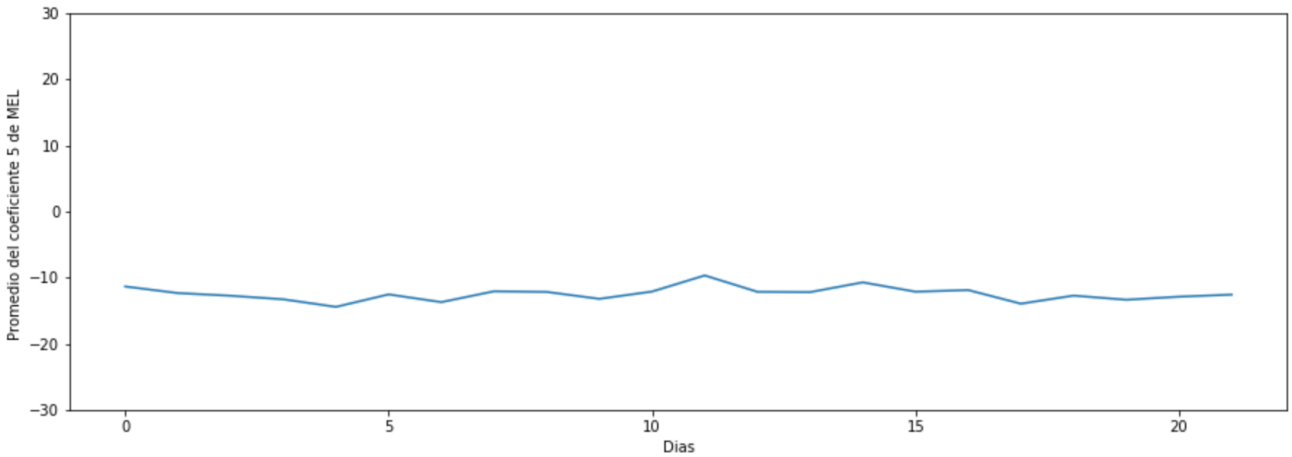

Figura 5.21: Promedio del coeficiente 5 de mel a través del tiempo

No parece existir ninguna tendencia temporal en el coeficiente de MEL 5.

Es interesante saber si el modelo está calibrado, es decir, si el score del modelo corresponde a probabilidades. La curva de calibración del modelo es la siguiente:

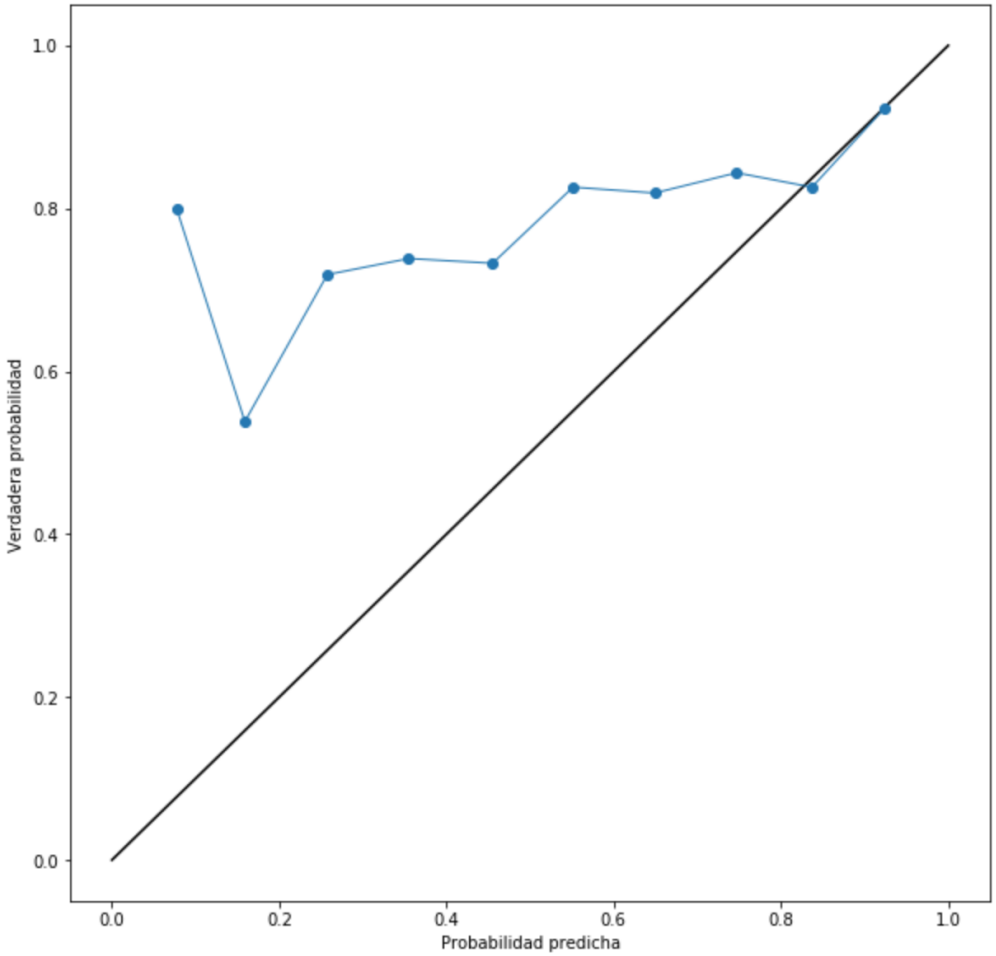

Figura 5.22: Curva de calibración del modelo

Podemos ver que para valores altos en la verdadera probabilidad, el score del modelo corresponde con estos valores. Conforme mas pequeño es el score, mas aljeado está de la verdadera probabilidad. Esta diferencia siempre es subestimando la probabilidad.

Capítulo 6

Conclusiones y trabajo a futuro

6.1. Conclusiones

Diferentes son los objetivos de esta tesis, pero en general, es mostrar técnicas para hacer tratamiento de audio para extraer información relevante, que pueda por si sola dar información de interés, o que sirva para entrenar modelos de machine learning para clasificar o predecir algunas variables de respuesta.

6.1.1. Extracción de características

Dentro de la extracción de características, el audio fue tratado para eliminar silencios, y después calcular coeficientes cepstrales en las frecuencias de Mel y el banco de filtros escalados de Mel. Los primero sirvieron para segmentación de audio y los segundos para clasificación de género. También se utilizaron modelos de aprendizaje profundo para extraer el texto de los audios y construir un clasificador de rol de los personajes de la conversación de los audios. Existen más características que pueden ser extraídas de un audio, pero estas fueron las que brindaron mejores resultados.

6.1.2. Clasificación de género

Para la clasificación de género, se mostró un modelo basado en redes neuronales convolucionales. En esta etapa, el audio se modeló con el banco de filtros escalados

de Mel. Se mostró el desempeño sobre un conjunto de datos en idioma inglés con diversos acentos, mostrando una exactitud por encima de 0.85.

6.1.3. Segmentación de voces

Para la segmentación de las voces en las conversaciones del audio, se utilizó el clasificador de género en el caso que los protagonistas fueran de género diferente, y en caso contrario, los coeficientes cepstrales en la frecuencia de Mel fueron utilizados junto con sus deltas de orden 1 y 2, utilizando un modelo de aprendizaje no supervisado basado en agrupamiento jerárquico.

6.1.4. Clasificador cliente-cobrador

Con el texto extraído de los audios, se extrajeron n-gramas y se aplicó la técnica de TFIDF, esto para extraer características que sirvieron de entrada a un modelo de aprendizaje supervisado basado en árboles llamado XGBoost. El modelo resultó tener una exactitud por encima del 0.98. Se hizo un análisis de interpretabilidad, y se encontró que la extracción de los n-gramas junto con el TFIDF, era capaz de reconocer frases que los cobradores dicen comunmente, haciendo que el modelo se fijara en estas frases para clasificar como cobrador, y la auscencia de estas como indicio de que es cliente.

6.1.5. Modelo de predicción de pago

Para el modelo de predicción de pago, fue imposible extraer el texto de todos los audios debido al costo económico de la API utilizada y el tiempo que llevaría extraerlo para todos los audios utilizados. Apesar de esto, se extrajeron diferentes características de los audios utilizando los coeficientes de MEL, y el audio visto como una serie de tiempo, y se utilizó el algoritmo deep feature synthesis para generar las características. Además se añadió información adicional como el número de días de retraso en el momento de la llamada, y el número de veces que se ha llamado antes de la llamada en cuestión. El modelo logró una exactitud por encima del 0.6 y un

promedio de recall de ambas clases por encima de 0.59.

Aunque estas métricas no son muy altas, son mejor que predecir de manera aleatoria. Se hizo una investigación de interpretabilidad, y se encontró que el coeficiente 5 de Mel tiene una relación fuerte con la decisión del modelo. Además del análisis de intepretabilidad, se hizo un análisis temporal del coeficiente 5 de Mel y un análisis de calibración del modelo.

6.2. Trabajo a futuro

Muchas fueron las ideas que quedaron en el camino, ya sea por el tiempo limitado y su complejidad de ejecución, o porque el trabajo estaba demasiado avanzado para regresar a ejecutar una idea de una etapa ya concluida. Estas ideas se esperan realizar en un trabajo a futuro, algunas de estas son:

- Detección de la contestadora en una llamada
- Detectar cuando la llamada es respondida por una persona diferente a la que se pretende llamar
- Uso de transfer learning con modelos de audios ya entrenados para la extracción de características para la segmentación de voces
- Crecimiento de los conjuntos de entrenamiento, tanto para el clasificador de género, como para el clasificador de cliente-cobrador y el modelo predictor de comportamiento de pago.
- Incluir información adicional para el clasificador de comportamiento de pago
- Incluir características de texto para el modelo del clasificador de comportamiento de pago
- Incluir análisis de sentimiento en las características extraidas del texto

A continuación se muestra un diagrama con el flujo de procesos para el modelo de predicción de comportamiento de pago incluyendo texto. Este trabajo se pretende realizar a futuro.

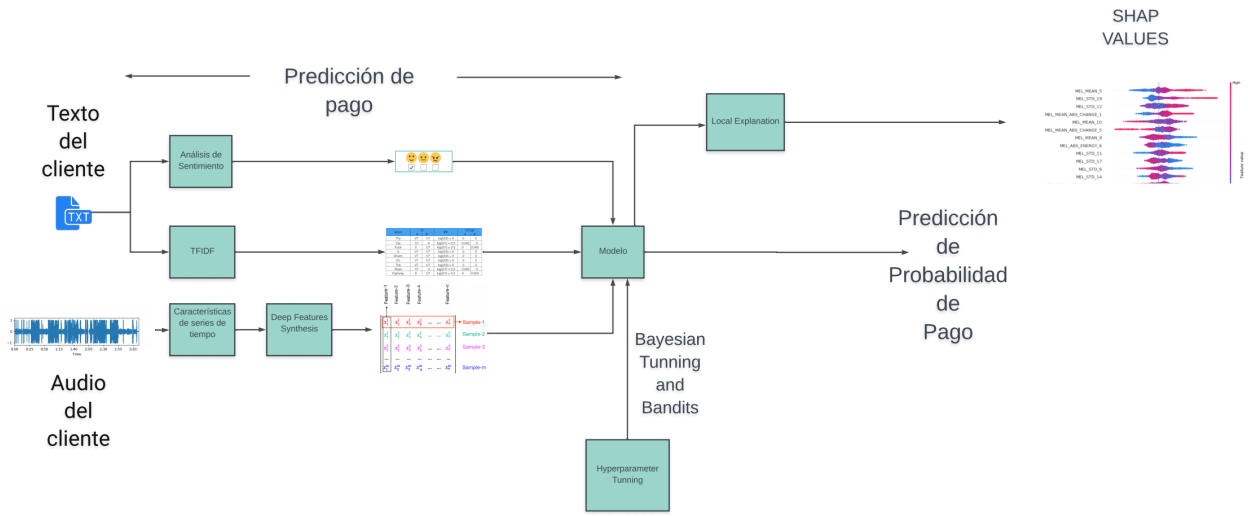

Figura 6.1: Pipeline de modelo final para un trabajo a futuro

Bibliografía

[1] Homayoon Beigi. *Fundamentals of Speaker Recognition*. Springer, 2011.

[2] C. Julian Chen. *Elements of human voice*. World scientific publishing Co. Pte. Ltd, 2016.

[3] Jean Carrive David Doukhan. "An Open-Source Speaker Gender Detection Framework For Monitoring Gender Equality". En: (2018). DOI: "https://ieeexplore.ieee.org/document/8461471".

[4] Laura Gustafson. "Bayesian Tuning and Bandits: An Extensible, Open Source Library for AutoML". M. Eng Thesis. Cambridge, MA: Massachusetts Institute of Technology, mayo de 2018. URL: https://dai.lids.mit.edu/wp-content/uploads/2018/05/Laura_MEng_Final.pdf.

[5] Kalyan Veeramachaneni James Max Kanter. "Deep Feature Synthesis: Towards Automating Data Science Endeavors". En: (2015). DOI: "http://www.jmaxkanter.com/static/papers/DSAA_DSM_2015.pdf".

[6] Carlos Guestrin Marco Tulio Ribeiro Sameer Singh. ""Why Should I Trust You?": Explaining the Predictions of Any Classifiers". En: (2016). DOI: "https://arxiv.org/pdf/1602.04938.pdf".

[7] Michael Nielsen. *Neural Networks and Deep Learning*. 2018.

[8] Davide Rocchesso. *Introduction to Sound Processing*. 2003.

[9] Abraham. Savitzky y M. J. E. Golay. "Smoothing and Differentiation of Data by Simplified Least Squares Procedures". En: (1964). DOI: "https://www.

researchgate.net/publication/270819321_Smoothing_and_Differentiation_of_Data_by_Simplified_Least_Squares_Procedures".

[10] Su-In Lee Scott M. Lundberg. "A Unified Approach to Interpreting Model Predictions". En: (2017). DOI: "https://arxiv.org/pdf/1705.07874.pdf".

[11] Dan Stowell. "Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning". En: (2014). DOI: "https://arxiv.org/abs/1405.6524".

[12] Miron Livny Tian Zhang Raghu Ramakrishnan. "BIRCH: an efficient data clustering method for very large databases". En: (1996). DOI: "https://www2.cs.sfu.ca/CourseCentral/459/han/papers/zhang96.pdf".

[13] Carlos Guestrin Tianqi Chen. "XGBoost: A Scalable Tree Boosting System". En: (2016). DOI: "https://arxiv.org/pdf/1603.02754.pdf".