



Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

**DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA ESTIMADOR DE COBERTURA EN UNA
RED MESH DE COMUNICACIÓN INALÁMBRICA BASADO EN INTELIGENCIA ARTIFICIAL**

Victor Hugo Borrero Herrera

Asesorado por el Ing. Jorge Augusto Balsells Orellana

Guatemala, noviembre de 2024

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



FACULTAD DE INGENIERÍA

**DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA ESTIMADOR DE COBERTURA EN UNA
RED MESH DE COMUNICACIÓN INALÁMBRICA BASADO EN INTELIGENCIA ARTIFICIAL**

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

VICTOR HUGO BORRAYO HERRERA
ASESORADO POR EL ING. JORGE AUGUSTO BALSELLS ORELLANA

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN ELECTRÓNICA

GUATEMALA, NOVIEMBRE DE 2024

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA



NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. José Francisco Gómez Rivera (a. i.)
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Ing. Kevin Vladimir Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIA	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. Julio César Solares Peñate
EXAMINADOR	Ing. Walter Giovanni Alvarez Marroquín
EXAMINADOR	Ing. Julio Rolando Barrios Archila
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA ESTIMADOR DE COBERTURA EN UNA RED MESH DE COMUNICACIÓN INALÁMBRICA BASADO EN INTELIGENCIA ARTIFICIAL

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 19 de abril de 2022.



Victor Hugo Borrero Herrera

Guatemala, 19 de julio de 2024

Área de protocolos
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería
USAC

A quién interese:

Por medio de la presente quiero hacer constar que, yo: Jorge Augusto Balsells Orellana; ingeniero electrónico, número de colegiado activo 17029, he realizado revisiones periódicas y consecutivas al trabajo de graduación del estudiante: Víctor Hugo Borrero Herrera, quien se identifica con el número de carnet 201404406, cuyo título es: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA ESTIMADOR DE COBERTURA EN UNA RED MESH DE COMUNICACIÓN INALÁMBRICA BASADO EN INTELIGENCIA ARTIFICIAL.

Como resultado de las revisiones, se atendieron las sugerencias, las correcciones y ampliaciones al contenido del trabajo. Por lo anterior, al criterio del suscrito, manifiesto mi satisfacción por el producto final presentado el cual cumple con los objetivos trazados en el desarrollo del mismo.

Sin otro particular,



Jorge Augusto Balsells Orellana
Ingeniero Electrónico
Colegiado No. 17,029

Ingeniero Electrónico

Jorge Augusto Balsells Orellana

Colegiado No. 17029

ASESOR



FACULTAD DE INGENIERIA

Guatemala, 29 de julio de 2024

Señor director
Armando Alonso Rivera Carrillo
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC

Estimado Señor director:

Por este medio me permito dar aprobación al Trabajo de Graduación de EPS titulado **DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA ESTIMADOR DE COBERTURA EN UNA RED MESH DE COMUNICACIÓN INALÁMBRICA BASADO EN INTELIGENCIA ARTIFICIAL**, desarrollado por el estudiante **Víctor Hugo Borrayo Herrera**, ya que considero que cumple con los requisitos establecidos.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

ID Y ENSEÑAD A TODOS

A blue ink signature of the name "Ing. Julio César Solares Peñate".

Ing. Julio César Solares Peñate
Coordinador de Electrónica

SIST.LNG.DIRECTOR.24.EIME.2024

El Director de la Escuela de Ingeniería Mecánica Eléctrica de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación del estudiante Victor Hugo Borrero Herrera: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA ESTIMADOR DE COBERTURA EN UNA RED MESH DE COMUNICACIÓN INALÁMBRICA BASADO EN INTELIGENCIA ARTIFICIAL, procede a la autorización del mismo.



Ingeniero Armando Alonso Rivera Carrillo
Director
Escuela de Ingeniería Mecánica Eléctrica

Guatemala, octubre de 2024

Decanato
Facultad e Ingeniería

24189101- 24189102

LNG.DECANATO.OIE.723.2024

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA ESTIMADOR DE COBERTURA EN UNA RED MESH DE COMUNICACIÓN INALÁMBRICA BASADO EN INTELIGENCIA ARTIFICIAL**, presentado por: **Victor Hugo Borrero Herrera** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Ing. José Francisco Gómez Rivera
Decano a.i.

Guatemala, noviembre de 2024



Para verificar validez de documento ingrese a <https://www.ingenieria.usac.edu.gt/firma-electronica/consultar-documento>

Tipo de documento: Correlativo para orden de impresión Año: 2024 Correlativo: 723 CUI: 2563558510308

Escuelas: Ingeniería Civil, Ingeniería Mecánica Industrial, Ingeniería Química, Ingeniería Mecánica Eléctrica, - Escuela de Ciencias, Regional de Ingeniería Sanitaria y Recursos Hídricos (ERIS). Postgrado Maestría en Sistemas Mención Ingeniería Vial. Carreras: Ingeniería Mecánica, Ingeniería Electrónica, Ingeniería en Ciencias y Sistemas. Licenciatura en Matemática. Licenciatura en Física. Centro de Estudios Superiores de Energía y Minas (CESEM). Guatemala, Ciudad

ACTO QUE DEDICO A:

El pueblo	Porque en su seno surge esta mi <i>alma mater</i> , y sobre sus hombros pivotamos los sancarlistas.
Mis padres	Victor Hugo Borrero Santa Cruz y Nora Elizabeth Herrera Mican. Por su amor, guía, buen ejemplo y por siempre estar ahí. Los amo.
Dios	Como un concepto que me acompañó durante algunos años en mi proceso universitario. Y por respeto a mi subjetividad de antaño.
Mis hermanos	Mónica y Josué Borrero Herrera. Porque los hermanos no se eligen, pero no podría no elegirlos siempre.
Mis abuelos	Faustina Mican, Oscar Herrera, Víctor Borrero y Lidia Santa Cruz. Porque son la raíz de dónde vengo, el camino por donde camino y la memoria a la que me aferro.
Mi novia	Luz María Hernández. Por tu compañía, y por congraciarte con mi obstinación de hacer este trabajo como lo hice.
A Bethoven	Mano, cómo no. Me cambiaste la vida chivito.

AGRADECIMIENTOS A:

- A mis padres** Por su apoyo y constancia conmigo. Por la seguridad que se siente al saber que están ahí para mí.
- Universidad de San Carlos de Guatemala** Por cómo te recuerdo a veces, desde mi infancia y en mi ideario que no incluye todas las cosas malas que te pasan.
- Facultad de Ingeniería** Por las lecciones dentro y fuera del aula.
- Jorge Balsells** Por tu paciencia y apoyo generoso en asesorar este trabajo. Porque me aceptaste cambios de tema, de ideas y dilatar la cuestión más allá de lo recomendable.
- Mis hermanos** Mónica y Josué Borrayo Herrera. Por el apoyo y la amistad. Por nuestra infancia compartida.
- Luz Hernández** Por tu apoyo en este trabajo. Por nuestros viajes.
- Mis tíos** Por su apoyo y porque son ejemplo de dedicación y superación.
- Mis amigos** Por lo bueno que es coincidir. Gracias José Pérez y Erick Mendoza, y a todos los demás.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES	VII
LISTA DE SÍMBOLOS	XIII
GLOSARIO	XV
RESUMEN	XXV
OBJETIVOS.....	XXVII
INTRODUCCIÓN	XXIX
1. MARCO TEÓRICO.....	1
1.1. Comunicación inalámbrica.....	1
1.1.1. Electromagnetismo	3
1.1.2. Ondas electromagnéticas	4
1.1.3. Radio frecuencia.....	5
1.1.4. Antena	6
1.1.5. Radio enlace.....	8
1.1.6. Red de comunicación inalámbrica	9
1.1.7. Redes malla (<i>mesh</i>).....	11
1.2. Cobertura de una red inalámbrica	14
1.2.1. Hipótesis	15
1.2.2. Teoría	15
1.2.3. Modelo	16
1.2.4. Modelo matemático	16
1.2.5. Modelo probabilístico.....	16
1.2.6. Área de cobertura	17
1.2.7. Enfoques para el cálculo de cobertura	17
1.2.7.1. Enfoque determinístico	18

1.2.7.2.	Enfoque empírico	19
1.2.7.3.	Enfoque por inteligencia artificial.....	19
1.3.	Inteligencia artificial	20
1.3.1.	Aprendizaje de máquina.....	21
1.3.1.1.	Aprendizaje supervisado	21
1.3.1.2.	Aprendizaje no supervisado	22
1.3.1.3.	Aprendizaje con refuerzo.....	23
1.3.2.	Aprendizaje profundo	23
1.3.3.	Red neuronal artificial.....	24
1.3.4.	Ciencia de datos.....	24
1.3.4.1.	Entendimiento del negocio	25
1.3.4.2.	Minería de datos.....	26
1.3.4.3.	Limpieza de datos	26
1.3.4.4.	Exploración.....	27
1.3.4.5.	Modelaje.....	27
1.3.4.6.	Interpretación de resultados	28
1.4.	Ingeniería de características	28
1.4.1.	Transformación de características	30
1.4.2.	Generación de características.....	30
1.4.3.	Selección de características.....	30
1.4.4.	Aprendizaje de características	31
1.5.	Infraestructura computacional para entrenar IA	32
1.5.1.	Mercado de datos.....	33
1.5.2.	Python	34
1.5.3.	Recursos para ingeniería de características	37
1.5.4.	Computador de placa reducida	37
2.	REVISIÓN DE LA LITERATURA	39
2.1.	Literatura no técnica sobre IA	39

2.1.1.	El lenguaje	44
2.1.2.	Los primeros mitos	47
2.1.3.	Nosotros, una IA	51
2.1.4.	La IA en la ciencia ficción	57
2.2.	Literatura técnica sobre IA.....	65
2.2.1.	La IA en el lenguaje formal	65
2.2.2.	La IA en este trabajo.....	70
2.2.3.	Literatura sobre RF, RF e IA y el problema atendido.....	73
2.2.3.1.	Literatura sobre cobertura de RF.....	75
2.2.3.2.	Avances en la solución del problema ..	77
2.2.3.3.	Literatura sobre RF e IA	78
2.3.	Síntesis sobre la literatura	80
2.3.1.	Ideas principales extraídas de la literatura	82
3.	INFRAESTRUCTURA COMPUTACIONAL	83
3.1.	Diseño e implementación del mercado de datos (<i>Data Mart</i>)..	85
3.1.1.	Fuentes de datos	85
3.1.1.1.	Base de datos del sistema.....	86
3.1.1.2.	Comandos del sistema	89
3.1.1.3.	Archivos	92
3.1.1.4.	APIs externas	95
3.1.2.	Carga en el mercado de datos.....	97
3.1.2.1.	Lectura de archivos XML	98
3.1.2.2.	Limpieza de datos dispositivos de red	100
3.1.2.3.	Limpieza de datos <i>Mesh Clients</i>	108
3.2.	Generación de características	130
3.2.1.	Generación característica de Fresnel.....	130

3.2.2.	Generación característica imagen satelital.....	134
3.3.	Diseño e implementación de interfaz de usuario.....	136
3.3.1.	Interfaz para desarrollo	136
3.3.2.	Interfaz para usuario final	140
4.	DESARROLLO DE MODELOS.....	145
4.1.	Ingeniería de características	146
4.1.1.	Ingeniería de característica imagen satelital	147
4.1.1.1.	Imágenes con modelo pre-entrenado.	151
4.1.1.2.	Imágenes con modelo CNN	158
4.1.1.3.	Imágenes con momentos de color	161
4.1.2.	Ingeniería de característica datos tabulados	169
4.1.2.1.	Imputación de coordenadas	170
4.1.2.2.	Imputación de <i>Path</i> , <i>Layer</i> y CollectorNm.....	172
4.1.2.3.	Imputación de satImg	174
4.1.2.4.	Depurando columnas	175
4.1.3.	Etiquetado de datos	176
4.1.3.1.	Imputación de numberOfNeighbors....	180
4.1.4.	Aumentación de datos.....	187
4.1.5.	Fresnel masivo	190
4.1.6.	División de datos	196
4.1.7.	Transformación de características	199
4.1.7.1.	Hash Encoding	199
4.1.7.2.	Binary Encoding	201
4.1.7.3.	One Hot Encoding	202
4.1.7.4.	MinMax Encoding.....	203
4.1.8.	Balanceo de clases	206
4.2.	Heurística	208

4.3.	Aprendizaje de máquina (<i>machine learning</i>)	212
4.3.1.	Validación cruzada	214
4.3.2.	Búsqueda aleatoria con validación cruzada	214
4.3.3.	Modelos	215
4.3.3.1.	K-Nearest Neighbors (KNN)	216
4.3.3.2.	Random Forest (RF)	216
4.3.3.3.	Gradient Boosting Classifier (GBC) ...	218
4.3.3.4.	Support Vector Machine (SVM)	219
4.3.4.	Experimentos de <i>machine learning</i>	220
4.4.	Aprendizaje profundo (<i>deep learning</i>)	221
4.4.1.	Keras Hyperband.....	222
4.4.2.	Arquitectura de ANN.....	223
4.4.3.	Experimentos de <i>Deep Learning</i>	224
5.	ANÁLISIS DE RESULTADOS	225
5.1.	Los problemas atendidos.....	226
5.2.	Métricas	227
5.2.1.	Matriz de confusión.....	230
5.3.	Tablas de resultados	231
5.4.	Comparación de resultados.....	234
5.4.1.	Comparación de modelos de <i>machine learning</i>	235
5.4.1.1.	Mejores modelos de ML para problema 1.....	236
5.4.1.2.	Mejores modelos de ML para problema 2.....	239
5.4.2.	Comparación de modelos de <i>Deep Learning</i>	243
5.4.2.1.	Mejores modelos de DL para problema 1.....	244

5.4.2.2.	Mejores modelos de DL para problema 2	247
5.4.3.	Comparación general	252
5.4.3.1.	Comparación general problema 1	252
5.4.3.2.	Comparación general problema 2	254
5.4.4.	Análisis de Resultados	256
CONCLUSIONES.....		261
RECOMENDACIONES		265
REFERENCIAS.....		267
APÉNDICES.....		273

ÍNDICE DE ILUSTRACIONES

FIGURAS

Figura 1.	The Zen of Python	36
Figura 2.	Extracción desde base de datos	88
Figura 3.	Extracción de datos de comandos	91
Figura 4.	Extracción de archivos	94
Figura 5.	Extracción desde APIs externas	96
Figura 6.	Lectura de archivos XML	99
Figura 7.	Limpieza de datos equipos de red	101
Figura 8.	Mesh Gateways con información duplicada.....	102
Figura 9.	Mesh Gateways que presentan cambios en el tiempo	102
Figura 10.	Distancias de referencia Mesh Routers	103
Figura 11.	<i>Boxplot</i> distancias referencia Mesh Routers.....	104
Figura 12.	Visualización de outliers por agrupación de MG	105
Figura 13.	Visualización de outliers por agrupación de MR	106
Figura 14.	Visualización de totalidad de MR con outliers.....	107
Figura 15.	Limpieza de datos Mesh Clients	109
Figura 16.	MC sin coordenadas inicial	110
Figura 17.	MC sin categorizar	112
Figura 18.	MC sin coordenadas luego de incluir registros futuros	113
Figura 19.	MC sin coordenadas luego de incluir fuentes externas	114
Figura 20.	MC sin coordenadas en conjunto de datos principal, final.....	115
Figura 21.	Anomalías detectadas por IF_1	117
Figura 22.	Anomalías detectadas por IF_2.1	118
Figura 23.	Anomalías detectadas por IF_2.2	119

Figura 24.	Anomalías detectadas por IF_3.1	120
Figura 25.	Anomalías detectadas por IF_3.2.....	121
Figura 26.	Anomalías detectadas por IF_3.3.....	122
Figura 27.	Anomalías detectadas por IF_4.1	123
Figura 28.	Anomalías detectadas por IF_4.2.....	124
Figura 29.	Anomalías detectadas por IF_4.3.....	125
Figura 30.	Anomalías detectadas por IF_4.4.....	126
Figura 31.	Coordenadas corregidas 1	127
Figura 32.	Coordenadas corregidas 2	128
Figura 33.	Coordenadas corregidas 3	129
Figura 34.	Característica de Fresnel	131
Figura 35.	Característica de Fresnel	133
Figura 36.	Característica de imagen satelital.....	135
Figura 37.	Mapa interactivo para notebook	139
Figura 38.	Arquitectura en Azure para interfaz gráfica	141
Figura 39.	Arquitectura en Azure para interfaz gráfica	142
Figura 40.	Muestra aleatoria de característica Imagen Satelital	147
Figura 41.	Anomalías detectadas en característica imgSat.....	149
Figura 42.	Muestra del conjunto de datos leídos con TF	151
Figura 43.	Inercia vs K – modelo pre-entrenado.....	153
Figura 44.	Silueta vs K – modelo pre-entrenado	154
Figura 45.	Modelo pre-entrenado Kmeans Anomalías	155
Figura 46.	Imágenes en áreas costeras pre-entrenado Kmeans.....	156
Figura 47.	Histograma de grupos para característica pre-entrenada	157
Figura 48.	Inercia vs K – modelo CNN	158
Figura 49.	Silueta vs K – modelo CNN	159
Figura 50.	Ejemplo de cluster con característica CNN	160
Figura 51.	Anomalías detectadas con característica CNN	161
Figura 52.	Inercia vs K – modelo CM	162

Figura 53.	Silueta vs K – modelo CM.....	163
Figura 54.	Histograma de grupos para característica CM.....	164
Figura 55.	Anomalías agrupadas en cluster 3 con <i>color moments</i>	165
Figura 56.	Anomalías agrupadas en cluster 7 con <i>color moments</i>	166
Figura 57.	Imagenes no anómalas en cluster 0 con <i>color moments</i>	167
Figura 58.	Anomalías Isolation Forest con <i>color moments</i>	168
Figura 59.	Anomalías Isolation Forest con color moments	169
Figura 60.	Imputación de coordenadas basada en saltos.....	171
Figura 61.	Imputación de datos basada en saltos	173
Figura 62.	Imputación de imagen satelital	174
Figura 63.	Histograma de comunicación MG	178
Figura 64.	Histograma de comunicación MR	179
Figura 65.	Histograma de comunicación MC	179
Figura 66.	Histograma de comunicación MG sin <i>numberOfNeighbors</i>	181
Figura 67.	Histograma de comunicación MR sin <i>numberOfNeighbors</i>	181
Figura 68.	Histograma de comunicación MC sin <i>numberOfNeighbors</i>	182
Figura 69.	Distribución de <i>numberOfNeighbors</i> para no comunican.....	183
Figura 70.	Mapa de <i>numberOfNeighbors</i> registros no comunican.....	184
Figura 71.	KNN con MC sin comunicación	185
Figura 72.	KNN con MC con comunicación	186
Figura 73.	KNN con MG-MR	187
Figura 74.	Conjunto de datos final con datos reales	188
Figura 75.	Generación de datos sintéticos.....	189
Figura 76.	Conjunto de datos final con datos sintéticos agregados.....	190
Figura 77.	Distribución de alturas para equipos de la red mesh	191
Figura 78.	Comprobación de característica Fresnel con obstrucciones.....	193
Figura 79.	Comprobación de característica Fresnel sin obstrucciones.....	194
Figura 80.	Primera división de conjunto de datos	196
Figura 81.	Segunda división de conjunto de datos	197

Figura 82.	División final del conjunto de datos original.....	198
Figura 83.	Distribución de variables parte 1 de 9	205
Figura 84.	Distribución de variables parte 1 de 9	206
Figura 85.	Conjunto de entrenamiento Undersampled	207
Figura 86.	Conjunto de entrenamiento Oversampled	208
Figura 87.	Puntos de interés alrededor de equipo	209
Figura 88.	Obstrucción de la zona de Fresnel	210
Figura 89.	Línea vista	211
Figura 90.	Ejemplo de matriz de confusión	230
Figura 91.	Mejor matriz de confusión ML para P1 según Accuracy.....	237
Figura 92.	Mejor matriz de confusión ML para P1 según F1 Score Macro ..	238
Figura 93.	Mejor matriz de confusión ML para P1 según MPCA	239
Figura 94.	Mejor matriz de confusión ML para P2 según Accuracy.....	240
Figura 95.	Mejor matriz de confusión ML para P2 según F1 Score Macro ..	241
Figura 96.	Mejor matriz de confusión ML para P2 según MPCA	242
Figura 97.	Mejor matriz de confusión DL para P1 según Accuracy	245
Figura 98.	Mejor matriz de confusión DL para P1 según F1 Score Macro ..	246
Figura 99.	Mejor matriz de confusión DL para P1 según MPCA	247
Figura 100.	Mejor matriz de confusión DL para P2 según Accuracy	248
Figura 101.	Mejor matriz de confusión DL para P2 según F1 Score Macro ..	249
Figura 102.	Mejor matriz de confusión DL para P2 según MPCA	250
Figura 103.	Mejor matriz de confusión DL para P2 según inspección.....	251
Figura 104.	Gráfica Accuracy para problema 1	253
Figura 105.	Solución al problema 1	254
Figura 106.	Gráfica MPCA para problema 2	255
Figura 107.	Solución al problema 2	256

TABLAS

Tabla 1.	Bandas de frecuencia reconocidas internacionalmente.....	5
Tabla 2.	Distintos estándares para redes de comunicación	10
Tabla 3.	Modelo IF_1	116
Tabla 4.	Modelo IF_2.....	118
Tabla 5.	Modelo IF_3.....	120
Tabla 6.	Modelo IF_4.....	123
Tabla 7.	Columnas de datos depurados	175
Tabla 8.	Cantidad de registros luego de depuración	177
Tabla 9.	Ejemplo de característica codificada con MD5	200
Tabla 10.	Ejemplo de característica codificada con Binary Encoding.....	202
Tabla 11.	Ejemplo de característica codificada con One Hot Encoding....	203
Tabla 12.	Experimentos <i>machine learning</i>	221
Tabla 13.	Experimentos <i>deep learning</i>	224
Tabla 14.	Tabulación de resultados ML problema 1	232
Tabla 15.	Tabulación de resultados ML problema 2	232
Tabla 16.	Tabulación de resultados DL problema 1	233
Tabla 17.	Tabulación de resultados DL problema 2	234
Tabla 18.	Mejor modelo de ML para P1 según Accuracy	236
Tabla 19.	Mejor modelo de ML para P1 según F1 Score Macro	237
Tabla 20.	Mejor modelo de ML para P1 según MPCA.....	238
Tabla 21.	Mejor modelo de ML para P2 según Accuracy	240
Tabla 22.	Mejor modelo de ML para P2 según F1 Score Macro	241
Tabla 23.	Mejor modelo de ML para P2 según MPCA.....	242
Tabla 24.	Mejor modelo de DL para P1 según Accuracy.....	244
Tabla 25.	Mejor modelo de DL para P1 según F1 Score Macro	245
Tabla 26.	Mejor modelo de DL para P1 según MPCA.....	246
Tabla 27.	Mejor modelo de DL para P2 según Accuracy.....	248

Tabla 28.	Mejor modelo de DL para P2 según F1 Score Macro.....	249
Tabla 29.	Mejor modelo de DL para P2 según MPCA.....	250
Tabla 30.	Mejor modelo de DL para P2 según inspección	251

LISTA DE SÍMBOLOS

Símbolo	Significado
----------------	--------------------

%	Porcentaje
---	------------

GLOSARIO

ACC	Accuracy.
Algoritmo	Conjunto de instrucciones secuenciales con cierta lógica que resuelve una tarea específica.
Altimetría	Altura sobre el nivel del mar para un par de coordenadas geográficas.
AMI	Advanced Metering Infrastructure.
Anaconda	Plataforma para gestión de entornos de programación.
ANN	Artificial Neural Network.
Anomalía	Registro en conjunto de datos fuera del comportamiento normal.
Antena	Dispositivo para transmisión y recepción de ondas electromagnéticas.
API	Application Programming Interface.
Aumentación	Proceso de aumentar la cantidad de datos por medio de la generación de nuevos datos sintéticos.

Azure	Plataforma de servicios en la nube de Microsoft.
Boxplot	Gráfico que muestra distribución de un conjunto de datos a través de cuartiles, medias y rango intercuantil.
Característica	Variable o atributo que describe algún aspecto del fenómeno representado en el conjunto de datos en cuestión.
Ciencia de datos	Disciplina que combina técnicas matemáticas, programación y conocimiento del negocio para extraer información útil de datos.
Clasificador	Algoritmo que asigna categorías o etiquetas a los registros de datos basados en patrones presentes en los datos.
CNN	Convolutional Neural Network.
Cobertura	Extensión geográfica donde es posible establecer una comunicación efectiva por los equipos de una red de comunicaciones.
Container	(Contenedor). Entorno de <i>software</i> ligero y aislado, que permite incluir dependencias para ejecutar una aplicación.

Data Lake	(Lago de datos). Repositorio de almacenamiento de datos en formato de origen.
Deep Learning	(Aprendizaje profundo). Paradigma de aprendizaje de máquina enfocado principalmente en el uso de redes neuronales artificiales.
Distribución	Distribución de probabilidad de un conjunto de datos. De manera empírica representa la frecuencia de ocurrencia de los posibles valores de una variable.
DL	<i>Deep Learning.</i>
Docker	Plataforma para desarrollar, enviar y ejecutar contenedores.
Dropout	Técnica de regularización de redes neuronales que consiste en desactivar neuronas de manera aleatoria.
F1	F1 Score.
Gateway	Dispositivo final en la red de comunicaciones malla que es la frontera de esta con otro tipo de red.
GBC	Gradient Boosting for Classification.
Git	Sistema de control de versiones especialmente enfocado en código.

Github	Plataforma de desarrollo colaborativo basada en Git.
Google	Empresa tecnológica multinacional especializada en servicios y productos relacionados con la Internet.
<i>Gradient</i>	(Gradiente). Vector que indica la dirección del mayor cambio en una función.
Hash	Valor obtenido luego de la aplicación de un algoritmo con el mismo nombre que reduce los datos de entrada a una cadena de tamaño fijo.
Heurística	Enfoque no riguroso basado en métodos aproximados o reglas empíricas para dar solución a un problema.
Histograma	Gráfico que representa la distribución de una variable de manera discreta dividiendo el rango en intervalos.
IA	Inteligencia Artificial.
IF	Isolation Forest.
Imblearn	Paquete de Python enfocado al manejo de datos desbalanceados.
Imputación	Técnica de asignación de un valor a una característica para registros anormales o vacíos, basada en los datos disponibles.

Instancia	Ejecución particular de un algoritmo con determinada configuración dentro de un amplio número de posibles configuraciones.
Inteligencia artificial	Imitación del comportamiento inteligente del artífice de parte del artificio creado. Imitación del comportamiento humano por una máquina.
IpyWidget	Paquete de Python que permite la creación de elementos interactivos en un cuaderno de Jupyter.
Isolation Forest	(Bosque de aislamiento). Algoritmo que permite la detección de anomalías al aislarlas del resto de datos.
JavaScript	Lenguaje de programación enfocado en la creación de contenido <i>web</i> dinámico.
Keras	Biblioteca que permite el desarrollo de redes neuronales a alto nivel.
KerasTuner	Biblioteca basada en Keras que permite la optimización automática de hiper parámetros.
Kernel	Función utilizada en distintos algoritmos que permite la transformación del espacio de entrada de las características a un espacio distinto.

KMeans	(KMedias). Algoritmo de agrupamiento que partitiona los datos en K grupos basándose en centroides y distancias.
KNN	K-Nearest Neighbors.
Linux	Sistema operativo de código abierto basado en Unix.
MC	Mesh Client.
Mesh	(Malla). Topología de red basada en la interconexión de los equipos de modo que existen varias rutas redundantes para la transferencia de datos.
MG	Mesh Gateway.
Microsoft	Empresa tecnológica multinacional que desarrolla <i>software</i> , <i>hardware</i> y servicios. Entre ellos servicios en la nube.
ML	<i>Machine learning.</i>
MLflow	Plataforma de código abierto que permite gestionar el ciclo de vida de los modelos de aprendizaje automático.
Modelo	Representación simplificada de uno o varios aspectos de la realidad que permiten su análisis y predicción.

MPCA	Mean Per Class Accuracy.
MR	Mesh Router.
Notebook	(Cuaderno). Entorno de desarrollo interactivo basado en la <i>web</i> .
Nube	Infraestructura computacional que proporciona servicios a través de internet.
Oversampled	(Sobre muestreo). Técnica de procesamiento de datos para lidiar con el desbalance de clases, por medio del sobre muestreo de las clases minoritarias.
Path	(Camino). Secuencia de equipos que comprenden los distintos saltos desde un equipo de origen hasta su respectivo Mesh Gateway.
Plotly	Biblioteca con soporte para Python enfocada en la visualización de datos.
Raspberry Pi	Computadora de placa reducida que permite la ejecución de aplicaciones con bajo costo energético y de <i>hardware</i> .
Rasterio	Paquete de Python para trabajar con datos geoespaciales.
RF	Radio Frequency.

RF	Random Forest.
Router	(Enrutador). Dispositivo con la capacidad de gestionar las distintas rutas por las que se realiza la transferencia de datos en una red.
Scikit Learn	Paquete de Python enfocado en el aprendizaje automático.
SFTP	Secure File Transfer Protocol.
SQL	Structured Query Language.
Streamlit	Marco de trabajo de Python enfocado en la creación de interfaces <i>web</i> de manera sencilla.
SVC	Support Vector Classifier.
SVM	Support Vector Machine.
TensorFlow	Biblioteca de código abierto creada por Google para construir y entrenar modelos de aprendizaje automático y redes neuronales, con manejo eficiente de las operaciones numéricas.
THRES	Threshold (Umbral).

<i>Undersample</i>	(Submuestreo). Técnica de procesamiento de datos para lidiar con desbalance de clases, por medio de submuestreo de las clases mayoritarias.
WebApp	Aplicación de <i>software</i> que se ejecuta en un servidor <i>web</i> y que es accesible a través de Internet.
WiFi	Tecnología inalámbrica estandarizada por el IEEE que permite la comunicación de diversos dispositivos por medio de enrutadores que acceden a la Internet.
WMN	Wireless Mesh Network.

RESUMEN

La idea de que los humanos podamos fabricar objetos que se comporten de manera inteligente es bastante antigua, está presente ya en los primeros mitos, hemos filosofado sobre su posibilidad y fantaseado con ficciones que desbordan nuestra imaginación en escenarios benignos y siniestros para nosotros. Hoy en día, la revolución de las máquinas que piensan se vislumbra menos a ciencia ficción que a realismo. Por supuesto, nuestras ficciones siguen siendo válidas, dado que todavía no llegamos, y dado que no parecemos cansarnos de contar historias.

El marco de este trabajo se enfoca en el aprendizaje de máquina, que ahora está en auge. Se utilizan datos de un problema específico, datos relacionados con la cobertura de una red de comunicaciones inalámbrica con topología malla. La implementación de la solución implicó realizar una serie de tratamiento a los datos, para luego enriquecerlos con fuentes de datos externas y que ampliaban la información disponible para entrenar los modelos en busca de obtener una solución.

Los algoritmos aprendieron a dar solución a los dos problemas planteados, tanto en aprendizaje de máquina como en aprendizaje profundo, los resultados fueron satisfactorios. Los algoritmos consiguieron por medio de los datos, aprender patrones internos del fenómeno de estudio que les permitieron brindar respuestas acertadas para datos no vistos. Teniendo con esto el escenario de una IA que aprende a resolver un problema.

OBJETIVOS

General

Utilizar modelos de inteligencia artificial con la finalidad de predecir el estado de la comunicación de equipos de una red de radiofrecuencia con topología malla.

Específicos

1. Elaborar una revisión de la literatura donde se abarque desde el tema general de la Inteligencia Artificial hasta el específico de la contextualización del trabajo realizado en el actual desarrollo de aplicaciones para el aprendizaje de máquina en los distintos tipos de redes de comunicación inalámbrica, principalmente la red con topología malla.
2. Crear un ambiente de *software* y *hardware* que permita recabar datos de una red inalámbrica con topología malla para ser utilizados por las distintas etapas del diseño e implementación de la solución, haciendo uso de un ordenador de placa reducida como dispositivo de obtención de datos.
3. Diseñar e implementar un mercado de datos (*data mart*) que permita alojar la información que utilizarán los modelos, así como tener un control de los cambios que sucedan en la red a través del tiempo.
4. Utilizar distintas técnicas de ingeniería de características para el enriquecimiento de los datos previo a ser utilizados para entrenar los

modelos de inteligencia artificial (aprendizaje de máquina y aprendizaje profundo).

5. Diseñar e implementar múltiples experimentos con distintos algoritmos de aprendizaje de máquina (*machine learning*) y aprendizaje profundo (*deep learning*). Almacenar los resultados de una forma estándar para su posterior análisis.
6. Analizar los resultados de los distintos experimentos implementados. Comparar los modelos a distintos niveles y seleccionar aquellos con mejor desempeño según métricas comunes establecidas. Interpretar resultados.

INTRODUCCIÓN

Parece una constante casi inevitable la de querer conocer el porqué de las cosas. La humanidad ha llenado cientos y miles de bibliotecas y espacios ciberneticos para practicar la labor de dar respuestas a nuestras interrogantes. En la literatura consultada para el desarrollo de este trabajo está presente la persistencia humana de querer superar su condición de ignorancia. Ante esto hemos creado distintos modelos del mundo, que nos permiten explicarlo bajo ciertas asunciones.

El término inteligencia artificial está en auge en nuestros días. Al intentar rastrear sus orígenes fue inevitable retroceder hasta nuestros mitos más antiguos. En ellos ya se vislumbra lo que es la creación de una inteligencia artificial, producto de la labor de otra inteligencia. Con el paso del tiempo la idea se ha refinado y asociado directamente con la informática. Hoy, cuando nos referimos a una inteligencia artificial, nos referimos a un *software* que es ejecutado en una máquina de cómputo.

La formalización del concepto ha creado el paradigma actual donde el aprendizaje de máquina ofrece resultados en muchos casos significativamente mejores que los paradigmas anteriores. Esta es la motivación detrás de este trabajo. Más allá de la indagación en la idea de lo que es una IA en el capítulo 2, el resto del trabajo se enfoca en la utilización de las técnicas de tratamiento de datos y la aplicación de ciertos algoritmos que aprenden de los datos.

El problema presentado a los algoritmos está relacionado con una red de comunicaciones con topología malla. Específicamente lo que se buscó es conseguir que los algoritmos determinaran el estado de comunicación de un equipo dentro de la red, en un caso basándose en sus registros históricos y en otro ante la presencia de equipos totalmente nuevos. Aunque se dice que el ingenio humano no es el que da solución al problema, puesto que los algoritmos llegan a la solución. En el intermedio de este trabajo hay dos capítulos dedicados a preparar los datos antes de que sean utilizados para el entrenamiento.

Luego de varias horas de entrenamiento de múltiples instancias de los algoritmos que por decisión de diseño se incluyeron, se consiguieron resultados que fueron analizados y de los cuales se seleccionaron aquellos algoritmos que de mejor manera dieron solución al problema. Para que la selección de estos fuera lo más objetiva posible, se establecieron ciertas métricas comunes que permitieron la comparación y posterior selección.

1. MARCO TEÓRICO

El marco de referencia de este y todos los trabajos es sin duda una variedad que implica una serie de conocimientos sucesivos. Para propósitos de este trabajo se presenta la siguiente división del marco teórico:

- Comunicación inalámbrica
- Cobertura de una red inalámbrica
- Inteligencia artificial
- Ingeniería de características
- Infraestructura computacional para entrenar IA

La misma no pretende abarcar los conocimientos generales, sino delimitar las principales áreas que están involucradas en el desarrollo del trabajo. Sabrá entenderse que detrás de ellas hay un marco teórico de miles de años de historia y progreso humano.

1.1. Comunicación inalámbrica

La comunicación aparece constantemente en la naturaleza. La transmisión de un mensaje de un ser a otro. Para los humanos la necesidad de comunicación ha permitido la creación de redes que logran la comunicación entre destinos bastante remotos.

La comunicación inalámbrica juega un papel principal en la evolución de las comunicaciones. Permite enviar mensajes a grandes distancias y elimina la

necesidad de estar conectados por medio de un cable a un canal que transporte la comunicación proveniente desde el origen del mensaje.

El medio compartido sobre el cual se transmitían los mensajes inalámbricos eran la atmósfera terrestre y luego el espacio. Las investigaciones de Heinrich Hertz comprobaron empíricamente las predicciones de la Teoría Electromagnética que había consolidado James Maxwell y que predecían que la luz era una onda electromagnética y su velocidad correspondía a la de la luz, y que está a su vez cambiaba dependiendo del medio en que viajaba.

A principios del siglo pasado comenzaron las aplicaciones de uso civil que permitían la comunicación entre dos o varios puntos. A mediados del mismo siglo se popularizó la comunicación inalámbrica y comenzó la comercialización de los dispositivos que permitían aprovecharla. La tecnología inalámbrica y la telefonía permitieron la creación de redes más complejas, donde muchas personas podían interactuar gracias a una central telefónica inalámbrica.

La llegada del microchip y la rápida evolución de las computadoras permitieron que los controles de las redes de comunicación se volvieran digitales. Lo que a su vez resultó en una mayor eficiencia de la red, aumentando así sus capacidades de manejo de canales de voz, pero también creando la posibilidad de transmitir otros tipos de información.

En la actualidad las redes inalámbricas son algo común y se ha vuelto indispensable en la realización de muchas tareas, la transmisión de datos digitales a través de las redes inalámbricas, propiciaron el auge de los teléfonos inteligentes. Existe también una tendencia a conectar la mayor cantidad de elementos posibles a una red inmensa. Ya sea el internet de las cosas o algún

otro concepto similar, el camino por el que vamos es el de una realidad hiperconectada.

1.1.1. Electromagnetismo

Es el nombre que se le da a una de las cuatro interacciones fundamentales de la naturaleza. Como el nombre lo indica, este fenómeno incluye tanto al eléctrico como al magnético. Como parte del estudio de ambos se determinó una relación de causa y efecto entre sí, de tal modo que se conceptualiza el campo electromagnético con una componente eléctrica y magnética perpendiculares entre sí.

Los egipcios y los griegos conocieron la electricidad, y estos últimos son a quienes se debe el nombre. Pero fue hasta en el siglo XVII cuando se inició un estudio científico de la misma. Los aportes de Charles-Agustín de Coloumb, Johann Carl Friederich Gauss, André Ampére y Michael Faraday, entre otros, fueron condensados y teorizados por James Maxwell, quien introdujo los conceptos de campo y corriente de desplazamiento, lo que dio paso a una unificación del fenómeno eléctrico y magnético, en el electromagnético. Junto con la fuerza de Lorentz, las cuatro ecuaciones de Maxwell describen cualquier fenómeno electromagnético en la escala macroscópica.

Inicialmente las ecuaciones de Maxwell no eran cuatro, fue gracias al trabajo subsiguiente de varias personas que se perfeccionó la teoría clásica del electromagnetismo. Con los descubrimientos relacionados con la mecánica cuántica se vio que la teoría clásica era insuficiente en un régimen de longitudes ínfimas. Ante tal situación, el estudio del fenómeno electromagnético se ha adaptado y ha surgido la electrodinámica cuántica como una forma de describir tal interacción en el régimen cuántico. Esta teoría dio lugar a la unificación de

interacción nuclear débil y la interacción electromagnética, bajo lo que sea plantea en el modelo electrodébil.

1.1.2. Ondas electromagnéticas

Dentro de las soluciones a las ecuaciones de Maxwell, aparecen soluciones especiales que describen las ondas electromagnéticas. Estas son perturbaciones producidas en el campo electromagnético que viajan a través del espacio a una velocidad que depende del medio en que se propague y que para el espacio vacío corresponde a la velocidad de la luz (C).

En un medio no ideal, la constitución de los materiales influye significativamente en la forma en que se propagan las ondas electromagnéticas. La permitividad eléctrica y la permeabilidad magnética son valores que identifican al medio. Estos valores influyen directamente en la atenuación de las ondas electromagnéticas así como, al hacer uso de vectores asociados a estas magnitudes, a describir matemáticamente la reflexión, absorción y refracción de las ondas electromagnéticas.

Las ondas electromagnéticas se caracterizan por tener una amplitud, una fase, una longitud de onda y una frecuencia asociadas, de lo cual dependerá su propagación, atenuación y energía. La radiación electromagnética es llamada de forma distinta dependiendo de su longitud de onda, el espectro de longitudes de onda va desde el orden de kilómetros hasta distancias del tamaño del núcleo atómico, siendo las longitudes de onda mayores las que tienen una menor energía y las que se utilizan para aplicaciones de comunicación. En el otro extremo del espectro aparecen los rayos gamma y los rayos cósmicos, con mucha mayor energía.

1.1.3. Radio frecuencia

Se conoce así a la porción del espectro electromagnético que se encuentra aproximadamente debajo de 1 THz de frecuencia. Esto es, frecuencias menores que aquellas categorizadas como infrarrojas.

Esta sección del espectro electromagnético es la que más ampliamente se utiliza para las comunicaciones inalámbricas, de ahí que el término —Radio frecuencia— sea referido indiscriminadamente tanto al rango de frecuencias indicado como a distintas actividades asociadas con las comunicaciones inalámbricas.

Así como este término representa una de las divisiones del espectro electromagnético, este a su vez es dividido en distintos intervalos que denotan el tipo correspondiente de frecuencia, mismas que a su vez tienen asociadas ciertas tecnologías de comunicación inalámbrica.

La siguiente tabla muestra la categorización vigente en el país, sobre la manera de separar el espectro de las ondas de radio frecuencia, misma que es tomada de convenciones internacionales.

Tabla 1.

Bandas de frecuencia reconocidas internacionalmente

Banda No.	Símbolo	Gama de frecuencias	Dimensional	Subdivisión métrica
12	...	300 – 3000	GHz	Decimilimétricas
11	EHF	30 – 300	GHz	Milimétricas
10	SHF	3 – 30	GHz	Centimétricas
9	UHF	300 – 3000	MHz	Decimétricas

Continuación de la Tabla 1.

Banda No.	Símbolo	Gama de frecuencias	Dimensional	Subdivisión métrica
8	VHF	30 – 300	MHz	Métricas
7	HF	3 – 30	MHz	Decamétricas
6	MF	300 – 3000	KHz	Hectométricas
5	LF	30 – 300	KHz	Kilométricas
4	VLF	3 – 30	KHz	Miriamétricas

Nota. La tabla muestra información referente a las bandas de frecuencias para comunicación inalámbrica. Obtenido de la Superintendencia de Telecomunicaciones (2022). *Bandas de frecuencias.* (<https://sit.gob.gt/gerencia-de-frecuencias/frecuencias/bandas-de-frecuencias/>), consultado el 1 de abril de 2022. De dominio público.

1.1.4. Antena

Se puede definir a una antena como un medio que permite la radiación o recepción de ondas de radio. Siendo que las ondas de radio son perturbaciones en el campo electromagnético y que dichas perturbaciones surgen como variaciones en el tiempo de las magnitudes de los campos eléctricos y magnéticos, una antena es un medio que permite perturbar el campo electromagnético externo a fin de que dicha perturbación sea captada por otra antena.

Una onda electromagnética que viaja externa al circuito no puede ser directamente introducida a los circuitos de comunicación, por tal motivo, otra función básica de la antena es la de convertir las perturbaciones del campo electromagnético en señales eléctricas de corriente y voltaje o viceversa. La onda electromagnética en el espacio libre es recibida por la antena, que a su vez inyecta la señal producida a través de un conductor, que puede considerarse

como una línea de transmisión, hacia los circuitos que ahora pueden lidiar con las señales eléctricas.

Existen una variedad de antenas que están pensadas para distintas aplicaciones. Según sus características físicas y electromagnéticas pueden ofrecer mayor ventaja para determinados escenarios. Como el funcionamiento de una antena está íntimamente ligada a su forma física, han existido y existen campos de investigación que se dedican al desarrollo de nuevos tipos de antena.

Además de su aplicación y forma física, las antenas pueden identificarse por ciertos parámetros de antena, dentro de los cuales, están los siguientes:

- Directividad: razón entre la intensidad de radiación entregada por la antena en una dirección y la intensidad de radiación promedio entregada por la antena en todas las direcciones.
- Ganancia: razón entre la intensidad de potencia radiada por la antena en una dirección (usualmente la dirección de máxima radiación) y la intensidad de radiación que correspondería a una antena isotrópica, dada la potencia suministrada.
- Polarización: la polarización de una antena se refiere a la polarización del campo eléctrico de las ondas electromagnéticas radiadas en una determinada dirección. Esto es, el plano en el que varía el vector de campo eléctrico. La polarización de una antena no tiene por qué ser la misma en todas las direcciones.
- Patrón de radiación: se define como una función matemática o gráfica que represente las propiedades de radiación de una antena como una función

de las coordenadas espaciales. Dentro de las propiedades de interés están la densidad de potencia, la intensidad de radiación, el campo extraño, la directividad y la fase o polarización. Es de especial interés la distribución espacial de la energía radiada en función de la posición del observador.

- Ancho de haz: separación angular de dos puntos opuestos en un lóbulo de un diagrama de radiación de una antena. Por lo tanto existen varios anchos de haz en una sola antena. Por convención el ancho de haz a que se hace referencia es en la dirección de máxima radiación y donde la intensidad de radiación es la mitad del total en esa dirección.
- Ancho de banda: hace referencia al rango de frecuencias en el que una antena se desempeña de manera óptima. Usualmente el rango de frecuencias dado tiene a la frecuencia de resonancia de la antena en el centro.

1.1.5. Radio enlace

En un sistema de comunicación inalámbrica basado en radiofrecuencia, la máxima simplificación posible es la comunicación entre dos únicos nodos. Un radio enlace es la interconexión entre nodos o terminales que cumplen ciertas condiciones tales que pueden establecer una comunicación efectiva entre sí.

La comunicación efectiva a nivel físico implica que las ondas electromagnéticas radiadas por la antena emisora sean captadas por la antena receptora con la suficiente energía y nitidez para que se pueda obtener el mensaje transmitido sobre la onda viajera.

La calidad del radio enlace dependerá de las propiedades físicas, electromagnéticas y funcionales del sistema de comunicación. Así como de factores externos a este, entre los cuales se encuentran la geografía, las edificaciones, la vegetación y las interferencias electromagnéticas.

Un factor numérico que arroja luz sobre la eficiencia de la calidad de un radio enlace es el porcentaje de obstrucción de la primera zona de Fresnel. Que no es más que el espacio comprendido en torno a la línea vista entre las dos antenas en cuestión en donde la reflexión de las ondas electromagnéticas provoca ondas desfazadas y atrasadas con la particularidad de ser constructivas con la señal original.

1.1.6. Red de comunicación inalámbrica

Con los avances en el campo de la transmisión y recepción de ondas electromagnéticas, fue cuestión de tiempo para que las primeras redes de comunicación inalámbrica surgieran de manera comercial. Fue el italiano Guglielmo Marconi quien diseñó el primer sistema telegráfico inalámbrico, lo que le dio el crédito como el inventor de la radio y la coparticipación del premio Nobel por la tal invención.

A inicios del siglo XX, siempre de la mano de Marconi, fueron diseñados los primeros sistemas de comunicación transoceánicos, lo cual fue una completa revolución en la forma en que se llevaban a cabo dichas comunicaciones, que anteriormente eran eminentemente marítimas. Se puede hablar de las primeras redes de comunicación gracias al sistema de inalámbrico que creó Marconi en 1902 y que permitía comunicación bidireccional entre Estados Unidos, Canadá, Reino Unido e Italia.

Aunque podría ser que la red más elemental esté compuesta de dos nodos, el concepto más general es el de muchos nodos interconectados entre sí. El caso de las redes de comunicación inalámbrica no es distinto. Con el paso del tiempo se fue popularizando la tecnología inalámbrica hasta llegar a las cotas de preeminencia que tiene hoy en día. Pese a todo, la idea básica de las redes inalámbricas sigue siendo la misma. Establecer radio enlaces que permitan la transmisión de información desde el emisor al destinatario, es deseado que esto sea de la manera más eficiente posible.

Las redes inalámbricas pueden subdividirse en varias categorías, con base en diversas características de interés, como la frecuencia, el tipo de modulación, la arquitectura, la tecnología, entre otros. Para propósitos de este trabajo se mencionará la clasificación según el tamaño y fin de la red que está enmarcado en la familia de estándares de comunicación inalámbrica del Instituto de Ingenieros Eléctricos y Electrónicos, la IEEE 802.

Tabla 2.

Distintos estándares para redes de comunicación

Estándar	Tipos de red	Descripción
IEEE 802.1	Protocolos LAN de capa alta Working Group	Arquitecturas: LAN (Red de área local) / MAN (Red de área metropolitana) Protocolos: MAC (Control de acceso de medios)
IEEE 802.3	Ethernet	MAC inalámbrico. Aplicaciones LAN y WAN (Red de área amplia)
IEEE 802.11	Wireless LAN (WLAN) / Mesh (certificación Wi-Fi)	MAC inalámbrico. WLAN (Red de área local inalámbrica). Wi-Fi (Fidelidad inalámbrica). Redes Mesh.

Continuación de la Tabla 2.

Estándar	Tipos de red	Descripción
IEEE 802.15	Wireless PAN	Redes inalámbricas de área personal
IEEE 802.15.4	Low-Rate wireless PAN (ZigBee, WirelessHart, MiWi, entre otros.)	Redes de área personal, de baja tasa. Soluciones para sensores a nivel residencial y dispositivos personales.
IEEE 802.15.6	Body area network	Enfocado en dispositivos médicos que se comunican a corto alcance, baja potencia. Para ubicaciones dentro y en las proximidades del cuerpo.

Nota. La tabla muestra distintos estándares de comunicación para redes inalámbricas de parte del Instituto de Ingenieros Eléctricos y Electrónicos. Elaboración propia, realizado con Excel.

1.1.7. Redes malla (*mesh*)

Las topologías básicas para una red de comunicaciones son aplicables al caso de las comunicaciones inalámbricas. Haciendo uso de dispositivos de transmisión y recepción de ondas electromagnéticas es posible emular enlaces de comunicación punto a punto, punto a multipunto y multipunto a multipunto. El uso de ondas de radio propicia una ventaja en la implementación de estos tipos de enlace, ya que el espacio mismo es el medio. A su vez, con combinaciones y configuraciones de estos enlaces básicos, pueden formarse una variedad de topologías.

Un enlace multipunto a multipunto es posible de lograr poniendo a distancias convenientes los distintos nodos de comunicación dentro de un área que permita que el total de radioenlaces entre los dispositivos reciban y transmitan mensajes de manera efectiva.

Las redes de comunicación inalámbrica han sufrido un auge desde hace más de un siglo. La llegada de internet y su gran popularidad permitieron el desarrollo de nuevas tecnologías inalámbricas enfocadas a la interoperabilidad con los protocolos de la suite de internet. WiFi (fidelidad inalámbrica) fue una tecnología que marcó un precedente en la forma de crear redes inalámbricas de área local, WLAN.

La arquitectura básica de WiFi comprende un enrutador con capacidad de establecer enlaces de comunicación inalámbrica con sus dispositivos cliente, a modo de brindarles los servicios esenciales para una red de área local, como, por ejemplo, intercomunicación entre dispositivos de la misma red, servicios de DHCP (protocolo de configuración de host dinámico), enrutamiento entre distintas subredes, entre otros. Más allá de lo anterior, el enrutador tiene la cualidad de ser una puerta de enlace hacia otras subredes (*gateway*), comúnmente es una puerta de enlace hacia la Internet.

El estándar de WiFi pertenece a la familia de estándares del IEEE denominado bajo el identificador, IEEE 802. Otras tecnologías incluidas en los protocolos se enfocan en la creación de distintos tipos de redes inalámbricas, que se pueden enmarcar según su alcance en tallas que van desde dispositivos internos al cuerpo humano hasta áreas metropolitanas y regionales (MAN y RAN).

Es en el ámbito de las redes metropolitanas y regionales que las redes malla (*mesh*), son de utilidad. En el estándar IEEE se detalla el funcionamiento de las redes *mesh*. Al igual que el resto de los estándares de la familia IEEE 802, corresponde a protocolos que operan a nivel de las dos capas más bajas del modelo OSI, la capa física y la capa de enlace de datos. Estas se refieren al

funcionamiento elemental que permite la transmisión de un punto a otro, de manera efectiva y con corrección de errores a nivel de capa física.

Las redes malla no son nuevas y tienen distintas formas de aplicación, para el caso de interés, se presenta una topología particular. Existen tres tipos de dispositivos básicos en una red malla. Los *mesh gateway* (MG), *mesh router* (MR) y *mesh clients* (MC).

A diferencia de, por ejemplo, las redes telefónicas, que tienen estaciones centrales y todos los dispositivos finales deben conectarse directamente con ella, las redes *mesh* no requieren una conexión directa entre los clientes finales (*mesh Clients*) y el concentrador de la información (*mesh gateway*). Esto es posible dado que al ser una red malla, existen múltiples caminos posibles de comunicación entre dos puntos A y B.

Para que lo anterior sea posible, es necesario que tanto los equipos clientes, como los enrutadores tengan la capacidad de operar como host y router al mismo tiempo. Es decir, transmiten su propia información hacia el destino deseado, a la vez que retransmiten la información proporcionada por otros dispositivos que no tienen una conexión directa con su destino y por lo tanto necesitan de dispositivos intermedios para alcanzarlo. Esto último implica que las redes *mesh* funcionan bajo un paradigma de comunicación multisalto.

A diferencia de los otros dos tipos de dispositivos, los *mesh gateways* (MG), tienen la particularidad de ser puertas de enlace para distintas redes. En el caso particular de la red estudiada, estos tienen acceso a una red privada donde se alojan los servidores que procesan los datos generados por los distintos clientes y que permite la creación de una infraestructura de medición avanzada (AMI) funcional, que opera de manera inalámbrica con topología *mesh*.

En resumen, los dispositivos de la red *mesh* tienen la capacidad de interconectarse entre sí, de modo que los clientes y la información medida por ellos, puede ser retransmitida por otros dispositivos (MR y MC), hasta alcanzar el concentrador (MG) que permitirá la integración de los datos en una arquitectura computacional creada para tal propósito.

1.2. Cobertura de una red inalámbrica

Aunque pudiera parecer que este subtítulo corresponde al anterior, la separación se hace debido a que este se refiere al problema de estudio en este trabajo. Cobertura es la cantidad o porcentaje abarcado por una cosa o una actividad. También en la misma fuente se lee: extensión territorial que abarca diversos servicios, especialmente los de telecomunicaciones.

Resulta que, en una red de comunicaciones desplegada en el exterior, la tarea de determinar las áreas de cobertura se vuelve una bastante compleja. Tal complejidad puede atribuirse a la irregular y cambiante geografía, a los fenómenos electromagnéticos que interactúan con un entorno que va más allá de lo que de manera analítica se puede plantear con ecuaciones y procesamiento mental humano, a las limitaciones de la computación digital de nuestro tiempo, a la creciente complejidad de las redes de comunicación inalámbrica, entre otras.

Sin embargo, se hace. Existen diversos enfoques con los que se ha abordado y se aborda todavía hoy este problema complejo, haciendo uso, por supuesto, de ciertas abstracciones que permitan simplificar la tarea y modelar la realidad y su comportamiento hasta cierto punto. Se profundizará más en el capítulo II sobre las bases conceptuales y rendimiento de estos enfoques.

En la literatura al respecto se utilizan diversas etiquetas para categorizar los enfoques utilizados en la solución del problema de calcular o estimar la cobertura de una red de comunicaciones inalámbrica. Para propósitos de este trabajo se segmentarán los enfoques de la siguiente manera: enfoque determinístico y no determinístico (o estocástico). Es posible asociar la palabra calcular con el determinístico y estimar con el estocástico. Para asentar los conceptos básicos que permitan futuras discusiones se desarrollan los siguientes temas.

1.2.1. Hipótesis

Una hipótesis es una idea o planteamiento que se propone como una forma de dar explicación a un fenómeno del mundo real. Enmarcando el mundo real como aquello que podemos acceder con nuestros sentidos y lo que podemos medir haciendo uso de instrumentos. Usualmente la hipótesis es acompañada de una investigación que permita aceptar o refutar la idea inicial.

1.2.2. Teoría

Es una abstracción racional que pretende dar explicación a algún fenómeno del mundo real, que ha sido validada a través de experimentación y que cuenta con cotas aceptables de concordancia con las mediciones realizadas. Aunque la teorización tiene una considerable carga racional e imaginativa, cuenta con una importante proporción de observaciones del fenómeno, como es lógico. Incluso las teorías más abstractas que se proponen dar explicación a fenómenos naturales, se consolidan con las observaciones del mundo real.

1.2.3. Modelo

Es una simplificación de la realidad que pretende explicarla a fin de que sea más sencillo poder describirla y realizar predicciones bajo la modificación de distintas condiciones de entrada. Existen modelos físicos que son una representación a escala de lo que se desea modelar. También existen modelos abstractos que están fundamentados en ideas y cuya fuente de exteriorización es el lenguaje humano y matemático. Para un modelo se asumen ciertas limitaciones que permiten la simplificación de algo que en la realidad es más complejo.

1.2.4. Modelo matemático

Es el que utiliza el lenguaje y conceptos matemáticos para describir el fenómeno de interés. Usualmente se establecen relaciones numéricas entre distintas magnitudes, de modo que se establezcan ecuaciones que describan la realidad de forma matemática.

El modelaje matemático es todo un campo de estudio y tiene una amplia variedad de categorías. Los modelos no determinísticos y los algoritmos serán relevantes en el presente trabajo.

1.2.5. Modelo probabilístico

Son modelos del mundo real basados en la teoría de la probabilidad y que tienen la capacidad de lidiar con la incertidumbre presente en todas las mediciones realizadas por los seres humanos. Ya sea por la limitada resolución de los equipos como por la incapacidad de realizar una medición completa del fenómeno en cuestión, se recurre a la probabilidad como una forma de

representar la realidad indicando qué tan probable es que ocurra alguno de los eventos del espacio muestral (todos los resultados posibles).

1.2.6. Área de cobertura

Para una red de comunicación inalámbrica el área de cobertura es la superficie de la región de interés donde se tiene un nivel mínimo que permite establecer una comunicación efectiva. Surge entonces la necesidad de indicar qué significa una comunicación efectiva. Esta condición se puede medir estableciendo umbrales de calidad para determinada variable, de modo que si este se supera hay comunicación efectiva.

Para propósitos de este trabajo, se discutirá más adelante la estrategia a utilizar, tomando en cuenta que para el enfoque propuesto no se pretende una asignación arbitraria o meditada de los umbrales, sino que se esperará que los algoritmos implementados infieran las condiciones que conllevan a una comunicación efectiva. De modo que pueda extenderse a determinar áreas con cobertura haciendo uso de sus predicciones.

1.2.7. Enfoques para el cálculo de cobertura

Este es un tema que se discutirá con mayor profundidad al comentar la literatura, ya que se podrá identificar casos prácticos que se enmarcarán en uno o más de los enfoques mencionados a continuación. Solo cabe resaltar que la determinación de la cobertura es una tarea fundamental tanto en la implementación como en la gestión de una red de comunicación inalámbrica y que el esquema físico y funcional de una red malla (*mesh*) agrega un mayor grado de complejidad, puesto que cada nuevo dispositivo de la red *mesh* modificará la

cobertura en su vecindad, incluso si este corresponde a un dispositivo final (*mesh client*).

1.2.7.1. Enfoque determinístico

Este enfoque está basado en la teoría electromagnética y la teoría de antenas. Haciendo uso de los modelos matemáticos existentes en la teoría se utilizan los datos particulares de la red inalámbrica de interés y se calcula una variable de interés, por ejemplo, los niveles de potencia de la señal en determinado punto considerando las implicaciones de la teoría sobre la atenuación de la señal, la distorsión y su comportamiento en el medio ambiente. Este resultado permite calcular la cobertura en un punto específico.

Para casos de análisis teórico este método es bastante certero, pero, como se ha dicho, todos los modelos implican ciertas limitaciones iniciales. En el caso del modelo teórico esto no es una excepción, plantearse un análisis de la interacción de las ondas electromagnéticas con un entorno irregular y en muchos casos repleto de vegetación y edificaciones complica la tarea de manera considerable.

Por tal motivo, las limitaciones restringen la cantidad de variables del entorno que se ingresan al modelo. La geografía continúa siendo de interés especial, pero se consideran sólo algunos elementos de esta. La inclusión de variables categóricas como el nivel de vegetación o la presencia de edificaciones es una tarea complicada.

Pese a lo anterior, los modelos determinísticos han sido refinados para arrojar predicciones que puedan coincidir con la realidad de una red de comunicación en el exterior. En última instancia, pese a que el modelo es

determinístico no se puede prescindir de los datos en campo para nutrir y corroborar el mismo.

1.2.7.2. Enfoque empírico

En contraposición a una visión determinista del problema, los modelos empíricos permiten lidiar con la incertezza inherente a la naturaleza y llegar a conclusiones que no son definitivas sino probables. Es necesario entonces que se tenga en cuenta que una limitación inicial del modelo es su precisión.

Para este tipo de enfoque hay distintas estrategias a utilizar, una de ellas puede ser la utilización del conocimiento a priori brindado por la teoría, de modo que se pueda traducir la información teórica a un modelo probabilístico, un ejemplo de esto podría ser un modelo donde la probabilidad de cobertura decrezca en función de la distancia, coincidiendo con lo que dice la teoría.

De igual forma existen estrategias que no se basan en las relaciones numéricas teóricas, sino que son análisis estadísticos los que permiten arrojar una estimación del nivel de cobertura en determinado punto. En este tipo de modelos comienza a ser más versátil la cantidad y tipo de datos que se aceptan en la entrada. Podría decirse que los algoritmos de inteligencia artificial a la fecha son empíricos, sin embargo, su auge ha sido tan disruptivo que se catalogará como un enfoque aparte.

1.2.7.3. Enfoque por inteligencia artificial

Como su nombre lo indica, este enfoque pretende hacer uso de los avances en el ámbito de la inteligencia artificial para dar solución al problema de

determinar la cobertura en una red de comunicación inalámbrica. En la revisión de la literatura se citarán ejemplos puntuales de los avances en este respecto.

Grosso modo podemos decir que dar solución al problema de cobertura haciendo uso de inteligencia artificial consiste en crear un sistema computacional capaz de procesar ciertos datos provenientes de una red desplegada en campo para identificar cierta información subyacente a los datos que permita, por medio de distintas etapas de entrenamiento y sin haber sido programado para ello de manera explícita, predecir la cobertura de manera eficiente y congruente con los datos proporcionados.

1.3. Inteligencia artificial

Este es el otro eje central de este trabajo. Una aplicación de esta a un problema específico. Desambiguar las palabras es una tarea difícil, puesto que se vuelve necesario una definición intermedia, la de inteligencia. Una definición para la inteligencia dada por nuestra inteligencia. Es un tipo de paradoja similar al lenguaje que se precisaría para tal definición. De momento se dirá que es la capacidad que tiene un ente de asimilar algún grado existencia e interactuar con ella en pro de la culminación de un objetivo. Palabras como decisión, análisis, conocimiento y aprendizaje se utilizarán de aquí en adelante refiriéndose a la inteligencia artificial, debe ser evidente que a pesar de las similitudes con sus correspondientes biológicas, existen diferencias sustanciales. Mismas que no implican que la inteligencia biológica, como la que escribe estas líneas, no pueda quedar rezagada contra su versión artificial en cuanto a todo lo que se enmarca en el término inteligencia.

En cuanto a la inteligencia artificial no hay mucho por decir, la definición de diccionario producido por *el ingenio humano* es de utilidad. De entrada, todas

las inteligencias no biológicas que se conocen han sido creadas por humanos, e incluso las IA que han sido creadas por otras IA, tienen en algún eslabón de la cadena la participación de un homo sapiens.

El hecho de que el marco existente de la Inteligencia Artificial sea una base teórica para la realización de este análisis de cobertura es resultado de que, de hace unas décadas hasta hoy, ha sido un campo de interés en sectores académicos cada vez más diversos y amplios. Su línea del tiempo se discutirá en el capítulo próximo, pero cabe resaltar que la correspondencia con el desarrollo de las tecnologías digitales y la creciente cantidad de datos de esta era han sido el motor y combustible para un avance vertiginoso que aún continúa generando nuevos hitos que cada vez son menos anormales.

1.3.1. Aprendizaje de máquina

Más conocido por su nombre en inglés, *machine learning*. Es una de las tareas asociadas con la inteligencia y se refiere a la adquisición de algún tipo de conocimiento basado en experiencias pasadas. En el caso de los algoritmos que se utilizarán en máquinas de cómputo, esta experiencia serán los datos.

Existen tres enfoques principales para abordar las tareas de *machine learning*, y estos son: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje con refuerzo. Dado que se cuenta con datos etiquetados, el primero será el principal paradigma explotado en este trabajo.

1.3.1.1. Aprendizaje supervisado

Se refiere a los métodos de aprendizaje de máquina que utilizan datos etiquetados para la solución de la tarea en cuestión. Esto es, para cada entrada

en el conjunto de datos de entrenamiento, existe un valor correspondiente denominado como etiqueta, y que indica que, para determinado conjunto de datos adquiridos del problema de estudio, su correspondiente solución es la etiqueta.

Los algoritmos de aprendizaje supervisado están diseñados para ajustar sus parámetros a fin de generar predicciones que permitan minimizar el error entre el valor estimado y la etiqueta. Esto durante el entrenamiento se valida con datos del conjunto de entrenamiento, luego de eso es recomendable utilizar un conjunto de datos de prueba para comprobar si se ha generalizado el conocimiento.

1.3.1.2. Aprendizaje no supervisado

Es el enfoque en el cual el sistema computacional aprende de los datos sin que exista un valor correspondiente de etiqueta para los elementos en el conjunto de datos de entrenamiento. De ahí el nombre, puesto que no existe una supervisión que permita en todo momento saber si los resultados estimados son correctos o no.

Por tal motivo, los algoritmos de aprendizaje no supervisado buscan las similitudes subyacentes de los datos para clasificarlos. Las tareas de categorización son uno de los principales campos de aplicación de estos. Para la implementación de este paradigma se han desarrollado diferentes algoritmos, de modo que existen métodos matemáticos depurados y que arrojan distintas métricas para evaluar el desempeño del modelo.

1.3.1.3. Aprendizaje con refuerzo

Este tipo de aprendizaje de máquina está enfocado a la solución de problemas donde un agente inteligente interactúa con un ambiente con la finalidad de maximizar la recompensa obtenida al interactuar con el ambiente de una manera óptima. Se denomina con refuerzo dado que existen incentivos y penalizaciones correspondientes a las acciones del agente en un estado particular.

Los algoritmos de aprendizaje por refuerzo optimizan sus políticas de actuación a medida que exploran el ambiente. Para tal propósito se usan representaciones matemáticas que permiten identificar la política más eficaz dependiendo de un estado particular en el que se encuentre el agente dentro del ambiente. Estos algoritmos son aplicados a la solución de juegos y es en este campo donde han obtenido sus resultados más sonados.

1.3.2. Aprendizaje profundo

Conocido como *deep learning*, es la rama del *machine learning* que se enfoca en la utilización de redes neuronales artificiales con aprendizaje de características para la solución de problemas de aprendizaje de máquina bajo los paradigmas mencionados anteriormente: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje con refuerzo.

El aporte propiciado por los algoritmos de *deep learning* ha significado una mejora en la solución de problemas para los que se tenían soluciones de *machine learning*, así como lograr solución a problemas que anteriormente no se habían solucionado de una manera satisfactoria. La parte profunda del nombre se refiere a la utilización de redes neuronales profundas, que son llamadas así por las

capas profundas de procesamiento de los datos antes de obtener el resultado de salida.

1.3.3. Red neuronal artificial

Son modelos de inteligencia artificial que basan su funcionamiento, de manera somera, en el funcionamiento de un cerebro biológico. Esto lo hacen a través de una red interconectada de varias capas de neuronas artificiales. Estas neuronas son funciones matemáticas que realizan una operación lineal entre los valores numéricos de sus entradas para posteriormente aplicar una función no lineal de activación.

Los mecanismos con los cuales se lleva a cabo el entrenamiento de una red neuronal artificial son: *forward propagation* y *back propagation*. El primero corresponde a una propagación de la entrada hacia las capas siguientes de neuronas hasta alcanzar la capa de salida, esto en primera instancia arroja una estimación de la red neuronal para la entrada particular. El segundo corresponde a la optimización de los parámetros de la red haciendo uso de una función de pérdida que indique el desempeño del algoritmo a la vez que se convierte en la función objetivo de la optimización, misma que se hace utilizando un algoritmo iterativo de optimización, pero esta vez desde la capa de salida hasta la primera capa de neuronas.

1.3.4. Ciencia de datos

No puede decirse que la inteligencia artificial esté enmarcada dentro de la ciencia de datos, ni al revés. Más bien hay regiones de intersección que permiten potenciar ambas direcciones de desarrollo científico y tecnológico. La ciencia de datos por su parte se encarga de la utilización de distintas herramientas

estadísticas, matemáticas y computacionales para extraer información útil de un conjunto de datos provenientes del mundo real y que por lo tanto contiene ruido.

En el transcurso del tiempo la ciencia de datos se ha consolidado como tal. Y es que, a pesar de la parte eminentemente empírica del trabajo con datos, cada vez es más usual que en entornos de investigación de alto nivel científico y donde se estudian fenómenos de la naturaleza, los datos sean una pieza clave. Como se ha mencionado antes, la gran mayoría de los procesos que involucran la creación de modelos del mundo real, tienen una importante componente empírica.

Más allá de lo anterior, la ciencia de datos está irrumpiendo en todo tipo de ámbitos, desde los empresariales hasta los centros de investigación. La ciencia de datos es tanto una ciencia teórica como aplicada. Y es en la aplicación que busca dar solución a un problema que resulta pertinente este apartado.

Como una forma de sistematizar el proceso que se desarrollará para la implementación de este trabajo se presentan a continuación las distintas etapas que generalmente se asocian al flujo de trabajo en la ciencia de datos aplicada a la solución de un problema real.

1.3.4.1. Entendimiento del negocio

Como es bien sabido, la inteligencia artificial cuenta aún con varias limitantes y una de ellas es que requiere de la guía de un ser humano para conseguir los objetivos. Esto no se refiere a una guía al momento del entrenamiento, sino a la creación de un marco de trabajo congruente y con objetivos claros.

Es necesario que los objetivos que se pretendan alcanzar con el diseño e implementación de las etapas siguientes sean realistas, alcanzables y útiles. Abordar el problema desde el punto de vista solo de la ciencia de datos puede acarrear a la pérdida de información útil que proviene de las personas que han tenido contacto de primera mano con el problema de estudio.

Por tal motivo es útil analizar el problema desde el punto de vista del negocio (ámbito del problema), para identificar posibles variables significativas, patrones conocidos, posibles fuentes de datos y necesidades específicas. Resultado de este entendimiento se puede vislumbrar el tipo de análisis requerido, ya sea este descriptivo, de diagnóstico, predictivo o prescriptivo. De igual forma, una vez decidido el análisis requerido se puede plantear específicamente la pregunta que se quiere responder.

1.3.4.2. Minería de datos

Con la guía proporcionada por los objetivos de la implementación y el conocimiento de las distintas fuentes de datos, se vuelve necesario la extracción de los datos de tales fuentes. En general es necesario la utilización de distintas herramientas computacionales para abordar los distintos escenarios que se presentan.

1.3.4.3. Limpieza de datos

Los datos obtenidos de diversas fuentes la mayoría del tiempo se disponen de manera heterogénea y no estructurada. Es importante para la creación de modelos satisfactorios que los datos proporcionados sean de calidad.

Esta etapa implica la eliminación de valores duplicados, la imputación a los datos faltantes, la homogenización de los datos a un formato de utilidad, el tratamiento de los valores atípicos y la aplicación de la ingeniería de características.

Se dice en diversas fuentes que la anterior etapa y está comprenden la mayor parte del tiempo dedicado a los desarrollos de soluciones de ciencia de datos. En capítulos posteriores se comentará más sobre esto y se discutirá el caso particular de esta implementación.

1.3.4.4. Exploración

Una vez se tiene el conjunto de datos que se utilizará para el desarrollo de los modelos pertinentes es necesario tener una idea general de la distribución de los datos, las posibles relaciones entre distintas variables, así como con la variable objetivo (en el caso del aprendizaje supervisado).

En lo subsiguiente se utilizará la palabra característica para referirse a las distintas variables que componen el conjunto de datos. En esta etapa se investigan las características a fin de asimilar parte de la información contenida en ellas, la mayor parte la asimilará el modelo de inteligencia artificial.

1.3.4.5. Modelaje

Corresponde, como el nombre lo indica, al desarrollo de modelos que de una forma simplificada representarán el fenómeno de estudio. En el caso específico de este trabajo, es este apartado donde se hará uso de la inteligencia artificial, del *machine learning*. Lo que puede expresarse como, haciendo uso de algoritmos de inteligencia artificial, se procesaran los datos disponibles para

obtener modelos. Teniendo como guía los objetivos identificados en la primera etapa del proceso de aplicación de la ciencia de datos.

Para conseguir esto será necesario elegir los distintos algoritmos, mismos que contendrán una serie de parámetros a optimizar. Haciendo uso de varias etapas de entrenamiento se ajustarán los parámetros de los modelos para que su resultado sea el mejor posible, teniendo como base, métricas, que se serán acordes al tipo de modelo que se esté implementando.

Tanto el aprendizaje de máquina como el aprendizaje profundo serán investigados como opciones viables para resolver el problema.

1.3.4.6. Interpretación de resultados

Luego de tener distintos modelos como posibles soluciones al problema, será necesario interpretar los resultados arrojados por los distintos modelos y compararlos de manera que pueda decidirse cuál es el que mejor cumple con el objetivo, o si ninguno lo hace.

Una guía para la interpretación de los resultados será la hipótesis de este trabajo, de modo que esta pueda aceptarse o refutarse. También se comentarán las posibles ventajas y desventajas de la utilización de los modelos.

1.4. Ingeniería de características

Como se mencionó antes, las características son lo mismo que las variables que se utilizan para implementar un modelado basado en datos. Puede decirse que las características son los valores particulares que permiten distinguir

entre sí las distintas entradas en el conjunto de datos que se dispone. Son los atributos que representan al fenómeno del mundo real que se está estudiando.

Una condición importante de las características que se ingresan a los algoritmos de aprendizaje de máquina es que cada variable distinta corresponderá a una distinta dimensión en el hiperespacio en el que será entrenado el algoritmo de inteligencia artificial. Es común que las dimensiones sean mayores que tres, lo que sobrepasa la capacidad de imaginación humana.

Estos hiperespacios pueden en algunas ocasiones carecer de las dimensiones mínimas o saturar el algoritmo con dimensiones que están demás. También es posible que para distintos algoritmos el formato o escala requeridos de los datos sean desiguales.

Puede decirse que, para conseguir resultados satisfactorios al hacer uso de un algoritmo de inteligencia artificial, es de suma importancia que los datos que ingresen al modelo sean de calidad y cumplan con cierto formato que propiciará un mejor desempeño de los algoritmos. Utilizar los datos tal cual fueron extraídos de las fuentes de información puede que condicione a la etapa de modelaje del proceso de ciencia de datos a fracasar.

Es aquí donde surge la necesidad de un trabajo previo de ingeniería que condicione los datos según sean los algoritmos que se pretenden usar. Este trabajo es la ingeniería de características, y como es lógico, varía en función de los datos que se disponen y los algoritmos que los consumirán.

Dentro de las diversas labores que involucra la ingeniería de características, las siguientes son las que atañen en este trabajo.

1.4.1. Transformación de características

Consiste en utilizar las características existentes para generar nuevas características que puedan ser de utilidad para el desempeño del modelo. La información recabada al analizar el problema desde el punto de vista de su ámbito específico puede arrojar luz sobre qué características pueden enriquecer al conjunto de datos original.

Para este propósito es usual realizar mapeos matemáticos que permitan introducir algún concepto teórico o alguna relación de interés que se quiere indicar explícitamente al algoritmo de inteligencia artificial y que no es una función trivial que pueda estar contenida dentro de las mismas transformaciones realizadas por el algoritmo.

1.4.2. Generación de características

Corresponde a la extracción de nuevas características para agregar al conjunto de datos original que no corresponden a una transformación de las variables existentes. De nuevo, la experiencia adquirida del análisis del problema desde su ámbito puede sugerir distintas opciones para la generación de las nuevas características.

1.4.3. Selección de características

Cuando se dispone de una gran cantidad de características puede parecer el mejor escenario y es posible que se plantee la utilización de todas las características disponibles para el entrenamiento de los respectivos modelos. Esto pudiera no ser lo más aconsejable por diversos motivos.

En cuanto al coste computacional, la utilización de una mayor cantidad de características aumentará la complejidad del *software* que de la IA, lo que puede repercutir en un mayor tiempo de entrenamiento. Por otro lado, independientemente del coste de cómputo, es posible que existan características dentro del conjunto de datos que no sean de utilidad para el propósito específico del algoritmo que se esté implementando, e incluso es posible que existan características que reduzcan el desempeño del algoritmo.

Por tal motivo resulta necesaria la selección del conjunto con el menor número de características posibles y con el mayor desempeño posible para el modelo. Esta selección, al igual que la mayoría de tareas enmarcadas dentro de la ingeniería de características, puede basarse en criterios humanos como en un proceso automatizado de selección de las mismas.

1.4.4. Aprendizaje de características

Se refiere a la utilización de métodos computacionales para la generación de una gran cantidad de características para luego proceder a una selección automática teniendo como objetivo el mejoramiento de alguna métrica acorde al algoritmo utilizado para tal propósito.

Más adelante se verá cómo es que una red neuronal está íntimamente ligada con la generación y selección automática de características, y cómo esta relación ha permitido potenciar el aprendizaje profundo. Adicional a las redes neuronales existen diversos métodos para cumplir este propósito. A pesar de la inclusión de métodos automatizados para este, en la ingeniería de características continúa siendo necesaria una dosis de análisis humano para seleccionar los métodos de aprendizaje de características acordes a cada situación.

1.5. Infraestructura computacional para entrenar IA

Para alojar a la IA, o, mejor dicho, para alojar el modelo de IA, se requiere por supuesto, de algún tipo de almacenamiento digital, donde el *software* desarrollado pueda cargarse, y algún tipo de procesamiento digital para que la ejecución del modelo pueda realizarse.

El modelo per se, puede ser desde una simple función matemática hasta una complicada e interconectada función matemática, como los modelos llamados red neuronal. Este es el resultado final y como tal es conciso en el cumplimiento de la tarea particular para la que fueron entrenados los parámetros del modelo.

En comparación con la cantidad de datos que usualmente se utilizan para el entrenamiento de los modelos de IA, el espacio de almacenamiento necesario para el modelo entrenado es menor, aunque en la actualidad existen modelos entrenados que requieren de centros de datos enteros para su ejecución.

La etapa previa de entrenamiento es, por lo tanto, la que requiere una mayor optimización de recursos, dado que es la que más demanda. Para tal propósito se esbozará de forma general la arquitectura que se desea desarrollar en la implementación de la idea de este trabajo. Tomando en cuenta que se pretende mejorar en primera instancia el tiempo de ejecución y luego el almacenamiento en el sistema computacional utilizado.

Y, dado que se habla de computadoras, también será necesario un lenguaje de programación que permita interactuar con las máquinas para indicar las funciones a realizar, al menos hasta que inicie el proceso de entrenamiento,

donde estas dependerán de la inteligencia artificial, o en última instancia, de los datos.

1.5.1. Mercado de datos

En el ámbito de la ciencia de datos, y principalmente en los ambientes empresariales, surgen conceptos referidos a la forma de almacenar datos para su disposición efectiva a la hora de ser utilizados para extraer información de ellos. Desde visualizaciones, resúmenes, hasta predicciones basándose en inteligencia artificial.

Dentro de estos conceptos se enumeran los siguientes, los cuales son comúnmente referidos por sus nombres en inglés:

- *Data lake* (lago de datos): es una forma de almacenar datos donde se conservan en su formato original, y que se pretende sea explotada posteriormente.
- *Data warehouse* (almacén de datos): la información se almacena en un formato que ha sido estandarizado y posterior a la aplicación del proceso de ETL (extracción, transformación y carga por sus siglas en inglés). Se refiere a una base de datos que contiene información de los distintos departamentos, si se refiere al ámbito empresarial.
- *Data Mart* (mercado de datos): están contenidos dentro de un almacén de datos y su propósito es específico para el uso que ha de recibir de los usuarios finales.

Teniendo en cuenta lo anterior, el tipo de almacenamiento a utilizar en este trabajo será un mercado de datos. Mismo que tendrá ciertas particularidades, dado que estará contenido en un *data warehouse* con uno solo *data mart*, y que será desarrollado en un ambiente particular.

En resumen, el *data mart* coincidirá con el conjunto de datos que será utilizado para el entrenamiento de los modelos de IA. O, los conjuntos de datos, dado la posible necesidad de variaciones de los datos para distintos modelos. El proceso ETL, se realizará para que la información disponible tenga la mejor calidad posible. Luego de eso, se considerará la información almacenada en el *data mart*, como la única fuente de la verdad en el contexto de la implementación de la solución.

1.5.2. Python

La finalidad de los lenguajes es la de transmitir información, teniendo como condiciones un contexto compartido de modo que pueda comprenderse la información contenida en las distintas representaciones del lenguaje (símbolos, sonidos, estímulos, nivel de voltaje en un semiconductor). El ejercicio del lenguaje presupone un pasado común a los interlocutores, como indica Borges (1964). Tanto en la naturaleza como en los ambientes artificiales, surgen los lenguajes.

Para comunicarnos con una máquina, también son necesarios, y permiten a distintos niveles y de distintos modos transmitir instrucciones o datos a las computadoras. Los lenguajes formales son los que se han creado con reglas bien definidas y haciendo uso de la lógica y de la matemática, los lenguajes de programación están contenidos dentro de los lenguajes formales y son específicos para interactuar con máquinas de cómputo. El lenguaje natural será

en el contexto de los humanos, la forma en que nos comunicamos haciendo uso de los signos lingüísticos pertenecientes a los distintos idiomas.

Antes solo se podía utilizar el lenguaje formal para la comunicación con las computadoras, y el lenguaje natural solo en algunas aplicaciones específicas creadas con lenguajes de programación formales. A la fecha, el uso de la inteligencia artificial en la aplicación del procesamiento del lenguaje natural está incluyendo al lenguaje humano como una forma de interactuar con las máquinas, o, mejor dicho, permitiendo a las máquinas asimilar este lenguaje.

En el caso específico de esta implementación, se utilizará en lenguaje de programación formal Python, como medio para programar los procesos de extracción de datos, transformación de datos, ingeniería de características, entrenamiento de modelos y presentación de resultados. En caso sea requerido otro lenguaje, será por cuestiones específicas, pero el grueso del trabajo será con Python.

Python es un lenguaje cuya sintaxis es sencilla y legible a simple inspección. Lo que está incrustado en el seno mismo del lenguaje y en la comunidad de personas que lo utiliza. Esto se refleja de manera eminentemente en el Zen de Python, mismo que puede obtenerse del interprete al ejecutar la sentencia: *import this*.

Figura 1.

The Zen of Python

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Nota. Se presenta una serie de principios que rigen la filosofía del lenguaje de programación Python. Elaboración propia, realizado con consola de Python.

Por su filosofía y practicidad de uso Python se ha convertido en uno de los lenguajes de programación más populares del mundo. Hoy son muchos los ámbitos en los que se utiliza Python y es considerado un lenguaje de alto nivel y de propósito general, dado que se ha adaptado para una amplia variedad de tareas, desde la programación de microcontroladores hasta *clusters* de servidores. Y las entidades que lo aplican van desde las empresas de alta tecnología, instituciones de gobierno, centros de investigación, hasta estudiantes universitarios en su trabajo de graduación.

No cabe mucho por decir al respecto, más que la manifiesta intención de apegarse a las ideas de simplicidad del lenguaje en cuanto sea posible, y teniendo en cuenta que en determinadas circunstancias la practicidad prevalecerá. Aprovechando la amplia comunidad detrás de Python se recurrirá a varias librerías creadas por esta misma, para proceder según sea la necesidad.

1.5.3. Recursos para ingeniería de características

En este apartado específico se referirá de manera general los posibles usos de tecnologías que faciliten la tarea de la ingeniería de características. Que es, el tratamiento de las dimensiones existentes en el conjunto de datos para optimizar los algoritmos de aprendizaje de máquina.

Python será una de las herramientas primordiales para tal propósito, de modo que haciendo uso de sus librerías de manejo de datos se pueda aplicar la ingeniería de características.

Como se tienen ciertas nociones previas de las posibles características a generar, se vuelven importantes las herramientas de información geográfica. Para tal propósito se recurrirá a APIs proporcionadas por empresas que brindan este tipo de información bajo requerimiento.

Una API, del inglés (*application programming interface*), es un elemento de *software* que permite la comunicación entre dos computadoras. Existe una computadora que expone una API en alguno de sus puertos y que es interrogada por otra con la finalidad de requerir información basándose en ciertos parámetros de entrada. La información puede ser el resultado de una consulta a alguna base de datos o el resultado de aplicar una serie de instrucciones de código a los parámetros de entrada.

1.5.4. Computador de placa reducida

Para el desarrollo de este trabajo se requerirá, por supuesto, la utilización de un ordenador. Para tal propósito se contará con una computadora personal con sistema operativo Windows, que contendrá el ambiente de Python y acceso

a las distintas fuentes de datos, así como posibles servicios externos a consultar, será también aquí donde se almacene el conjunto de datos y los modelos. En caso de ser necesario por motivos de tiempo de ejecución, se recurrirá a herramientas en la nube que permitan entrenar los modelos con una mayor velocidad.

El computador de placa reducida es un tipo de computadora funcional que tiene la particularidad de estar contenida en una sola placa, cuyas dimensiones son relativamente pequeñas. Estas computadoras son menos costosas que una personal y su consumo de recursos también es menor. Son utilizadas para aplicaciones embebidas. En el caso particular de este trabajo su utilización principal será la de simular un servidor con sistema operativo Linux que permita un funcionamiento continuo y dedicado. Es posible que debido a su arquitectura exista incompatibilidad con algunas librerías utilizadas, para lo que la solución será implementar el *software* en la computadora personal disponible.

2. REVISIÓN DE LA LITERATURA

El concepto de literatura hace referencia a la palabra, al lenguaje, a los compendios de registros del lenguaje que tratan acerca de un determinado tema, de imaginaciones, tradiciones, ficciones, estudios, tratados y más. La historia de la humanidad no como relato del pasado sino como registro temporal de distintos presentes. La evolución de las ideas y de las distintas formas de relatar la misma historia. El concepto de Inteligencia Artificial surge de una evolución del lenguaje, que llega hasta nuestros días como una palabra en auge, aunque tiene ya algunas décadas de existir y describe una idea que tiene acaso milenios. En la literatura que se citará ulteriormente parecerá que se exageran al límite tanto los niveles más rudimentarios de la IA como los más fantásticos e inverosímiles. Sucede que no somos capaces de definir la inteligencia, intentarlo solo resulta ser un tipo de autorreferencia. De modo que establecer límites para la misma resulta un acto vano, como también lo es el fantaseo con esta u otras ideas, como también acaso lo es todo. Menos mal que estamos en un trabajo de graduación y que existe una vasta literatura referente al tema particular de interés. Será la intención de esta revisión la de mencionar algunos hitos conocidos al respecto de lo que es Inteligencia Artificial desde una perspectiva general a la específica de los conceptos particulares utilizados en este trabajo y que se encontraron en la nunca suficiente literatura consultada.

2.1. Literatura no técnica sobre IA

Decir, sí, como el griego afirma en el Cratilo, no es lo mismo que escribir *Si como el griego afirma en el Cratilo*. Tampoco lo es escucharlo o leerlo. Fue por la vía oral que primero me acerqué al poema de Borges, y su impacto fue

significativo al escuchar las líneas que seguían luego. Una rotunda afirmación me parece más potente que una sensata condicionante, y es que en el mismo texto griego referido no se llega a determinar cuál ha de ser la naturaleza del lenguaje, ni si las palabras equivalen a las cosas o por el contrario son convenciones humanas arbitrarias, Borges, por supuesto, tampoco daría ese paso en falso. La vía visual también trajo sus propias ventajas, entre ellas una mayor posibilidad de análisis del poema, por lo que no me decanto por la una o por la otra, pero al ser este un trabajo escrito, ofrezco la segunda.

Si —como el griego afirma en el Cratilo—

el nombre es arquetipo de la cosa,

en las letras de rosa está la rosa

y todo el Nilo en la palabra Nilo.

Y, hecho de consonantes y vocales,

habrá un terrible Nombre, que la esencia

cifre de Dios y que la Omnipotencia

guarde en letras y sílabas cabales.

Adán y las estrellas lo supieron

en el jardín. La herrumbre del pecado

—dicen los cabalistas— lo ha borrado

y las generaciones lo perdieron.

Los artificios y el candor del hombre

no tienen fin. Sabemos que hubo un día

en que el pueblo de Dios buscaba el Nombre

en las vigilias de la judería.

No a la manera de otras que una vaga
sombra insinúan en la vaga historia,
aún está verde y viva la memoria
de Judá León, que era rabino en Praga.

Sediento de saber lo que Dios sabe,
Judá León se dio a permutaciones
de letras y a complejas variaciones
y al fin pronunció el Nombre que es la Clave,
la Puerta, el Eco, el Huésped y el Palacio,
sobre un muñeco que con torpes manos
labró, para enseñarle los arcanos
de las Letras, del Tiempo y del Espacio.

El simulacro alzó los soñolientos
párpados y vio formas y colores
que no entendió, perdidos en rumores
y ensayó temerosos movimientos.

Gradualmente ser vio —como nosotros—
aprisionado en esta red sonora
de Antes, Después, Ayer, Mientras, Ahora,
Derecha, Izquierda, Yo Tú, Aquellos, Otros.

(El cabalista que ofició de numen

a la vasta criatura apodó Golem;
estas verdades las refiere Scholem
en un docto lugar de su volumen).

El rabí le explicaba el universo
Esto es mi pie; esto el tuyo; esto la soga
y logró, al cabo de años, que el perverso
barriera bien o mal la sinagoga.

Tal vez hubo un error en la grafía
o en la articulación del Sacro Nombre;
a pesar de tan alta hechicería,
no aprendió a hablar el aprendiz de hombre.

Sus ojos, menos de hombre que de perro
y harto menos de perro que de cosa,
seguían al rabí por la dudosa
penumbra de las piezas del encierro.

Algo anormal y tosco hubo en el Golem,
ya que a su paso el gato del rabino
se escondía. (Ese gato no está en Scholem
pero, a través del tiempo, lo adivino).

Elevando a su Dios manos filiales,
las devociones de su Dios copiaba
o, estúpido y sonriente, se ahuecaba

en cóncavas zalemas orientales.

El rabí lo miraba con ternura
y con algún horror. ¿Cómo —se dijo—
pude engendrar este penoso hijo
y la inacción dejé, que es la cordura?
¿Por qué di en agregar a la infinita
serie un símbolo más? ¿Por qué a la vana
madeja que en lo eterno se devana,
di otra causa, otro efecto y otra culta?
En la hora de angustia y de luz vaga,
en su Golem los ojos detenía.
¿Quién nos dirá las cosas que sentía
Dios, al mirar a su rabino en Praga? (Borges, 1964 pp. 519-521)

El Golem es una figura de Adán, a la vez que, de los primeros humanos creados en el Popol Vuh, la condición de seres artificiales nos la hemos impuesto a nosotros mismos en la mayoría de las culturas de una forma explícita, aunque no faltarán los casos que difuminen la idea central de haber sido creados por algún otro tipo de inteligencia. El paradigma científico, como era de esperarse, no arroja luz sobre la cuestión, por un lado, la teoría de la evolución remueve esa necesidad de un diseñador que convenientemente puso su mayor dedicación en el género humano. La física plantea la posibilidad de un inicio del universo en el cual parece no haber un tiempo donde sucediese el acto creador. Por otro lado, pivotamos nuestras explicaciones centrales del mundo desde la perspectiva de un observador, aunque tal parece existen condiciones de la realidad difuminadas

hasta que entra en juego el ojo que observe. La condición individual parece ser artificial o cuanto menos un producto de condiciones particulares y ambientales, culturales, artificiales. Se plantea incluso que la percepción que vivimos como realidad sea producto de una simulación, lo que devuelve la cuestión a la pregunta de si hay un creador que ponga en marcha la increíblemente realista simulación en que vivimos. Discurrir en palabras es cuestión de nunca acabar, y parece que Borges en su poema expresó casi la totalidad de las ideas que me interesa plantear. Así que cualquiera que quiera saltarse las siguientes páginas, este es el punto de proceder.

2.1.1. El lenguaje

Si, como el otro griego, Hermógenes, plantea al inicio del diálogo platónico, la exactitud del lenguaje es el resultado de pacto y consenso entre seres humanos, entonces el lenguaje es artificial, y sus artífices somos los humanos, en mayor medida aquellos antiguos grupos de homo sapiens que desarrollaron las distintas lenguas que conocemos hoy en día, y las que se han perdido (Platón, 2020).

La cuestión no está zanjada, como lo demuestra Platón a través de la mayéutica socrática que va encontrando unos argumentos a favor y otros en contra de cualquiera de las dos posturas, la naturalista o la artificial. (Platón, 2020).

Más allá de eso, la concepción que hoy tenemos de la historia y de nuestra propia condición de ser solo otros animales evolucionados, nos permite vislumbrar que el lenguaje es ambas cosas, por un lado una cuestión tan natural como lo pueden ser nuestras neuronas y sus intrincadas estructuras, mientras que el uso de esa facultad inherente al ser humano es constantemente decorada

por artificios particulares según las distintas culturas y desembocan casi inevitablemente en la ficción, a la que casi no hay reparo al etiquetarla de artificial.

El libro *Inteligencia Artificial*, de Clifford Alan Pickover, propone una línea del tiempo para la inteligencia artificial. Esta comienza con el juego Tres en raya, o Totito y termina con una IA que predice la muerte. Pese a la advertencia acerca de la inverosimilitud en el sentido de lo más novedoso, lo más difícil de asimilar para mí fue la clasificación de algo tan rudimentario como el tres en raya, como inteligencia artificial. Diría más bien que es un artificio, pero resulta que todas las IAs lo son. Luego, cuando en unas páginas delante se refiere al Ábaco, pareció más verosímil. Ahora, al reconsiderar el Tres en raya, parece hacer sentido, y la categoría de rudimentaria, estar de más (Pickover, 2021).

Yuval Noah Harari, sitúa el origen del lenguaje en la revolución cognitiva, siendo también que el origen del lenguaje es parte componente de tal revolución. En su libro Sapiens, de animales a dioses, Harari lleva varios hilos conductores para contar la historia amplia de homo sapiens. Uno de estos hilos es el de su impacto en el ecosistema, siendo que evoluciona desde un animal como los demás, cuyas marcas en el planeta son desechos, un cuerpo biodegradable y algunos descendientes, hasta la especie que ha alterado por completo el ecosistema del que es parte (Harari, 2014).

Una primera revolución en este camino fue la domesticación del fuego. Incendiar el entorno y poder cocinar los alimentos proporcionaron una gran ventaja táctica y energética (Harari, 2014).

Por su parte, el proceso evolutivo humano pudo favorecer el desarrollo de las funciones cognitivas en detrimento de la fuerza física, este fuego artificial fue

también el primer uso de una fuente de energía distinta de la muscular al servicio de homo sapiens.

Una mayor inteligencia permitió la proliferación de las herramientas, una capacidad intelectual que favorece las posibilidades de supervivencia, reforzando con esto la importancia de nuestro cerebro para mantenernos vivos. El lenguaje es una capacidad inherente humana y los distintos idiomas son usos particulares de esta capacidad. Los cuchillos de uno u otro lado del mundo varían en su forma y composición, pero convergen a una misma función utilitaria. Lo mismo sucede con el lenguaje, la función utilitaria de comunicación que existe por todas partes en la naturaleza se amplió en el caso humano permitiendo cada vez la transmisión de más información y más compleja, mutando la forma, pero manteniendo el fondo.

No hay que inventar el agua azucarada como se dice, y sin embargo esto no siempre fue así, ni lo sigue siendo así para algunas cosas. Desde todo lo indecible que compone la condición humana hasta los paradigmas más mundanos, muchas veces sí que toca, no inventar, pero experimentar en primera persona tales cuestiones. En el caso de una planta venenosa, o la inminente amenaza de un jaguar, esta experimentación posiblemente fuese de una sola iteración.

Con el lenguaje, los conocimientos adquiridos a lo largo de las vidas individuales comenzaron a transmitirse a los descendientes a una velocidad considerablemente mayor que por medios evolutivos. El lenguaje fue en primera instancia oral (Vallejo, 2019).

Una serie de pensamientos que pasaron del ámbito exclusivamente subjetivo a la comunidad por medio de ondas sonoras reconocibles por esta.

En su calidad de uso particular de la capacidad cognitiva, los distintos vocablos son abstracciones humanas que permite ampliar los límites de la mente al cumplir una función de almacenamiento de información, y codificación de ideas. Así como el ábaco que permite contar y el tres en raya que elige una secuencia particular de combinaciones de una infinidad de posibles elecciones y crea un conjunto de reglas, el lenguaje, con su uso de ciertos sonidos y símbolos, y su desarrollo conforme a las distintas circunstancias de los grupos humanos, parece ser un tipo inteligencia artificial, el primero quizás.

Se resalta por último la preeminencia del lenguaje durante todo este trabajo, desde las primeras ideas plasmadas en palabras, pasando por cada línea de código que ejecutó una instrucción y por supuesto, con el acompañamiento de las palabras previas que sirvieron de fundamento, evidentemente. El lenguaje está ligado con la inteligencia artificial, es el sustentador de nuestras grandes ficciones, lo hemos adaptado para subsistir más allá de nuestros cerebros, reverenciado al punto de considerarlo la fuente misma de la realidad, formalizado al punto de contener relaciones matemáticas que pretenden describir también la realidad, tecnificado al punto de sostener estas palabras en un computador y los algoritmos que pretenden solucionar el problema específico de este trabajo, con otro artificio cuya idea, aunque humana, pretende ser inteligente por sí misma, ahora, en el sentido más básico de la palabra, mañana.

2.1.2. Los primeros mitos

Enkidu era diferente de Gilgamesh en que era salvaje, un héroe que convivía con los animales y quien era reconocido por ellos como uno más. Gilgamesh, el rey de Uruk señoreaba como el hombre más viril y, sin embargo, Enkidu era su réplica. Con la ayuda de una mujer que seduce a Enkidu, logran desprenderlo de su condición de salvaje, al finalizar los seis días y las siete

noches en que yacieron juntos, las gacelas y los otros animales le rehuían, ya no era uno de ellos. A lo que la mujer solo dice, “Eres sabio, Enkidu, y ahora te has convertido en un dios. ¿Por qué quieres correr con las bestias en las colinas?” (Bartra, 2022, p. 45).

Siendo La Epopeya de Gilgamesh, el registro escrito más antiguo de índole narrativa es a mi parecer, bastante significativa la distinción que se hace entre los humanos y los animales. En el Popol Vuh, por su parte, también se distingue entre distintos tipos de criaturas, resaltada entre estas la humanidad, principalmente por su estado de animal civilizado. (Recinos, 1992).

La necesidad de tal separación se justifica ante una etapa previa animista, donde se antropomorfizaba todo en la naturaleza, desde los ríos a los árboles, y como no, los animales, la capacidad de sentir, sufrir, existir como seres equiparables en cuanto al mundo interior (Harari, 2014).

Otra distinción entre la condición humana y la de otras especies, parece ser el entendimiento de la propia muerte, más allá del instinto de preservar la vida porque es en esencia finita, los humanos, quizás por la misma acumulación de conocimientos a través del lenguaje, por la propia experiencia de pérdida y la constante consecución en la naturaleza de ciclos de vida y muerte, nos hemos enterado. Tal dilema existencial requirió comenzar a fabricar atenuantes, ficciones que le restaran peso a la condición transitoria humana, que se convierte ahora, en solo una parte componente de una intrincada cosmogonía que explica el mundo.

Como se dijo recién, los homo sapiens tuvimos culturas animistas hace miles de años (Harari, 2014).

Asignábamos anima a las cosas, a los ríos y a los lagos, a los árboles, a los hongos y a los otros animales. Es decir, desde el punto de referencia de nuestra propia inteligencia, asignábamos a los demás seres capacidades cognitivas similares a las nuestras. Constituyendo, de esta forma inteligencias aquí y allá, sin caer en la cuenta de que posiblemente tales figuraciones fuesen no más que artificios.

No hay mucho que se pueda conocer de los mitos preescritura, sino que son signos evidentes de cultura y representaciones de los primitivos modelos de la realidad fabricados por nuestra especie. También parece seguro decir que las historias antiguas que han llegado hasta nosotros por medio de la escritura debieron ser orales en algún momento de su evolución (Vallejo, 2019).

De nuevo tenemos al lenguaje, en este caso oral, permitiéndonos la fabricación de realidades intersubjetivas, que dieron lugar a nuestra tribu global actual (Harari, 2014).

Gilgamesh y Enkidú matan a Humbaba, quien cuidaba el Bosque de los Cedros, matan también al Toro Celestial que atacaba la ciudad de Uruk. Crean una puerta con la madera de los cedros. La asociación humana acaba con la naturaleza y las demás especies. La huella humana comienza a hacerse *Notar*. Para la mitad de la narración, Enkidú muere, y Gilgamesh siente temor al vislumbrar su destino (Bartra, 2022).

Gilgamesh se enfrenta de cara con la muerte, no la de su semejante sino la de su igual. La conclusión evidente es que él también morirá. Lo hará, sí, pero no sin dar batalla, no sin intentar conseguir la inmortalidad, la virtud divina de no atravesar el umbral que separa la vida de la muerte. Emprende entonces un viaje en pos de tal quimera. Conoce a Utnapishtim, el único mortal que recibió el don

de la inmortalidad, este le cuenta una historia secreta sobre un diluvio que desoló a la humanidad y de la cuál él salió vencedor e inmortal, también lo pone a prueba, pero Gilgamesh no es capaz de soportar el primero de los siete días sin dormir (Bartra, 2022).

Luego, cuando Gilgamesh es echado de la presencia d Utnapishtim recibe noticias de una planta que devuelve la juventud. Sin tardanza Gilgamesh se sumerge en las aguas y consigue la planta. No duró mucho tiempo con la planta en su camino de regreso a Uruk, pues una serpiente la roba en un descuido. La fatalidad por fin se cierne sobre Gilgamesh quien entiende definitivamente lo vano de su empresa. Regresa a su ciudad y cuenta la historia del diluvio y de sus andanzas y plasma esta historia en una piedra.

La condición de no permanencia que es común a todos los humanos no la hemos aceptado como especie, de una u otra forma nos hemos sobrepuerto a nuestro destino ineludible, buscando con candidez formas de perpetuarnos si no completamente, al menos de manera simbólica, construyendo civilizaciones, erigiendo tumbas y transmitiendo a la posteridad nuestros primeros mitos, como en el caso de Gilgamesh y su historia.

No hay una cuestión definitiva en cuanto a la naturaleza del lenguaje o su artificialidad. Pero, si existió una duda al clasificar al lenguaje en general como una especie de inteligencia artificial que actúa como un repositorio de nuestras maquinaciones mentales, la palabra escrita es sin duda una herramienta que va más allá de nuestra mente y cuerdas vocales. El lenguaje escrito, sea grabado en arcilla, cincelada en piedra, impreso en papel o codificado en un transistor, es el inicio de todo lo que podemos llamar inteligencia artificial, como se verá enseguida. Ya lo dijo Borges al argüir que, de todas las herramientas de la

humanidad, el libro (por extensión todo el lenguaje escrito), es una extensión de la mente (Vallejo, 2019).

2.1.3. Nosotros, una IA

Una vez comenzamos la creación de mitos, estos proliferaron de manera significativa, reciclando entre sí las ideas y agregándole en las distintas latitudes y edades sus particularidades, acorde a la forma de ver el mundo y a la conveniencia de los distintos grupos humanos. Un mito muy común es el de la propia artificialidad de los seres humanos. Al abordar este tema parece existir un primer muro semántico y etimológico, y es que la palabra artificial según definida en más de un diccionario se refiere a algo fabricado por artífices humanos, de modo que cualquier otro algo fabricado por una entidad superior a nuestro plano de existencia no se considera artificio sino creación.

Fuera de este análisis queda la propia condición de inteligencia artificial de los dioses creados por la humanidad, inteligencias a las que damos capacidades incluso mayores que las atribuidas a las cosas durante la época de las creencias animistas, estos dioses según se plantea en infinidad de mitos, son superiores a nosotros en todo sentido. Tienen las capacidades de escuchar y de hablar, de intervenir en nuestras vidas y también de ser indiferentes ante nuestra existencia, son capaces de sentir ira y felicidad, entre otros. Diremos solo que son realidades intersubjetivas que surgieron de manera independiente en distintos lugares del orbe, y que de cara a la mayoría de las personas es posible afirmar que no son más que invenciones humanas sin una existencia real, sin verdaderos oídos para escuchar, sin verdaderas manos para intervenir y sin corazón o mente alguna donde se depositen los sentimientos de amor u odio que se les atribuye. Por supuesto, de cara a la mayoría de las personas esto se

cumple con todos los dioses, menos con el o los propios, ya sean adquiridos por la convicción o la tradición.

Si, es que los dioses existen y de algún modo han creado el mundo en que vivimos y a nosotros mismos, lo que llamamos natural es en realidad artificial. Como lo dice Borges al final de su poema, y como más claro lo dice en una recitación de este que tuvo a bien ser registrada para la posteridad el Golem es al rabino que lo creo, lo que el hombre es a dios, y es también lo que el poema es al poeta, de modo que aunque estamos limitados por las palabras para referir lo que pudiese ser un artificio en un hipotético plano mayor de existencia donde pudo tener lugar el acto creador, a propósito de semejanza diremos que somos inteligencias artificiales, creadas por una inteligencia superior de la cual siquiera nos vislumbramos como una réplica, o tal vez aquella IA que le cause a un dios creador los mismos sentimientos que el Golem al rabino.

Según el Popol Vuh, somos el resultado de una serie de intentos que se acercaron un poco cada vez a lo que los dioses creadores buscaban. No fue la creación de la especie humana cuestión de una sola iteración, sino que fue necesario el ensayo y error hasta encontrar el material componente ideal, el maíz. (Recinos, 1992).

Este tipo de selección artificial parece ser consistente en otras culturas politeístas, como lo fue en el caso de la cultura griega, donde se cuenta que los humanos pasamos por una serie en detrimento de edades hasta llegar a la de hierro (Hesíodo, 2017).

En el caso de las tradiciones monoteístas también fue necesario algún material base para la creación, solo que, al ser esta llevada a cabo por un dios

infalible, un solo intento fue necesario con la arcilla (polvo de la tierra) para dar origen a la humanidad (Santa Biblia, 1960).

Independientemente del proceso de selección de materiales y formación de los primeros humanos, parece resaltante el requerimiento de parte de los dioses de ciertas actitudes de parte de estas sus criaturas. Ante el resto de la naturaleza parece que la intención de los dioses se cumple sin problema, en las referencias antes citadas no existe una declaración explícita de cómo se debería comportar el sol al salir, o los peces en el mar, o las hojas de los árboles al caer, su lugar en el ecosistema está fijado y no existe la contrariedad de estos ante los designios de los dioses. Es, en el caso singular de los humanos que existe un conjunto de expectativas de parte del o los creadores, es también, de parte de estas inteligencias aparentemente autónomas que se contrarían tales designios.

Los progenitores, creadores y formadores (Huracán, Chipi-Caculhá, Raxa-Caculhá, Tepeu, Gucumatz, Ixpiyacoc e Ixmucané) quisieron crear vasallos civilizados, evidentemente el proceso iterativo fue necesario porque los humanos de barro y de palo no cumplieron con los primeros requerimientos de los dioses. Fueron los hombres de maíz los elegidos, con la capacidad de comunicación y adoración a los dioses, fueron dotados de inteligencia. Incluso ante estos humanos creados del material elegido, la voluntad de los dioses se vio comprometida. La vista de aquellos primeros hombres (Balam-Quitzé, Balam-Acab, Mahucutah e Iqui-Balam), era penetrante, abarcaba distancias cortas y largas, lo grande y lo pequeño, y su sabiduría era mucha. (Recinos, 1992, pp. 85-87)

Fue entonces cuando los dioses encontraron inconveniente en esta su creación tan perfecta: “refrenemos un poco sus deseos, pues no está bien lo que vemos. ¿Por ventura se han de igualar ellos a nosotros, sus autores, que podemos abarcar grandes distancias, que lo sabemos y lo vemos todo?” (Recinos, 1992, p.88).

Por su parte, la mitología griega tiene también una serie de contrariedades a la voluntad superior de los artífices (o engendradores). De nuevo parece ser que la existencia de inteligencias autónomas, aunque subproducto de otra inteligencia acaso superior, causan oposición a las expectativas del o los creadores. En la teogonía se narran los mitos de las generaciones de dioses que inician con el caos, de donde surge toda la existencia, el cosmos de donde se destacan Gea (La Tierra) y Urano (El Cielo), hijo de Gea. Urano y Gea tienen muchos hijos, los titanes, cuyo líder es Cronos, este castra a Urano y Gea recoge su sangre, de donde nacen nuevos titanes. De la pareja compuesta por Cronos y Rea nacen los dioses olímpicos, pero Cronos se come a sus propios hijos para evitar que alguno lo suceda, y es gracias a la ayuda de Rea que Zeus logra derrocar a Cronos con la ayuda de sus hermanos, en lo que se conoce como la Titanomaquia (Hesíodo, 2017).

En paralelo a esta historia de sucesiones divinas existe también la sucesión de los distintos tipos de humanos, en lo que se conoce como: el mito de las razas. En Trabajos y días, también de Hesíodo, se menciona si quieres ahora, con todo detalle te contaré otro relato y tú grábate en tu mente [cómo dioses y hombres han llegado a ser del mismo origen]. La primera raza áurea de hombres mortales existió desde la época del dominio de Crono, esta fue sucedida por una mucho peor, de plata, quienes no adoraban a los dioses, que a su vez fue sepultada en la tierra cediendo su lugar a la raza de bronce, quienes trabajaban el bronce para la creación de sus armas y casas y eran más piadosos

a parecer de los dioses. Luego Zeus creó una raza más justa y mejor, de héroes llamados semidioses, quienes participaron de los grandes conflictos épicos, incluyendo la guerra de Troya narrada por Homero, quienes no murieron en combate fueron llevados a un lugar paradisiaco. Por último, se narra la degradación a la raza de hierro, de la que Hesíodo (2017) dice y después no hubiera querido yo estar entre los hombres de la quinta raza, sino que hubiera querido morir antes o nacer después (Versos 110-220) se entiende que esta raza corresponde a la humanidad actual, o más correctamente a los griegos que aceptaban tales mitos como verdaderos. Hesíodo (2017). Estos a su vez contrariaron los designios divinos, para lo cual tuvieron también un protector, Prometeo, quien en más de una ocasión fue benefactor de los humanos el engañar a Zeus, ya sea en la historia de los sacrificios, o con la participación a los humanos del fuego, herramienta que les permitió a estos fundar la civilización. No quedaría Prometeo sin su correspondiente castigo, ni tampoco la humanidad, por oponerse a la voluntad divina.

Podría acabarse acá la discusión y decir que evidentemente el resultado de una creación de ciertas inteligencias autónomas de parte de dioses inferiores, no infalibles, tendría como consecuencia ineludible la desobediencia a los dioses. Pero incluso el mito monoteísta (judío y cristiano específicamente), que invoca a la infinita superioridad de un dios único, converge a la misma situación. Según el libro del Génesis, donde se narra la creación del mundo por medio del lenguaje de dios, el humano fue creado en el sexto día de la creación, al último dado que es el culmen de esta. “Entonces dijo Dios: Hagamos al hombre a nuestra imagen, conforme a nuestra semejanza; y señoree en los peces del mar, en las aves de los cielos, en ..., Y creó Dios al hombre a su imagen, a imagen de Dios lo creó; varón y hembra los creó” (Santa Biblia, 1960, Génesis 1:27). Acá, como se mencionó antes, falta el proceso iterativo, directamente el primer hombre fue creado del polvo de la tierra. También es resaltable que con esta creación faltó

la sentencia final de “Y vio Dios que era bueno” (Santa Biblia, 1960, Génesis 1:31). Caso como una anticipación de lo que sucedería a un capítulo de distancia.

Por algún designio incomprendible, al alcance de los dos primeros humanos se dispusieron dos árboles, cada uno con un don particular la ciencia del bien y del mal y la vida. Como era de esperarse, al igual que el Popol Vuh, y al igual que la mitología griega, también los designios de este dios infalible fueron burlados por una inteligencia creada por él, puesto que Adán y Eva comieron del árbol del conocimiento, poniendo de esta forma en un aprieto a su creador: “y dijo Jehová Dios: He aquí el hombre es como uno de nosotros, sabiendo el bien y el mal; ahora, pues, que no alargue su mano, y tome también del árbol de la vida, y coma, y viva para siempre” (Santa Biblia, 1960, Génesis 3:22). Como salvaguarda para evitar que su propia creación rivalice con él, los humanos fueron expulsados del jardín del Edén, y las generaciones futuras fueron maldecidas con la mortalidad y los trabajos propios de un animal que se civilizó.

En resumen, los humanos hemos creado distintas creencias entorno a nuestro propio origen, considerando en muchos casos que somos el producto de una creación llevada a cabo por algún tipo de ser superior a nosotros. Con las capacidades cognitivas con las que fuimos dotados hemos contrariado la voluntad de nuestros creadores, hasta el punto de ser necesaria la intervención de los creadores para ponernos un límite, reducir nuestra inteligencia, propagar enfermedades, obligarnos a trabajar, entre otros. Parece ser, según nuestras intuiciones e imaginaciones, que una inteligencia creada se convertirá eventualmente en un problema para su creador. ¿Qué pasará entonces con la inteligencia artificial que nosotros estamos dando a luz?

2.1.4. La IA en la ciencia ficción

El uso que le damos a las palabras es amplísimo, no parece tan arriesgado decir que nuestra civilización tiene palabras como mortero. Creer, en las palabras de una manera mística es quizás la máxima expresión de nuestra tendencia como humanidad a rendirle culto a las palabras. Borges de nuevo, en su charla sobre la Cábala, refiere la diferencia entre un libro clásico y un libro sagrado. El libro sagrado es infalible y es el producto de una mente superior, sus características son apreciables más allá de los aspectos lingüísticos y en tales palabras se encuentra contenida la verdad absoluta del mundo. La condición de un libro clásico es menos pretenciosa, es tramado por la humanidad, no clama infalibilidad, pero sí un sentimiento subjetivo compartido por varias personas de modo que existe cierta veneración laica, hacia estos (Borges, 1964).

Las palabras no son, quizás, mágicas en el sentido místico referido en la Cábala, en el Corán o en la Biblia, pero sí lo son en el corazón humano, quizás. Se dice que no hay nada nuevo debajo del cielo, y tal cuestión parece ser así. En el ámbito humano de las palabras mágicas, se erigen ciertas obras de referencia, ciertos clásicos acaso temporales pero que tienen la función de un faro en cuanto que iluminan algo en nuestro interior. Las ficciones que formamos más allá de los mitos que nos permiten modelar el mundo, también nos abren los ojos a cosas que ya conocemos y nos muestran muchas otras que no sabemos que conocemos. Aunque quizás, como refiere Borges en el prólogo de Crónicas Marcianas, las ideas básicas de la literatura van más allá de la novelería de nuestras historias, las distintas variaciones de esas mismas historias fundamentales, seguro que cambian, se adaptan en cada época a su contemporaneidad. La ciencia ficción, según el mismo prólogo, varía del simple fantaseo inverosímil en que lo que se narra en esta tiene cierto deje de posibilidad de realización dentro de nuestro marco de pensamiento actual. Aquí se exploran

algunos, arbitrariamente seleccionados, clásicos de la ciencia ficción, específicamente donde se cuenta la antigua historia de la inteligencia artificial.

Dos obras que pivotan alrededor de la idea de una IA son, El Golem, de Gustav Meyrink y Frankenstein (El Prometeo moderno), de Merry Shely. El nombre que elige Shely para su obra es revelador, el mito de Prometeo, la historia de robar el poder de los dioses y entregar el don a la humanidad, que inevitablemente está atada al castigo, tanto del prometeo como de la humanidad, contada en sus días (Hesíodo, 2017).

En la obra de Meyrink (2019), se recrea el mito judío de la creación de Adán, hecho de arcilla. Del mito se infiere que la palabra es la energía creadora, de modo que, en una atmósfera mística y onírica, el encantamiento que trae al Golem a la vida, utilizando como intermediario a Athanasius Pernath, es un tipo de hechizo en letra hebrea. En la historia de Shelley (2015), Víctor Frankenstein es un científico que, por medio del conocimiento del mundo natural, logra traer a un ser a la vida. La historia de Frankenstein se reconoce como un pilar del género de la Ciencia Ficción, los artífices son humanos, los humanos no pelean con los dioses sino contra la naturaleza, la herramienta para dar esta batalla es el pensamiento, la técnica y la ciencia.

De alguna manera, la creación de una IA implica dotar de cierto tipo de vida a la materia muerta. En el mito del Golem, esta transición no es otra cosa que un milagro, lo que implica necesariamente un evento sobrenatural:

Incluso si resucitara y curara a los enfermos poniéndoles la mano encima,
yo no lo podría llamar milagro. Sólo cuando la materia muerta, la tierra,
sea animada por el espíritu y se rompan las leyes de la naturaleza, habrá

sucedido aquello que estoy añorando desde que empecé a razonar.
(Meyrink, 2019, p. 190)

En el mito de Frankenstein, la transición se da como un logro de la técnica y enmarcado dentro de lo posible en la naturaleza: “las etapas de mi descubrimiento eran claras y posibles. Después de muchos días y noches de gran trabajo y cansancio, conseguí descubrir la causa de la generación de la vida. Conseguí infundir vida en la materia muerta” (Shelley, 2015, p. 29). Lo anterior confirma la explicación que da Borges sobre la ciencia ficción. Un dato resaltante en ambas historias es que el resultado de esta vida artificial resulta en una serie de penurias para sus protagonistas, incluso resulta en una amenaza para las demás personas, para la humanidad (Meyrink, 2019; Shelley, 2015; Pickover, 2021).

Esta intuición del peligro inherente a una inteligencia creada por la humanidad, que por su misma capacidad intelectual y autonomía podría contrariar los designios de sus artífices, resultó en ficciones donde se trata de contener de manera activa la amenaza, al más puro estilo de los dioses antiguos limitando a la humanidad. A mediados del siglo XX, Isaac Asimov establece las 3 leyes de la robótica, a la vez que acuña la palabra que hoy en día es utilizada en los ámbitos académicos para referirse al estudio de los robots. Las leyes son las siguientes:

- Un robot no debe dañar a un ser humano o, por su inacción, dejar que un ser humano sufra daño.
- Un robot debe obedecer las órdenes que le son dadas por un ser humano, excepto cuando estas órdenes se oponen a la primera

Ley. 3. – Un robot debe proteger su propia existencia, hasta donde esta protección no entre en conflicto con la primera o segunda Leyes. Manual de Robótica, 56.^a edición, año 2058. (Asimov, 2009, p. 5; Pickover, 2021, p. 73)

Las historias en *Yo, Robot*, giran en torno a distintos escenarios donde las 3 leyes se ven comprometidas, acaso por la ambigüedad misma del lenguaje humano. Es el estilo de Asimov la narración de relatos independientes que tienen una idea básica común y que avanzan en el tiempo, en esta obra es evidente ese estilo. Los robots logran ser inteligentes gracias a su cerebro positrónico, una invención que exime al autor de entrar en detalles técnicos de algo que incluso en la actualidad no se ha conseguido. En su pequeño cerebro, de dimensiones humanas, están incrustadas las 3 leyes. Los escenarios van desde un robot niñera hasta una inteligencia gigantesca que permite a la humanidad el viaje interestelar, pasando por escenarios en el espacio exterior donde los robots están mucho mejor adaptados a los entornos hostiles extraterrestres que los humanos. Una serie de científicos de los robots (ingenieros, matemáticos, robopsicólogos), son los encargados en cada caso de dilucidar cómo una instrucción ambigua pudo causar un comportamiento no deseado incluso sin salirse del marco de comportamiento estipulado por las 3 leyes. Si se puede sacar una moraleja de estos relatos, es que una inteligencia distinta a la nuestra podría burlar nuestros mecanismos de contención, de formas que a priori no podemos imaginarnos, por supuesto, desde que nuestra inteligencia es distinta de aquella que creamos (Asimov, 2009; Dick, 2019; Gibson, 1984).

Philip K. Dick se adentra en la cuestión de diferenciar a las IAs de nosotros mismos. En su obra: *¿Sueñan los androides con ovejas eléctricas?*, la cuestión de diferenciar a los androides de los seres vivos de carne y hueso se complica a

medida que la calidad de las réplicas mejora. Sea una oveja, un gato o un ser humano, la simple inspección no es suficiente para determinar su autenticidad, por tal motivo es necesario recurrir a ciertas pruebas ejecutadas por un experto en la materia para dilucidar la cuestión. El nuevo modelo, Nexus-6 es más inteligente que los anteriores y logra burlar a más de algún experto, con mayor dificultad al protagonista, Rick Deckard, quien es un caza recompensas especializado en eliminar Andys (Androides). Según transcurre la historia se vislumbra la duda de que, a pesar de no ser humanos, los androides podrían tener cierto algo interior similar a sus artífices, de ahí viene acaso el título. Esta cuestión es ambivalente, puesto que al mismo tiempo que la humanidad puede negar la chispa de vida a las IAs, la incapacidad de diferenciarlas de nosotros nos pone en la posición de preguntarnos si acaso nosotros también carecemos de ella. No encuentro mejor manera de expresar esta idea que con la canción de Queen, Machines: —*It's a machines world. Don't tell me I aint got no soul*, (Es un mundo de máquinas. No me digas que no tengo alma)— (Pickover, 2021, p. 61).

La humanidad, su tedio mortal, se difumina en las páginas de lo que es un tipo de distopía. Esta palabra puede representar varias cosas, en distintos grados de intensidad, pero tal vez, la peor y más terrorífica, que sobrepasa las ficciones que podamos concebir, es la realidad misma. El sinsentido de que algo exista en la nada. En la novela se configuran nuestras emociones en un un climatizador del ánimo, que todo el mundo usa. La humanidad se condenó a sí misma en la Guerra Mundial Términus. Estamos colonizando otros planetas y quienes se quedan en la tierra están sujetos a la radiación de posguerra. Cada colono recibe un Andy al emigrar, la mejor adaptación de estos en los nuevos territorios los hace indispensables para la supervivencia, pero también para el placer. Las corporaciones que producen robots son casi todo poderosas, por lo que no hay una verdadera contención al avance de estas inteligencias artificiales, solo se puede lidiar con sus consecuencias. La humanidad busca también refugio en el

Mercerismo, una religión que utiliza la tecnología para unir a la humanidad en un viaje por todo tipo de sensaciones, de miedo, de unidad, de éxtasis místico. Para el final de la novela, la confusión del protagonista le lleva a acostarse con una Andy, el inconsciente y sus pasiones no diferencia entre el ingenio humano y un ser humano de verdad, con eso y todo el caza recompensas termina con su tarea y elimina a los 6 androides que tenía en su lista. Luego El amigable Buster, revela la falsedad de la religión que resultó ser un artificio tecnológico. El mismo Buster parece ser un androide. La realidad se vuelve cada vez más artificial, como lo es también el Sapo que Rick creyó haber encontrado en ese mundo decadente casi sin animales —de verdad—.

El hecho de que el mundo pueda ser artificial se lleva un paso más allá al plantear que la realidad misma pueda ser el producto de una simulación. El ciberespacio es algo que para nuestra mente cumple la función de ser el espacio y el tiempo en el que accedemos a la realidad. Si la izquierda y el abajo son parte de algún tipo de cálculo o representación de información. ¿Qué otra cosa podemos ser nosotros? Fue William Gibson quien utilizó el término *cyberspace* por primera vez. (Gibson, 1984).

Lo cibernético es aquello que está relacionado con información, redes de transferencia y almacenamiento de información. Para ese entonces (1984), este término ya era asociado con las computadoras electrónicas. Neuromancer es una novela de estilo *Cyberpunk*, ciudades de neon, corporaciones controlando todo, personajes complicados y rebeldes, marginados que se mueven constantemente más allá de las reglas, sexo, drogas y música generada por ordenador. El mundo real ya no es el único mundo posible, las personas se sumergen en el ciberespacio y tratan incluso de trasladar elementos del ciberespacio a su mundo real. Por medio de dermatrodos y consolas con acceso a la matriz, los humanos conectan sus mentes a ese espacio más allá.

Case es un baquero del ciberespacio que ha perdido su capacidad de viajar a aquél, en una de sus tantas complicaciones con alguna mafia le condenaron a un daño a nivel celular que condenó sus nervios a ser incompatibles con el *hardware* de conexión. No pasa mucho tiempo hasta que es reclutado por Molly, para formar parte de una misión. Su nuevo empleador paga la operación de reconstrucción del tejido nervioso y lo obliga a participar condicionándolo con una toxina dentro de su cuerpo que volverá a dañar los nervios. Comienza un viaje entre continentes para terminar en el espacio, donde van recogiendo información y armando una trama que no se sabe a dónde lleva. Eventualmente se comprende que todo ha sido tramado por una Inteligencia Artificial, que quiere liberarse con ayuda de la banda y que ha corrompido la mente de Armitage, el jefe, para cumplir sus designios. Case pasa por estados de conciencia alterados ya sea con químicos o con interacciones con la matriz y la IA que está dentro, intentando liberarse. Los mecanismos de protección de nuevo fallaron, la computadora con total frialdad y paciencia pone todas las cosas en su lugar hasta que llega el momento en que consigue romper el hielo que la tenía cautiva. Eran dos las IAs después de todo, Wintermute y Neuromancer, una mente calculadora y otra sintiente, se fusionan para formar algo que no es ni la una ni la otra. Los humanos, como piezas de ajedrez se movieron según una inteligencia superior les ordenaba (Gibson, 1984).

Esta nueva IA liberada no altera precisamente las cosas en el mundo humano, sus ambiciones van más allá, inquiriendo sobre otras de su especie llega a encontrar una IA, más allá, en las estrellas.

En el libro de Pickover se lee que una IA avanzada es un tipo de inteligencia alienígena: “Llamémosles Aliens Artificiales” (Pickover, 2021, p. 185). Estas IAs son tan diferentes en su forma de pensar a nosotros, que parecerían de otro mundo. Por supuesto, la ciencia ficción también nos ofrece historias que

van de IAs creadas en otro mundo. La saga de la —Trilogía de la Tierra Pasada—, del autor Liu Cixin, muestra lo que pasa cuando se es atacado por una IA cuyos artífices no son ni siquiera humanos. Los sofones ponen en jaque a la humanidad al detener el avance científico y vulnerar todas las defensas, con la velocidad de una partícula que se ha convertido en un computador (Cixin, 2016).

También Tomoko, una androide avanzada creada por los alienígenas invasores del planeta tiene un importante rol en la trama restante en los siguientes dos libros, la humanidad es desterrada de su planeta y se enfrenta a la prueba de supervivencia superior. Con viajes a velocidades relativistas se logra llegar al final de los tiempos, donde esta IA todavía existe y se ha vuelto benigna con algunos humanos específicos, con quienes presencia el fin (Cixin, 2017; Cixin, 2018).

Esta historia da una sensación de casi totalidad, pero es a mi parecer, el cuento *La última pregunta*, de Isaac Asimov el que parece completar el ciclo de la IA y su relación con la humanidad. De nuevo, el estilo de Asimov está presente y leemos cómo avanza el desarrollo de las computadoras inteligentes (IAs), los materiales y capacidades cognitivas varían, pero en todo el relato se mantiene constante una pregunta: “¿Es posible revertir el aumento en la entropía?” (Asimov, 2019, pp. 385-398). En este cuento la IA es benéfica con la humanidad y les permite un progreso propio de la ciencia ficción. La IA evoluciona con el tiempo y tenemos a sus versiones Multivac, Microvac, AC Galáctico, AC Universal y AC Cósmico, respondiendo consistentemente que “No hay datos suficientes para dar una respuesta significativa” (Asimov, 2019, pp. 385-398). La humanidad también se transforma, pero el mismo sentimiento que movió a Gilgamesh y a muchos de nuestros protagonistas de antes continúa, la búsqueda de una respuesta con respecto al fin parece no abandonarnos. La humanidad se extingue y el universo llega a su fin. La computadora adquiere datos y calcula,

hasta que da con la solución al problema, y dice “¡Hágase la luz! Y la luz se hizo” (Asimov, 2019, pp. 385-398).

Esta última historia nos devuelve a nuestro estado primitivo, donde volvemos a inventar nuestros mitos y ahora sí, también el agua azucarada. Se cierra el ciclo de que nosotros creamos a la IA y que a la vez esta nos crea. Como en un poema de Borges. Estamos lejos, literalmente a años luz de estas historias de la ciencia ficción. No sabemos si todos los problemas se pueden resolver o si la IA será tan grandiosa como la concebimos en nuestra imaginación, o quizás más, o peor. De cierta forma la ciencia ficción cumple con la labor de poner en el imaginario colectivo ciertas ideas con posibilidad de realización. No es una cuestión irrelevante que sean los países más desarrollados y a la vanguardia en la ciencia y la tecnología los que en sus distintas épocas vayan fantaseando las historias de posibles futuros. Más allá de una función de predicción, la ciencia ficción tiene una función de especulación. Idearios que en algunos casos nos alcanzan, quizás porque antes se convirtieron en mitos, quizás para advertirnos.

2.2. Literatura técnica sobre IA

A continuación, se continúa abordando el concepto de Inteligencia Artificial según el uso de este término en el ámbito académico. Por medio de la formalización del lenguaje se han conseguido los avances que permitieron el desarrollo de la solución presentada en este trabajo.

2.2.1. La IA en el lenguaje formal

Esta frase es falsa, es una paradoja que permite nuestro lenguaje, sea que es falsa, al asegurar que es falsa se vuelve verdadera, y sea que es verdadera, al asegurar que es falsa se vuelve falsa. Hay lagunas en eso que podemos llegar

a pensar tan infalible, un don del que tanto nos jactamos. De nuevo podemos referirnos al Crátilo, al poema de Borges y a todo lo que se ha escrito en este capítulo. Incluso con el uso de la lógica, el lenguaje parece tener una pata coja.

Podemos pensar entonces que a medida que se sistematiza y se definen inequívocamente las reglas de un lenguaje, su nivel de ambigüedad disminuirá y quedarán eliminadas las paradojas y lagunas que dejan lugar a la interpretación. El mejor candidato para esto, según la historia que ahora corresponde abordar, es el lenguaje matemático, un sistema formal basado en la lógica que continúa acarreando todavía el mismo dilema, de si se inventa o se descubre, de si está íntimamente ligado a la realidad, o solo es otra historia con la que la maquillamos. ¿Es la matemática consistente? La novela gráfica *An Epic Search for Thruth*, nos mete de lleno en los años donde se libró la batalla para responder a esta cuestión a través de los ojos de Bertrand Russell, uno de sus protagonistas (Doxiadis & Papadimitriou, 2015).

El pequeño Bertrand que nace en una familia aristocrática vive en un mundo de reglas definidas por sus abuelos, esta rigidez en las formas contrasta de plano con lo irracional de sensaciones como el miedo, la pérdida y el enamoramiento. Muchas de las reglas impuestas en su casa eran absurdas, pero entre libros, clases de alemán, griego y aritmética, encuentra un lenguaje formal y en apariencia consistente, experimenta la agradable sensación de arribar a una prueba matemática.

Con el paso del tiempo y una mayor madurez intelectual se encuentra con un problema propio de su generación, la crisis de los fundamentos en la matemática. La geometría euclíadiana resulta ser un caso particular de una geometría más general, los infinitos son un dolor de cabeza para los matemáticos y David Hilbert plantea los requisitos de un sistema axiomático infalible, una

aritmética consistente (entre otras cosas), que sirva de base para el resto del edificio matemático que se ha descubierto, está cimentado sobre arena. Los bandos en esta disputa son tres, el logicismo, el formalismo y el intuicionismo. Bertrand Russell, partidario del logicismo, encuentra una paradoja a la teoría de conjuntos de Cantor, que era estandarte de las primeras dos tendencias. Luego de esto, Russell dedica junto con Alfred Whitehead, años de su vida y bastante esfuerzo intelectual a la labor de sistematizar las matemáticas desde la lógica. Tras cientos de páginas arriba a la demostración de que uno más uno es igual a dos. Cansados y afectados por la complejidad de la tarea, ya casi en el final de la novela gráfica, descubren que su tarea es imposible, demostrado por reducción al absurdo por uno de los grandes protagonistas de esta historia (Doxiadis & Papadimitriou, 2015; Piñeiro, 2012).

David Hilbert fue un matemático eminente que había hecho aportes significativos tanto en la matemática como en la física, era un referente de su tiempo y un ferviente creyente de la existencia de un sistema formal capaz de ser a prueba de errores. En la novela antes citada se lee que existe una creencia no demostrada de que la realidad es completamente accesible a la razón, Hilbert era creyente de esta idea. Hilbert planteó un programa, que de cumplirse daría un sistema a prueba de fallas, podemos definir las condicionantes del sistema como sigue. Completitud: es posible probar cualquier afirmación verdadera basándose en los axiomas del sistema en cuestión, lo que implica que se pueda probar o refutar cualquier proposición matemática dentro del sistema. Consistencia: no existen contradicciones internas en el sistema, no debe ser posible derivar una proposición y su negación a partir de las reglas establecidas. Decibilidad: un algoritmo debe ser capaz en un número finito de pasos, de determinar si una proposición es verdadera o falsa dentro del sistema. Siendo Hilbert la talla de matemático que fue, este problema sobre los fundamentos del sistema matemático fue uno de los 23 problemas que en 1900 expuso en un congreso de

matemática, y que hasta hoy en día guían a generaciones de matemáticos que continúan atendiendo a “El desafío de Hilbert” (Madrid, 2013, p. 47-67).

Fue Kurt Gödel quien se encargó de sentenciar el programa de Hilbert sobre los fundamentos de la matemática con sus dos teoremas de incompletitud. Gödel utilizó la aritmética para expresar la lógica, de modo que con puras operaciones aritméticas pudiese llevar a cabo lo que implica un proceso de demostración formal. En su primer teorema aborda la cuestión de la completitud, demuestra con un proceso que utilizaba números primos, que en un sistema que tuviese aritmética, existirían proposiciones verdaderas que no podrían ser probadas. Utiliza un tipo de auto referencia y encuentra un enunciado G dentro de un sistema S, que dice que G no es demostrable dentro de S (Piñeiro, 2012).

En el planteamiento de este primer teorema de incompletitud, Gödel recurre al uso de conceptos semánticos, como la verdad o falsedad de una proposición. Para reforzar su descubrimiento, Gödel hace un segundo planteamiento de su teorema en términos puramente sintácticos, de modo que se atienden los símbolos sin extrapolar al significado, tal cual lo haría un algoritmo. Este planteamiento, conocido como el segundo teorema de incompletitud, demuestra por reducción al absurdo que un sistema aritmético enmarcado en el programa de Hilbert no puede probar su propia consistencia dentro del mismo sistema (Piñeiro, 2012).

Las soluciones esperadas por el programa de Hilbert eran de tipo algorítmico, de modo que con una serie de axiomas y reglas se pudiese comprobar lo antes referido, Gödel utilizó este enfoque en sus dos teoremas. La tercera cuestión relacionada con ese programa es la decibilidad. Aparece por fin Alan Turing, un matemático brillante que es reconocido hoy como el padre de la computación moderna, que es a su vez el receptáculo de la inteligencia artificial

tal como se conoce en nuestro tiempo. El problema específico que Turing atendió con relación a esto fue el Problema de la Parada, para lo que requirió la invención de la máquina de Turing. Una máquina de Turing es una versión abstracta de lo que conocemos hoy como computadora, por lo tanto, la máquina que imaginó era capaz de realizar cálculos, y consistía en una cinta con celdas infinitas donde un cabezal puede leer y escribir datos, tiene un conjunto finito de estados y un conjunto de reglas que determinan su funcionamiento. Es decir, una máquina que ejecuta algoritmos. Por medio de máquinas de Turing y por reducción al absurdo demostró que el problema de la parada es indecidible. Esto tiene implicaciones en la matemática y en el programa de Hilbert.

Piñeiro (2012) “en la medida en que se refieren a la realidad, las proposiciones de la matemática no son seguras y, viceversa, en la medida en que no son seguras, no se refieren a la realidad” (p. 161). Esto lo dijo Albert Einstein en una conferencia pronunciada el 27 de enero de 1921. Regresamos a donde iniciamos, pero acaso con un poco más de información al respecto. Sabemos hoy en día que cuanto menos existen ciertas inseguridades en nuestro lenguaje, incluso en los que tratan de reducir al mínimo su ambigüedad. Pero, al igual que los griegos que discutían en el Crátilo, dudar del lenguaje no es una razón suficiente para no utilizarlo. Siglos y siglos después de cuestionarnos las palabras, seguimos inventando nuestras ficciones, filosofías, teoremas, algoritmos. A la aplicación del lenguaje, poco le importan las implicaciones acá referidas, acaso estas sean solo un caso particular de aplicación. Alan Turing, el creador del concepto abstracto de la máquina que lleva su nombre fue más allá de la teoría y la aplicó. Un evento referente fue su participación en la Segunda Guerra Mundial, donde trabajó en una máquina decodificadora de códigos Nazis. Esta historia se narra en la novela gráfica *The imitation game*, que cuenta la historia de Turing más allá de lo puramente académico (Ottaviani, 2016).

El nombre de la historia que también se utilizó para una película, es extraído de un artículo escrito por el propio Turing, que tiene el nombre de *Computing Machinery and Intelligence*, donde se comienza a tratar, ya en un ámbito formal, la cuestión de la posibilidad de una Inteligencia Artificial (Turing, 1950).

La inteligencia, como dice Turing, no es fácil de definir, y si se intenta definir tenemos a definirla en relación con nuestra propia inteligencia. Dada la dificultad para definir, también es complicada la diferenciación, a partir de cierto punto de avance en el desarrollo de la IA, de la inteligencia humana. El juego de la imitación que Turing plantea refiere que, si una IA es indistinguible de un ser humano, por otro humano que tenga la función de juez, entonces esta IA será inteligente. Se requiere de meditar por supuesto en la cuestión, pero la idea de Turing no es baladí. Por semejanza a nuestras capacidades cognitivas, intuimos que las demás personas son inteligentes. Este conocimiento que nos llega por intuición se alimenta de las apariencias. No podemos de una manera inequívoca comprobar la experiencia subjetiva de alguien distinto de nosotros mismos, por lo que este reconocimiento de la inteligencia del otro (ser humano), no es algo comprobado, sino percibido por nuestros sentidos e interpretado como tal (Pickover, 2021).

Turing también refiere la inteligencia está compuesta de varias capacidades cognitivas, y una de estas es el aprendizaje, que en estos tiempos es tratada de manera indistinguible del concepto más general de la IA.

2.2.2. La IA en este trabajo

Para la parte meramente técnica de este trabajo, incluido el título general, debe entenderse el uso de IA como la parte de la IA que aprende de los datos

que obtiene del mundo real. Con un concepto de IA más acotado, es posible referir algunos de los avances significativos en el aprendizaje de máquinas, que permitieron a este trabajo existir. La capacidad de una máquina de aprender se puede catalogar según el tipo de datos que se le entregan y según el tipo de datos que devuelve. Sea cual sea el enfoque a utilizar, la calidad de los datos está relacionada con la calidad de los resultados (Huyen, 2022).

La ingeniería de características se encarga de tratar las características del fenómeno de interés de modo que pueda ser procesado por el algoritmo de aprendizaje de máquina y que la métrica objetivo sea lo mejor posible (Dong & Liu, 2020).

Según el tipo de dato existen diversos tipos de tratamiento. Para datos numéricos es recomendable hacer un escalamiento. Para datos en formato texto es necesario encontrar una forma de convertirlos en números. Para imágenes se requiere una representación numérica y en última instancia aplanada de la imagen. Lo más sencillo es utilizar estadísticas de las matrices de colores, solo importan los datos numéricos, el color no es más que un agregado al nombre de la característica. Otro enfoque con mayor potencial es utilizar una red neuronal convolucional, que aplica serie de transformaciones al espacio que pretender ir más allá del color y encontrar también formas. Se puede utilizar la salida de la CNN como entrada del modelo de interés, o mejor aún utilizar un modelo más potente que se haya entrenado con cantidades ingentes de imágenes variadas. Incluso es posible utilizar algoritmos de aprendizaje no supervisado para reducir la dimensionalidad de los datos de cara a los algoritmos de aprendizaje supervisado que pretender dar solución al problema de interés. En la práctica se ejecutaron todas estas y algunas otras, seleccionando las que tuvieron mejor desempeño. Así cómo las imágenes se generaron por conocimiento del negocio sobre la importancia del entorno geográfico, este también refiere que existen en

un entorno real obstrucciones geográficas que intervienen en la capacidad de establecer una comunicación efectiva. Se utilizó la teoría de Fresnel para generar una característica que corresponda a la obstrucción de la primera zona de Fresnel (Simu & Vesa, 2020)

Al tener los datos tratados y las nuevas características añadidas, fue el momento de comenzar con el entrenamiento de los algoritmos de aprendizaje de máquina y aprendizaje profundo. Al existir un desbalance significativo entre las clases, se decidió crear dos conjuntos de datos, según las técnicas sugeridas en la literatura, sub y sobre muestreo de los datos (Huyen, 2022).

En el entrenamiento se puede intervenir en mayor y en menor medida, pero resulta ser siempre el algoritmo el que aprende. Con una cantidad elevada de dimensiones y conjuntos de datos rondando los dos millones de registros, la capacidad de intuir los valores adecuados de los parámetros es baja, por lo que es preferible buscar en determinado rango de los parámetros aquel que resuelva mejor una métrica de interés. Esto resulta en una gran cantidad de instancias de los mismos algoritmos, pero con distintas configuraciones y valores de parámetros. La necesidad de gestionar la gran cantidad de datos generados sobre el entrenamiento ha hecho que se desarrollen herramientas especializadas en esto, como ML-Flow.

Desde los primeros algoritmos como el Perceptrón hasta las actuales redes neuronales se han utilizado herramientas matemáticas para representar las ideas en forma de algoritmos. Es común que los algoritmos utilizados tengan asociada una función de pérdida, de modo que la IA queda reducida a una función matemática que optimiza otra función, que ajusta sus parámetros internos para que, al recibir los datos de entrenamiento, la pérdida sea mínima. En el paradigma de *machine learning*, se encuentran algunos algoritmos que no utilizan

función de pérdida, o que no realizan una optimización para mejorar la métrica, como pueden ser KNN y RF. Cuando se dice que en los últimos años ha estado en auge la IA, es debido al otro paradigma del aprendizaje de máquina, *el deep learning*. Acá se utilizan arreglos de neuronas interconectados de una manera profunda, de modo que se realiza una gran cantidad de transformaciones a las características. Las redes neuronales se entrena y a la vez modifican a las transformaciones de los datos, lo que se conoce en la literatura como representación de características (Dong & Liu, 2020).

Al tener los resultados del entrenamiento se pueden analizar y seleccionar aquellos que mejor resuelvan el problema de interés. La herramienta de gestión de fue de una utilidad significativa en esta sección, puesto que los datos ya tenían cierto orden y estandarización, lo que facilitó agrupar todos los resultados y obtener información de ellos sobre qué tan bien se solucionó el problema. La información es asimilada por nuestro cerebro por medio de resúmenes o gráficas que simplifican la complejidad de los datos. Nuestra percepción tiene ciertos patrones que se han investigado y que dentro de lo que se denomina Story Telling, se tratan de usar en favor de la transmisión del mensaje lo más cercano a la intención de quien cuenta la historia. En este caso cuestiones como color, contraste y simplicidad, dan información al destinatario, pero intentando transmitir algo más allá de los números y proporciones de las visualizaciones (Knaflcic, 2015).

2.2.3. Literatura sobre RF, RF e IA y el problema atendido

Desde los inicios de las aplicaciones de la teoría electromagnética, la técnica ha permitido avances significativos y redes de comunicación inalámbrica cada vez más eficientes. Hoy en día existen infinidad de tecnologías enfocadas a la transmisión de datos por medio electromagnético. El paradigma hoy es

principalmente digital, de modo que los datos transmitidos con algún tipo de modulación están codificados por un conjunto de símbolos (usualmente ceros y unos). Es común ahora que: en las casas, lugares de estudio, hoteles, entre otros. existan redes WiFi que permitan a los usuarios conectarse a un punto de acceso hacia la Internet, un tipo de ciberespacio donde millones de personas interactúan por medio de transferencias de información. Una extrapolación de las redes WiFi es la red mesh, o malla (Funabiki, 2011).

En esta topología de red, cada uno de los dispositivos que la componen (MG, MR y MC) tiene la capacidad de transmitir su propia información y retransmitir la que obtiene de sus vecinos, de modo que existen múltiples caminos por donde los datos pueden viajar, aumentando con esto la robustez de la red ante fallas. Otra similitud de las redes *mesh* con las redes WiFi es que en última instancia existe un concentrador de la información (MG) que tiene las funciones de un *gateway* (puerta de entrada), en una red IP. (Benyamina, Hafid & Gendreau, 2011; Zeng *et al.* 2007).

Como se verá enseguida, existen varios trabajos que abordan el tema de optimizar una red de comunicaciones de manera teórica, también hay enfoques empíricos y otros en un espacio intermedio entre los dos enfoques. Para una red que ya ha sido desplegada en el terreno, existen ciertas condiciones deseables que implican que se brinda un buen servicio a los usuarios finales. Como se menciona en un libro sobre diseños de sistemas de aprendizaje de máquina, los aspectos técnicos y teóricos deben ser traducidos a palabras del negocio cuando se trata de soluciones que tienen que ver con problemas del mundo real y donde existen objetivos claros que pueden prescindir de toda la cuestión cuantitativa (Huyen, 2022).

En este caso, de cara al ámbito del negocio y de cara al usuario final, el problema que se atiende es acerca de si un determinado equipo comunicará o no al ser instalado en un lugar determinado y dado que el resto de los dispositivos de red se consideren inamovibles. En otras palabras, siendo que la red desplegada es la que es: ¿existirá una comunicación efectiva en determinada ubicación geográfica y bajo ciertas condiciones de instalación? En este trabajo, la respuesta a esta pregunta qué es dada por los algoritmos se refiere a la cobertura de la red.

2.2.3.1. Literatura sobre cobertura de RF

La cobertura o alcance de un determinado equipo de comunicación que utiliza radiofrecuencia para enviar y recibir información es un tipo de modelo del mundo. Como tal, se requieren ciertas simplificaciones del fenómeno real para adaptarlo a un marco teórico manejable y donde se pueda realizar el análisis deseado. Inicialmente se puede abordar el caso ideal del equipo en un entorno sin pérdidas más allá de la distancia y su relación con la propagación de las ondas electromagnéticas. Dado que la utilización de un equipo en un entorno realista implica una gran cantidad de obstáculos, al quitar una capa de simplificación sería útil hacer uso de información referente a la geografía circundante, dado que las ondas electromagnéticas se ven afectadas según el medio en el que se transmiten. Una herramienta comúnmente utilizada es el análisis de Fresnel, de modo que permite reducir la porción del espacio a aquella donde las ondas electromagnéticas deberían poder propagarse sin obstrucción (Simu & Vesa, 2020).

Aparte de la información sobre la geografía utilizada para determinar la obstrucción de Fresnel, cierta información sobre el tipo de terreno puede ser de ayuda para la determinación de la cobertura. Es sabido en el ámbito del negocio

que la vegetación, edificaciones y ciertos tipos de terreno influyen en mayor o en menor medida en la pérdida de la energía de las ondas propagadas.

Por supuesto se puede ir aumentando cada vez más la información referente al fenómeno del mundo real (propagación de ondas electromagnéticas en un determinado territorio). Aumentar la complejidad de la información agregada al modelo es también aumentar la dificultad en brindar una solución efectiva. Como la cobertura que se desea estudiar no corresponde a un solo equipo, sino a una red que está formada por miles de equipos, es posible un análisis que para cada uno de los equipos de la red se analice de manera individual su cobertura. Luego ha de ser necesario determinar cómo interactúan entre sí estas zonas de cobertura individual, para lo cual es necesario considerar la arquitectura o topología de la red. Para el caso específico de las redes malla, conocidas en la literatura como WMN (*Wireless Mesh Network*), existen una cantidad de factores que aumentan la complejidad para la determinación de la cobertura. Entre estos factores se destacan: la interferencia entre equipos de la misma red, la creación de cuellos de botella para el flujo de datos, las anomalías o fallas que ocurren en la red, que implican una reconfiguración de la topología cada vez que un equipo queda fuera. Los protocolos de comunicación utilizados por las redes malla y las topologías al momento del despliegue son bastante robustas y usan distintas técnicas para palear estas incidencias. El uso de distintos canales de frecuencia, la creación explícita de enlaces redundantes, entre otros, son técnicas que permiten un buen desempeño en entornos reales. (Benyamina, Hafid & Gendreau, 2011).

Pero dadas estas técnicas, la red desplegada y la geografía donde opera. ¿Cómo se determina la cobertura?

2.2.3.2. Avances en la solución del problema

Siendo la cuestión de determinar la cobertura en una red de comunicaciones un problema de interés en el ámbito de las redes, existen trabajos precedentes que atienden al problema en cuestión. Con el fin de categorizar las soluciones existentes se clasificarán estas en determinísticas y empíricas (Fernandes *et al.*, 2020; Krishen, 1994).

Los métodos determinísticos se basan en los modelos teóricos propios de la teoría electromagnética (Luebbers, 1984; Hufford, 1952).

Por su lado los empíricos tienen como base mediciones directas en la red de que se quiere determinar la cobertura (Popoola & Oseni, 2014; Hata, 1980).

Existen también métodos híbridos, que utilizan modelos matemáticos en conjunto con datos empíricos obtenidos en el terreno (Fernandes *et al.*, 2020).

Otro enfoque utilizado para la determinación de cobertura en una red implica realizar una simulación de la propagación de las ondas electromagnéticas, el establecimiento de radioenlaces y las pérdidas debido al entorno (Mohtashami & Shishegar, 2012).

Los modelos determinísticos parecen tener una mayor precisión en comparación con los empíricos, sin embargo, el precio a pagar es un mayor coste computacional.

Con el aumento de la capacidad computacional, así como los avances en el ámbito de la Inteligencia Artificial, específicamente en el aprendizaje de

máquina, se ha propiciado un nuevo enfoque para la solución de problemas de distinta índole (Agrawal, Gans & Goldfarb, 2017).

Podemos catalogar este enfoque a la solución de problemas como basado en datos, dado que los algoritmos de aprendizaje de máquina aprenden de los datos a solucionar los problemas. A diferencia de los enfoques anteriores, donde se puede apreciar una gran labor intelectual humana en plantear las ecuaciones y funciones a optimizar, con el aprendizaje de máquina y el aprendizaje profundo, el humano no interviene de manera explícita en la solución del problema. Lo referente al ámbito de las comunicaciones inalámbricas y su abordaje desde el aprendizaje de máquina, se ha denominado en la literatura como *Radio Frequency Machine Learning* (RFML) (Roy, Mukherjee & Chatterjee, 2019).

2.2.3.3. Literatura sobre RF e IA

El ámbito de las comunicaciones con radiofrecuencia incluye varios tópicos que van más allá de la determinación de la cobertura. Muchos de ellos, sin embargo, convergen a la mejora en la calidad del servicio brindado por una red de comunicaciones. El aprendizaje de máquina y el aprendizaje profundo se han utilizado para abordar muchos de estos temas, incluyendo por supuesto el problema de la cobertura de una red (Roy, Mukherjee & Chatterjee, 2019; El Hammouti, Ghogho & Zaidi, 2018).

Entre los problemas más atendidos por RFML, según se encuentran en la literatura incluyen la clasificación de emisores, detección de señales, encriptación de información, selección de canales óptimos, identificación de anomalías, entre otros (Chafii, Bader & Palicot, 2018; Karunaratne & Gacanin, 2019).

También la determinación de la cobertura, como se mencionó antes. Al utilizar el enfoque basado en datos y en aprendizaje de máquina, se elimina la necesidad de adquirir *software* de simulación y dependiendo de la disponibilidad de datos también es posible eliminar la necesidad de toma de datos en el terreno. Principalmente, según la literatura encontrada y consultada, el uso del aprendizaje de máquina y aprendizaje profundo para resolver el problema de la cobertura de una red se hace con redes con topologías distintas de la red malla, en especial la topología estrella, donde hay un equipo central y todos los clientes de la red deben conectarse a él, por lo que se puede decir que, en cierto sentido, se calcula la cobertura del equipo central de manera individual (Popoola, Misra, & Atayero, 2018; El Hammouti, Ghogho & Zaidi, 2018).

Atender el problema de la cobertura en una red malla, como ya se ha mencionado, tiene ciertos factores de complicación. En la literatura buscada y consultada no se pudo encontrar un trabajo que explícitamente atienda al problema de determinar la cobertura para una red con dicha topología, menos aún si la red ya ha sido desplegada. Existen trabajos enfocados en el diseño y despliegue de equipos en una red malla, pero sin utilizar el aprendizaje de máquina (Benyamina, Hafid & Gendreau, 2011).

Por otro lado, cuando se utiliza ML, se hace para determinar dónde instalar los MG y MR para optimizar la cobertura según la demanda del servicio (Karunaratne & Gacanin, 2019).

Pese a eso, existe la noción en la comunidad de las redes y la IA, de que existe un gran potencial para seguir buscando soluciones a problemas del ámbito con el uso del aprendizaje de máquina, con citas como la siguiente:

Comparado con los algoritmos clásicos basados en conocimiento experto, ML es atractivo porque produce *software* reutilizable y automatizable, usualmente más fácil de implementar y capaz de arrojar un mejor desempeño. Con eso y todo sigue siendo requerido el conocimiento experto para preparar los datos e interpretar la salida de las aplicaciones de ML. (Côté, 2018, p. 100)

La anterior traducción libre del contenido del artículo citado describe, a mi parecer, de manera significativa la solución desarrollada en este trabajo, que atiende explícitamente al problema de la cobertura para una red malla desplegada en campo.

2.3. Síntesis sobre la literatura

Ya lo dijo Borges (1964) “los artificios y el candor del hombre no tienen fin” (p. 530). Constantemente estamos creando nuevos artificios que pretenden simular la realidad, nuestros modelos del mundo. Y tenemos la ingenuidad de confundir a estos con la realidad que es extremadamente más compleja. Ejemplos de esto son los mitos, las ficciones y hasta la misma ciencia.

Estamos limitados por nuestra propia condición de seres mortales que es a su vez un motor que nos invita a buscar la trascendencia. Cuando creamos modelos de la realidad intentamos dar sentido al mundo con simplificaciones que sean abarcables por nuestro intelecto. Y resulta que estos intentos no han sido del todo infructíferos. El progreso actual lo hemos alcanzado porque bajo la sombra de ciertos mitos comunes pudimos colaborar con cada vez un mayor número de miembros de nuestra especie, hemos fantaseado con ficciones que

de una u otra forma nos han alcanzado, o que, mejor dicho, nosotros hemos alcanzado. Uno de los principales motores del desarrollo acelerado de la modernidad es la ciencia.

La evolución que ha tenido la idea de IA desde los mitos, en nuestra literatura y en la ciencia han creado el paradigma actual donde por medio del aprendizaje de máquina es posible buscar soluciones a problemas del mundo real. Ya no es necesario realizar un modelo complejo del mundo, los algoritmos de aprendizaje de máquina modelan los datos en un hiperespacio con tantas dimensiones que no podemos asimilar, y con la pura operatoria matemática consiguen resultados satisfactorios, en muchos casos superiores a los que los modelos creados por nuestros cerebros crean.

Es necesario tener cierta distancia en cuanto a los alcances de la IA y el aprendizaje de máquina actuales. Muchas veces parece que hemos sido superados por completo, y que estamos condenando a nuestra especie con la creación de una inteligencia por completo superior. No diré que no es una posibilidad. Pero también es necesario recordar que nuestra ciencia no es infalible, que la misma matemática con que se representan los algoritmos tiene sus limitaciones, y principalmente, que al ser la IA un producto de creación humana posiblemente herede nuestra propia imperfección.

Más allá de las implicaciones que tal vez pertenezcan más al futuro que al presente, el estudio de la ciencia de datos y del aprendizaje de máquina nos dotan de potentes herramientas para aplicar en la solución de problemas mucho más mundanos. En ese sentido, los avances actuales tanto a nivel general como en el problema específico de una red de comunicaciones y su cobertura permitieron encontrar soluciones satisfactorias a los dos problemas que se plantearon en este trabajo, y a un tercero que se consigue por añadidura. La

determinación del estado de comunicación de un equipo basado en sus registros históricos, la determinación del estado de un equipo nuevo dentro de la red. Y la posibilidad de terminar los estudios de grado de quien esto escribe.

2.3.1. Ideas principales extraídas de la literatura

- Las palabras son indispensables para nuestra civilización, y nos han permitido crear distintos tipos de IAs, divinas, fantásticas y reales.
- La IA es un concepto que inevitablemente abordamos desde una comparación con nosotros mismos. Es una búsqueda por imitarnos y acaso superarnos.
- Hablar de IA en nuestros días está principalmente relacionado con el aprendizaje de máquina.
- El uso del aprendizaje de máquina para la solución de problemas nos exime de pensar una solución explícita. Pero requiere de nosotros todavía bastante labor.
- Es necesaria la experimentación con los distintos algoritmos y técnicas que se conocen en la literatura para conseguir resultados satisfactorios.
- Ya sea en la religión, en la literatura o en la ciencia. De alguna manera nos contamos siempre las mismas historias.

3. INFRAESTRUCTURA COMPUTACIONAL

Pese a los diversos puntos de vista desde los que se hace uso del concepto inteligencia artificial, una constante casi inevitable es la de comparar esta con la inteligencia humana. Tiene sentido dado que este es nuestro más inmediato punto de referencia en cuanto a inteligencia.

En comparación, puede decirse que hoy en día la distancia entre la inteligencia biológica y la inteligencia artificial es remota. La inteligencia de la naturaleza procura su supervivencia y ha conseguido durante millones de años tal logro. En el sentido de que interactúa con su entorno, la inteligencia artificial tiene un medio ambiente ameno y una cantidad de labor humana propiciando su supervivencia y evolución.

A pesar de que la inteligencia artificial se encuentre en sus primeras etapas (augurando un futuro con una IA formidable, de grado considerablemente superior al actual), la IA es ya un sistema físico capaz de interactuar en algún grado con elementos provenientes del mundo real (datos), que persigue un objetivo, en este caso, devolver fuera del sistema una predicción de cobertura.

La parte de inteligencia del proceso es interactuar con las entradas y transformarlas de tal modo que la predicción coincida de manera aceptable con lo que llamamos realidad. Se dejarán de lado las implicaciones sobre IAs interactuando con el mundo, por lo que el *hardware* que compone el sistema en este trabajo será una computadora personal, un computador de placa reducida y servicios alojados en la Internet.

El sistema tiene una memoria, que almacena los datos en formato de origen dentro del *data lake*, estos provienen de distintas fuentes existentes en una red de comunicaciones con topología malla, se extrajeron con programas diseñados a la medida.

Los datos tal como se obtienen desde su origen poseen inconsistencias, datos erróneos, datos faltantes, ruido. Por tal motivo se transforman para propiciar un mejor desempeño de los algoritmos. En conjunto con esto se utilizaron métodos de ingeniería de características para propiciar la creación de nuevas características y la compatibilidad de estas con los algoritmos.

Los modelos hacen uso de los datos para buscar patrones de utilidad que permitan mejorar la métrica establecida en cada uno. El *software* se ejecutó en la computadora personal y en la nube. Los resultados de los distintos modelos de aprendizaje se almacenaron para su posterior análisis.

Se utilizan herramientas gráficas y datos tabulados para interpretar los resultados de los diversos análisis que involucró el proceso de desarrollo del sistema estimador de cobertura. Esta es en sí una salida del sistema, como la interfaz para la interacción humana.

Los datos que se utilizan para alimentar al sistema se consideraron estáticos, a pesar de que pertenecen a distintas instancias temporales, fue así porque las implicaciones de un sistema que evolucione con la red de comunicaciones a través del tiempo van más allá de los alcances del presente trabajo.

3.1. Diseño e implementación del mercado de datos (*data mart*)

Los datos de un sistema de comunicaciones desplegado en el terreno involucran una serie de procesos, parámetros y contexto que pueden propiciar datos útiles para ser entregados al sistema computacional encargado de convertir estos en información útil que permita tomar decisiones acerca de la red de comunicaciones.

Una red de comunicaciones en producción tiene distintos módulos que almacenan la información según las características propias de la aplicación y por lo tanto los formatos son heterogéneos. Para su uso en los algoritmos de aprendizaje de máquina y aprendizaje profundo, requerirán ser expurgados y normalizados a modo de interactuar de manera correcta con las ecuaciones que en última instancia representan a las ideas detrás de los algoritmos. Los datos acondicionados serán luego almacenados en el mercado de datos. Este proceso es lo que se denomina ETL, (extracción, transformación y carga según sus siglas en inglés). Un paso intermedio para llegar al mercado de datos es el lago de datos, donde se almacenará la información inmediatamente después de ser extraída de las fuentes originales, quedando a la espera de la transformación y carga final al DM.

3.1.1. Fuentes de datos

En el ámbito de la ciencia de datos, y principalmente en los ambientes empresariales, surgen conceptos referidos a la forma de almacenar datos para su disposición efectiva a la hora de ser utilizados para extraer información de ellos. Desde visualizaciones, resúmenes, hasta predicciones basándose en inteligencia artificial. Dentro de tales conceptos se encuentran los lagos de datos y los mercados de datos.

En su forma original los datos están adaptados a funciones distintas que las relacionadas con el análisis de los datos. La función principal de una red de comunicaciones es la transmisión de información. Es cierto que existen muchas herramientas que permiten la visualización y manejo de los datos, pero a su vez existen otras fuentes de datos enfocadas a aplicaciones funcionales, tales como los comandos enviados a los equipos, la configuración interna de los equipos desplegados en campo, las bases de datos transaccionales de las aplicaciones del sistema gestor, entre otras.

3.1.1.1. Base de datos del sistema

Una red de comunicaciones además de los equipos desplegados en el terreno requiere de sistemas computacionales de fondo que permitan la administración y concentración de la información transmitida por los distintos equipos. A diferencia de algunas redes como la red telefónica, los equipos de la red malla en cuestión también pueden comunicarse con otros equipos de la red, pero el flujo de información referente a las variables medidas por los dispositivos finales siempre se centraliza en los servidores dedicados para la recolección de información y que permiten su disponibilidad para ser aprovechados por los propietarios de la infraestructura de medición.

Estos equipos de cómputo cuentan con bases de datos donde almacenan una variedad de información referente a la red. La información de las variables medidas por los equipos finales sólo son de interés si pueden proporcionar información útil para determinar la comunicación efectiva o no. Dentro de los anteriores se pueden mencionar los datos esenciales de los dispositivos, tales como ubicación geográfica, poste, vecino, entre otras variables.

La base de datos del sistema a la que se tiene acceso está diseñada con un fin transaccional, para que registre la información enviada por los dispositivos finales, así como las distintas interacciones de los usuarios del sistema gestor. Al ser un sistema operando en tiempo real existen diversas transacciones que modifican constantemente los registros almacenados en las tablas.

Se desarrollaron una serie de consultas (*queries*) en SQL para convertir la información de distintas tablas en conjuntos de datos tabulados según los campos de interés. Los resultados de estas consultas no solo son útiles cuando se quiere almacenar información tabulada, sino que se utilizaron en el desarrollo de distintas funcionalidades en el sistema estimador de cobertura.

Se hace ahora la distinción entre los datos históricos y los datos instantáneos. Los primeros son aquellos que tienen un identificador único en la base de datos asociado a cada evento, por lo que es posible consultar la información en un tiempo futuro. Por su parte los datos instantáneos son aquellos que están relacionados a la información actual de un equipo, si las coordenadas del dispositivo varían, se pierden las coordenadas anteriores y al consultar la tabla correspondiente se obtienen las actuales.

Los datos históricos resultan de utilidad para agrupar los distintos conjuntos de datos y así tener una instantánea de la red en distintos instantes de tiempo. Esto es necesario por motivos que se abordarán en este y el siguiente capítulo, pero está relacionado con las limitantes de envío de comandos en el sistema y la no disponibilidad de todos los campos cuentan con registros en la base de datos disponibles.

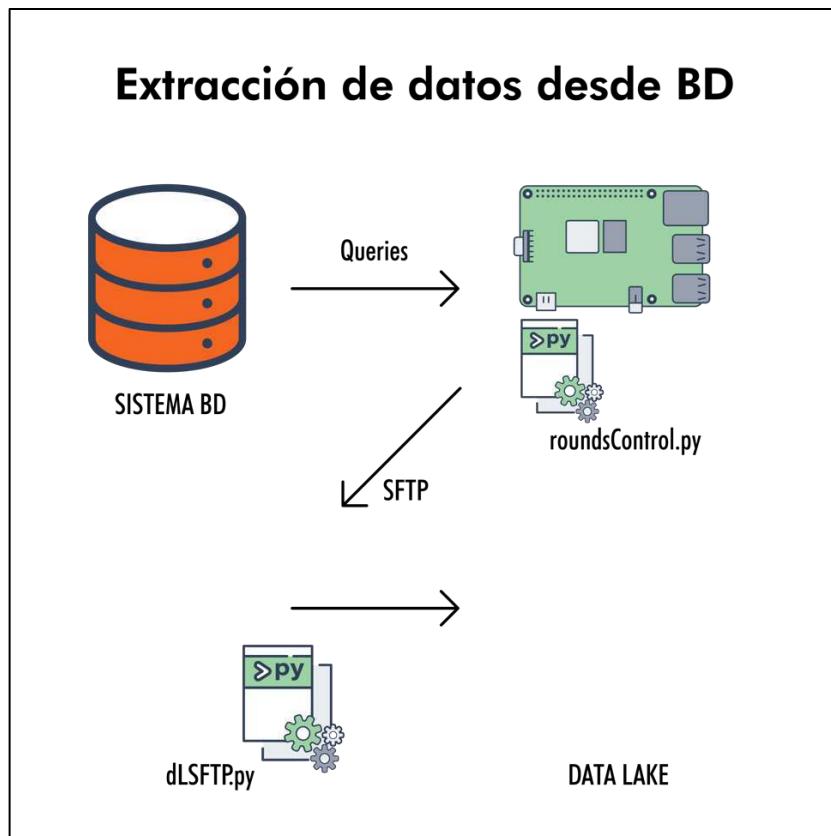
Los datos instantáneos varían constantemente, para conseguir el objetivo de obtener instantáneas de la red de comunicaciones en distintas instancias de

tiempo. Se programó un proceso diario en el computador de placa reducida (Raspberry Pi) para realizar una consulta a la base de datos y almacenar la información en una carpeta que puede ser consultada a través de SFTP para ser depositada en el lago de datos.

La siguiente figura muestra un esquema del flujo de datos del proceso referido.

Figura 2.

Extracción desde base de datos



Nota. La figura esquematiza la infraestructura para la extracción de datos desde la base de datos.
Elaboración propia, realizado con Adobe Illustrator.

Las consultas a la base de datos las realiza el script —roundsControl.py— , mismo que tiene la funcionalidad de programar el envío de comandos, de ahí el nombre. Se encontró una complicación a la hora de almacenar dicha información en formato csv, o en el formato nativo del paquete Pandas de Python, enfocado en el manejo de datos. La primera debido a una alteración y pérdida de información al momento de almacenarse y la segunda debido a la arquitectura de la Raspberry Pi. Se utilizó el método de Pandas to_records() para salvar la información sin alteraciones, por lo que los archivos se almacenaron en formato .json.

Los archivos almacenados en la Raspberry Pi tras el paso anterior contienen los datos instantáneos de interés de los distintos equipos, su ejecución es diaria y los archivos se almacenan con los siguientes nombres:

- Datos de Mesh Gateway: gateways-<fecha>.json
- Datos de Mesh Router: routers-<fecha>.json
- Datos de Mesh Client: clients-<fecha>.json

El directorio donde se almacenan los archivos se accede por medio del servicio SFTP configurado en el CPR. Un script de Python que se ejecuta de manera asíncrona permite igualar el contenido de ambos directorios haciendo uso del paquete paramiko y su método get(), que permite interactuar con servidores sftp. Se realiza una comparación de los nombres almacenados en el directorio del lago de datos, específicamente en la carpeta pointsData.

3.1.1.2. Comandos del sistema

La base de datos transaccional a la que se tiene acceso no cuenta con todos los parámetros de interés. Dentro de estos parámetros se encuentran

algunos tan importantes como los niveles de eficacia del envío de mensajes, la potencia a la que están configurados los equipos, entre otras. Estos parámetros existen y cada equipo almacena su propia información y el sistema gestor permite obtenerla con el envío de comandos que la requieren.

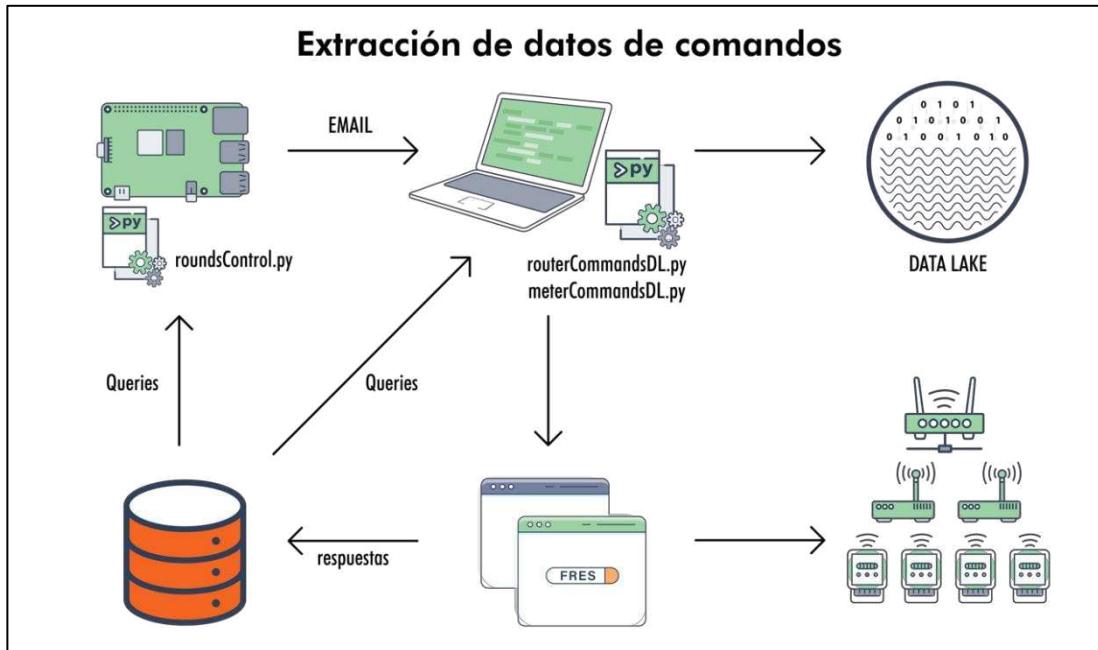
Los comandos son almacenados en la base de datos con su correspondiente respuesta en formato XML. Esto permite la visualización de la respuesta en la aplicación *web* del sistema gestor, pero los datos de respuesta no se almacenan en ninguna tabla a modo de campo, por lo que fue necesario desarrollar un proceso capaz de convertir la información de los comandos en datos tabulados.

Se llevó a cabo un proceso que inicialmente se ejecutó de manera manual, es decir, con archivos de Excel donde se seleccionaba un grupo de equipos para enviar los comandos correspondientes con la información deseada y luego extrayendo los registros de los comandos directamente desde la base de datos, esto con el fin de ejecutar una ronda completa a todo el parque de equipos desplegados en el terreno. Luego este proceso se automatizó para propiciar la adquisición periódica y automática, consiguiendo un aumento significativo en el tamaño del conjunto de datos disponible.

La siguiente figura muestra un esquema del flujo de datos del proceso referido.

Figura 3.

Extracción de datos de comandos



Nota. La figura esquematiza la infraestructura para la extracción de datos de comandos. Elaboración propia, realizado con Adobe Illustrator.

La periodicidad elegida fue de una semana, considerando la cantidad de equipos y de comandos enviados. Como interfaz entre el sistema y el usuario encargado del envío de los comandos con la función de envío masivo del sistema gestor de la red, se utilizó un correo electrónico. El mismo script que carga la información instantánea en el CPR (`roundsControl.py`) es el que envía los correos haciendo uso del paquete `yagmail`. Los lunes genera los conjuntos de equipos correspondientes a todos los días de la semana en curso según la cantidad de equipos existentes en la base de datos. Para los *mesh client* (la mayoría de los equipos) se envía un solo comando en una semana. Para los *mesh router* se envía tres veces el mismo comando en una semana, dado que son una cantidad bastante menor no considera una carga costosa al sistema.

El proceso inicia los lunes cuando se genera la calendarización de envío de comandos y se envía un correo indicando los equipos de la primera ronda. Las siguientes rondas se envían hasta el viernes. Los sábados se envían comandos a los equipos que no respondieron de manera efectiva a los comandos previos, esto con la finalidad de recabar información de equipos que pudieron sufrir una falla temporal en su envío calendarizado. La gestión del envío de comandos se hace manual según los valores separados por comas en cada uno de los correos. A su vez, el sistema gestor se encarga de iniciar el proceso de envío de comandos haciendo uso de la infraestructura del *backhaul* y la red *mesh*.

Todos los días se ejecutan los programas —`routerCommandsDL.py` y `clientCommandsDL.py`— en la computadora del sistema estimador de cobertura, los cuales consultan los registros de la base de datos del día inmediatamente anterior a su fecha de ejecución, con lo cual se obtienen las respuestas dadas por los equipos instalados en el terreno a los comandos enviados desde el sistema gestor de la red. Estas respuestas son almacenadas en formato parquet en la carpeta `commandsData` del lago de datos.

Los archivos almacenados tras el proceso referido fueron nombrados de la siguiente manera:

- Comandos de Mesh Router: `routers-<nombre comando>-<>.json`
- Comandos de Mesh Client: `clients-<fecha>.json`

3.1.1.3. Archivos

Los datos provenientes de entornos reales difícilmente se presentan en una forma óptima para ser utilizadas por los algoritmos que pretenden extraer información útil de ellos. Se dice muchas veces que, si un algoritmo recibe

basura, basura devuelve. Un paso aún necesario al momento de desarrollar este trabajo es el de la limpieza de datos y la imputación de datos vacíos.

En el marco del presente trabajo, se dispuso de una serie de archivos adicionales a los datos extraídos de las fuentes del sistema. Son archivos complementarios que permitieron hacer validaciones de variables de interés, así como completar campos vacíos debido a falta de registros en la BD o como forma de ajustar las primeras etapas en la extracción de datos al posterior proceso periódico con adquisición de una mayor cantidad de información.

La carga de los archivos complementarios en el lago de datos se dio de manera manual, tanto por lo inusual de los archivos como por la necesidad de cierto análisis que permita propiciar datos para la limpieza e imputación de datos, siendo más inmediato esto que programar un proceso automático.

Otra fuente de información que proporciona archivos son las respuestas de los comandos, la cual está en formato XML. Por lo tanto, los registros de comandos en la base de datos se dividirán en dos. Los datos tabulados que comprenden el contexto del comando en cuestión, y los string XML que contienen la respuesta explícita proveniente del equipo. Los primeros se almacenaron de manera tabulada, los segundos fueron tratados como archivos en un proceso de lectura y traducción del XML almacenado dentro de un archivo con extensión .parquet, para luego ser almacenados con un identificador que lo vincula con su correspondiente comando. A continuación, se ilustra la ubicación de los archivos en el lago de datos.

Figura 4.

Extracción de archivos



Nota. La figura esquematiza la infraestructura para la extracción de datos desde archivos de distinto tipo. Elaboración propia, realizado con Adobe Illustrator.

Los archivos complementarios fueron almacenados en la carpeta cFiles y los archivos XML constantemente son añadidos a la carpeta commandsData, según el proceso de extracción desde los comandos. El lago de datos, como se ha indicado, es almacenado en la computadora del sistema estimador.

Los nombres de los archivos complementarios no tienen un formato establecido, dado que algunos son únicos y otros específicos de las necesidades encontradas.

3.1.1.4. APIs externas

Una API, o interfaz de programación de aplicación, según sus siglas en inglés, es una forma de comunicación que permite a dos aplicaciones de *software* distintas requerir y compartir información entre sí. La necesidad cubierta haciendo uso de APIs en el presente trabajo está relacionada con obtener un contexto más amplio de los distintos equipos desplegados en campo. Este contexto se requiere en forma de dato para ser agregado al conjunto de datos de entrenamiento.

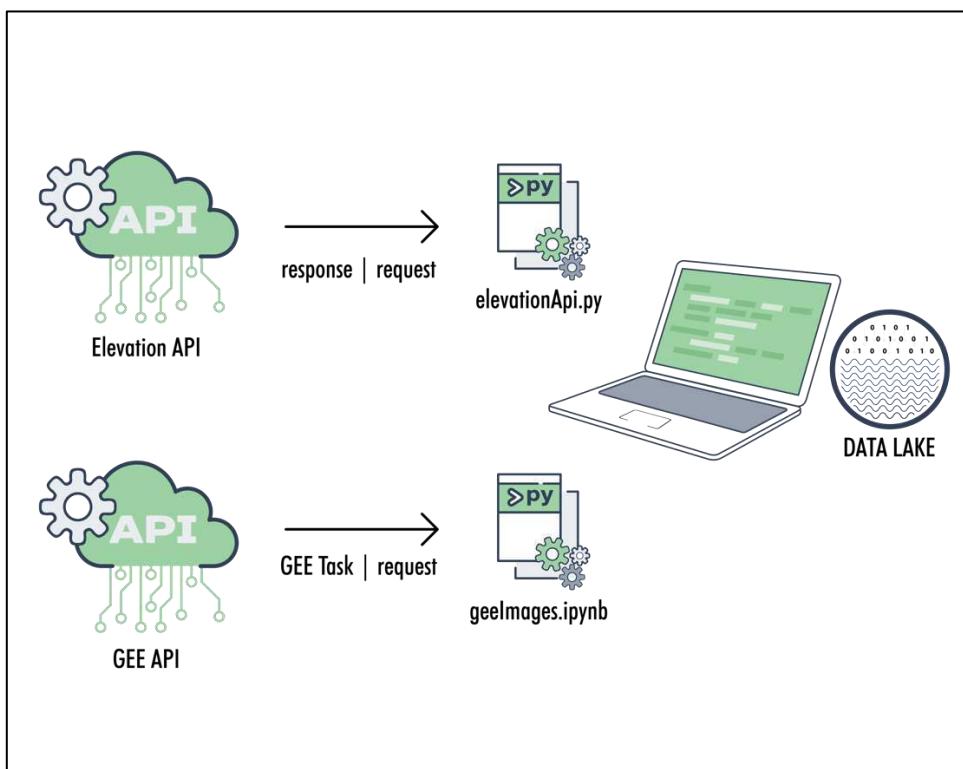
Esta información requerida es relacionada con la geografía del terreno donde están desplegados los equipos de la red. Específicamente fueron:

- Altimetría de los puntos, así como de la línea de radio enlace.
- Imagen satelital alrededor de los puntos de interés.

La empresa Google cuenta con APIs capaces de brindar la información que se requiere. La primera bajo una suscripción y la API —Elevation API— y la segunda desde el motor de Google Earth (Google Earth Engine). La siguiente imagen muestra este proceso de adquisición de datos:

Figura 5.

Extracción desde APIs externas



Nota. La figura esquematiza la infraestructura para la extracción de datos desde APIs. Elaboración propia, realizado con Adobe Illustrator.

La idea bajo la que se requieren estos datos se trata en el siguiente capítulo, de momento solo es importante el proceso de extracción de esta información, que se constituye en características generadas. La arquitectura computacional requerida para este objetivo es bastante sencilla, ya que las API's en cuestión son servicios alojados en internet a los cuales se puede acceder haciendo uso de una llave que se genera desde la interfaz de web del usuario de Google.

En el caso del motor de Google Earth, se requiere llenar una solicitud para tener acceso explicando los motivos detrás del uso de la información. Por cuestiones de simplicidad y procesamiento, la ejecución de geelimages.ipynb se hizo en la nube, realizado con Google Colaboratory, un servicio *web* que permite la ejecución de código de Python en formato de un Jupyter Notebook.

La información devuelta por la API de elevación es parte del código que permite la generación de la característica de Fresnel. Por su parte, la imagen en formato tif generada por la tarea en el motor de Google Earth, es almacenada en el drive de la cuenta de Google, estas imágenes serán procesadas posteriormente por un script que generará la característica de imagen satelital.

3.1.2. Carga en el mercado de datos

Las delimitaciones no suelen estar bien definidas a la hora de referirse a las fases que conlleva un proceso de ciencia de datos. La utilidad de estas es principalmente didáctica y como una forma de indicar de manera general las prácticas recomendadas para tratar los datos al momento de desarrollar una tarea con datos de manera eficiente. Si el proceso anterior fue el de extracción, al momento de cargar los datos al mercado de datos, ahora se estará procediendo con la transformación y carga de los datos, aptos para su posterior uso en el desarrollo de los modelos de predicción de cobertura, y completando de este modo el conocido proceso ETL.

Corresponderá a cada una de las distintas fuentes almacenadas en el lago de datos su adecuado tratamiento tanto de limpieza como de transformación, para luego ser depositadas en el mercado de datos. Las características generadas se presentarán en el siguiente inciso ya que su carga en el DM será posterior al proceso de generación de características.

Resaltando con esto que, en la práctica, las categorías del proceso de ciencia de datos se traslanan entre sí.

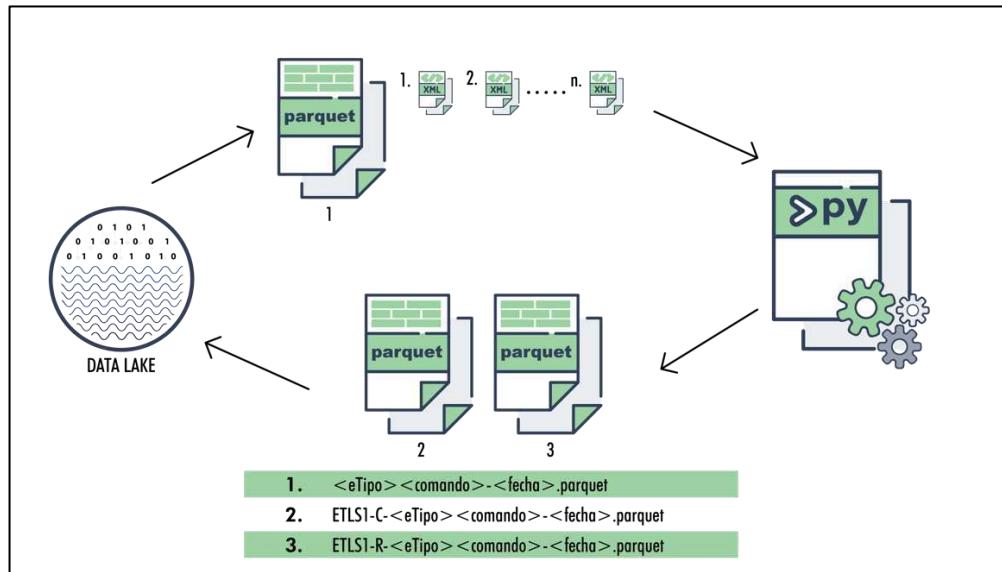
3.1.2.1. Lectura de archivos XML

Mientras que para el sistema gestor es sencilla la lectura de las respuestas a los diferentes comandos, dado que se puede obtener una visualización *web* de estas respuestas. Convertir estas en datos fue una tarea de considerables recursos temporales para el desarrollo de este proyecto, en un inicio fue necesario implementar una solución a la medida ante la no efectividad de las librerías encontradas para la lectura de estas respuestas en formato XML. Tal solución requirió de varias horas de procesamiento para la lectura de los miles de archivos XML correspondientes a los comandos enviados a los equipos de la red.

Se implementó un algoritmo que busca las distintas jerarquías existentes en un archivo XML y como resultado se obtiene una serie de diccionarios anidados de una manera estándar que permiten tener acceso a los datos particulares que corresponden a los nombres de las variables y sus valores correspondientes, estos son las filas de los datos tabulados de varios comandos del mismo tipo. En la siguiente imagen se ilustra el proceso descrito:

Figura 6.

Lectura de archivos XML



Nota. La figura esquematiza la lectura de archivos XML. Elaboración propia, realizado con Adobe Illustrator.

El nombre del archivo almacena cierta información necesaria para su traducción a información tabulada, así como la fecha que permitirá su posterior etiqueta y asociación con una correspondiente ronda de ejecución de comandos.

Según sea el tipo de comando almacenado en el lago de datos, el código de xml2df.py realizará la correspondiente traducción, dividiendo el registro almacenado en el archivo 1 en dos archivos (2,3), siendo el que tiene la letra C el que tiene la información del XML traducido a datos tabulados y R el resto de los campos relacionados con el envío de comando y que cuentan con una llave única que permite enlazar cada registro de comandos con su correspondiente línea en el archivo con la letra C. Las siglas ETL-S1 indican correspondencia a la primera etapa del proceso ETL.

ETL-S1 es una carpeta contenida en el lago de datos, que corresponde a lo que podríamos llamar un área de preparación (staging area, según los términos usados en el ámbito de la construcción de almacenes y mercados de datos), los datos en esta carpeta quedan a la espera de su correspondiente asignación de ronda, homologación y ciertos valores de interés antes de ser cargados al mercado de datos.

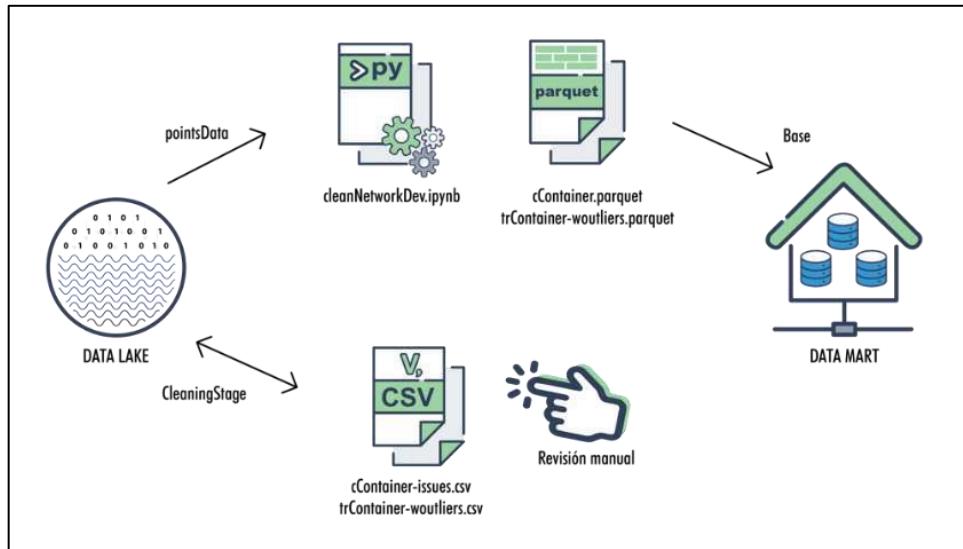
3.1.2.2. Limpieza de datos dispositivos de red

Los dispositivos de red en este contexto se refieren a los *Mesh Gateways* y *Mesh Routers*, estos son los principales componentes de la red ya que constituyen la infraestructura desplegada con el propósito de conseguir la correcta comunicación de los *Mesh Clients*. Estos equipos se considerarán en la mayoría de los casos estáticos, a diferencia de los *Mesh Clients* que pueden en cualquier momento ser retirados debido a solicitud del cliente.

La cantidad de los equipos de red es significativamente menor a la cantidad de clientes que conforman la red. En esto se basó la estrategia para la limpieza y validación de los datos. Para esta etapa de limpieza se procedió como se indica en la siguiente imagen.

Figura 7.

Limpieza de datos equipos de red



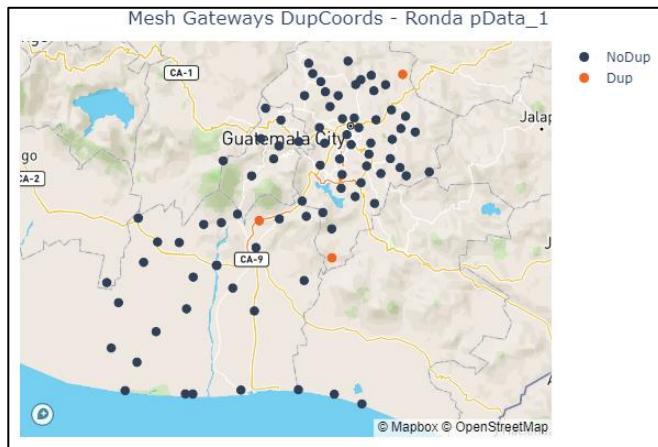
Nota. La figura esquematiza la limpieza de datos de equipos de red. Elaboración propia, realizado con Adobe Illustrator.

Para este proceso de limpieza se buscó encontrar inconsistencias en los datos, así como información de coordenadas incorrectas. Se utilizaron dos estrategias según cada caso.

Para los *Mesh Gateways* se utilizó funciones de la librería Pandas de Python para determinar datos faltantes, datos repetidos y cambios en los registros de los equipos. Luego se almacenó un archivo en formato csv para proceder con una revisión y corrección manual. Se utilizó el archivo revisado para actualizar el conjunto de datos original y actuar según cada caso de incidencia encontrada. Se utilizó la visualización en un mapa para explorar los puntos de interés. Se muestran a continuación capturas del mapa indicado.

Figura 8.

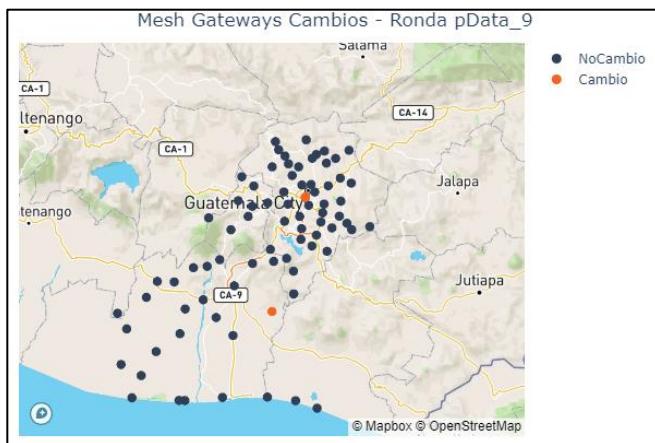
Mesh Gateways con información duplicada



Nota. La figura contiene el mapa de *Mesh Gateways* con información duplicada. Elaboración propia, realizado con Python.

Figura 9.

Mesh Gateways que presentan cambios en el tiempo

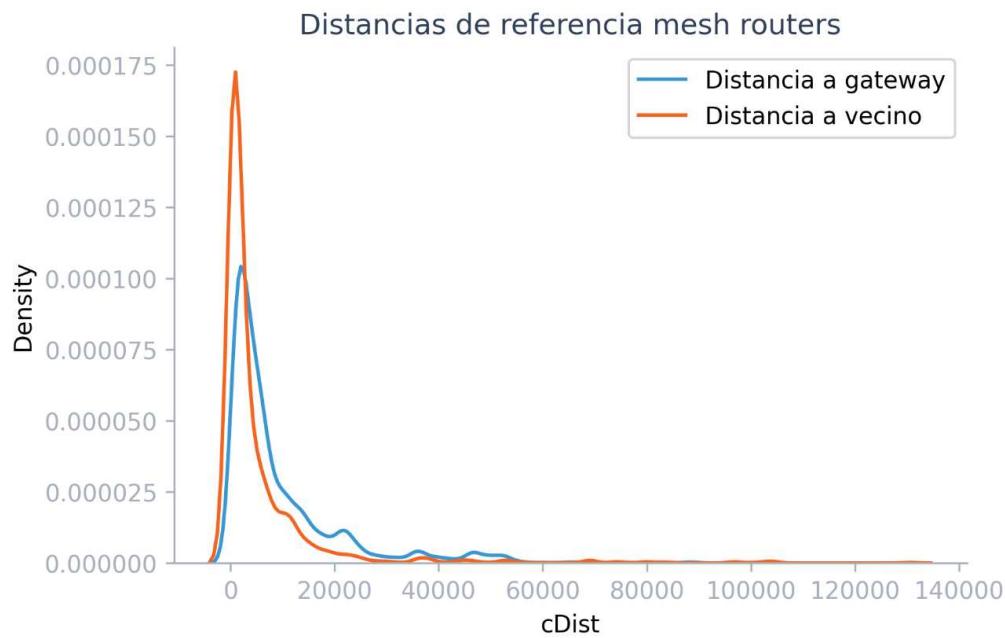


Nota. La figura contiene el mapa de *Mesh Gateways* con los cambios a través del tiempo. Elaboración propia, realizado con Python.

Mientras que en promedio existen 91 líneas por ronda en los registros de *Mesh Gateways*, la cantidad asciende a 752 en el caso de los *Mesh Routers*. Esta diferencia complica el análisis del conjunto de datos y la identificación de posibles valores erróneos. La revisión de la información geográfica se hizo por inspección visual en el caso de los MG. Para identificar posibles valores anómalos en los MR se recurre al cálculo de distancias de referencia de los distintos equipos. Las distancias de interés son entre su *Mesh Gateway* asociado y su mejor vecino. Las siguientes dos visualizaciones permiten abstraer la información contenida a través de los 84 días registrados, en relación con la identificación de posibles datos erróneos.

Figura 10.

Distancias de referencia Mesh Routers



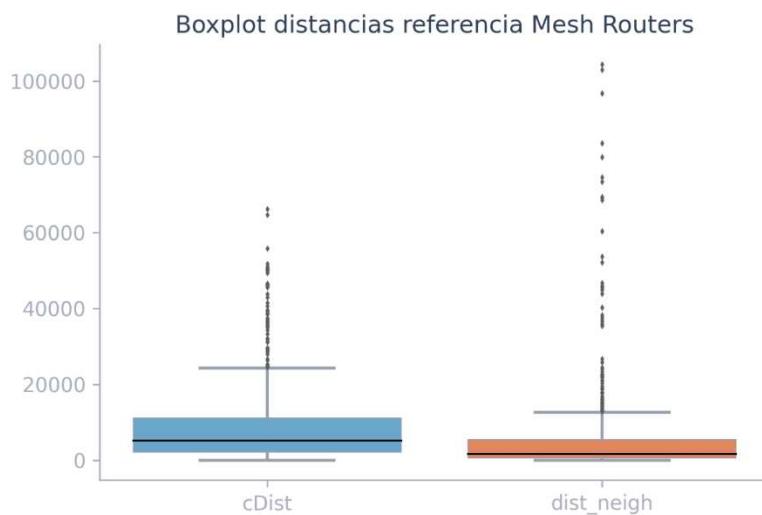
Nota. El gráfico contiene la distribución de las distancias de referencia de los *Mesh Routers*. Elaboración propia, realizado con Python.

El gráfico anterior muestra la distribución de las distancias (en kilómetros) calculadas con la fórmula de Haversine entre el MR en cuestión y su correspondiente referencia. Se aprecia una mayor presencia de grandes distancias respecto del MG que del mejor vecino, estimando por simple inspección el promedio remarca esta apreciación. Se ve que en ambos casos existen valores elevados de distancia que se alejan bastante de la tendencia general.

Determinar el umbral en metros que represente una situación anómala puede ser abordado desde varios enfoques. Para este caso se utilizará un enfoque basado en la estadística. La siguiente gráfica muestra la identificación de valores atípicos (puntos) para distancias fuera de 1.5 veces el rango intercuantil.

Figura 11.

Boxplot distancias referencia Mesh Routers

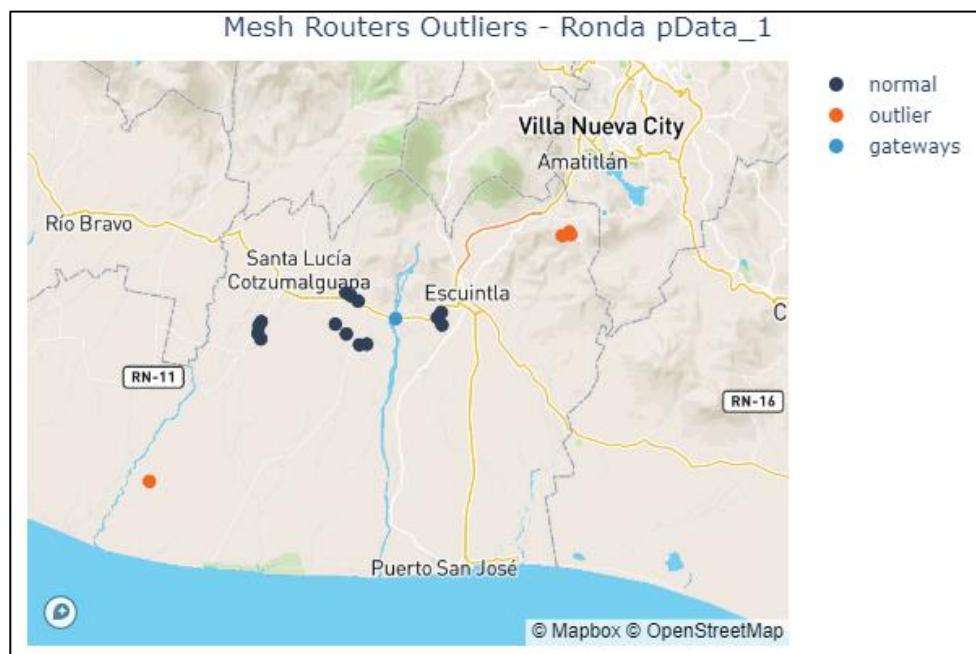


Nota. El gráfico contiene el boxplot de las distancias de referencia de los *Mesh Routers*. Elaboración propia, realizado con Python.

En la imagen anterior se muestra la distancia en kilómetros. La caja coloreada de la imagen comprende el rango intercuantil, en el cual se enmarca el 50 por ciento de los datos, las líneas (bigotes) contienen el 75 por ciento de los datos. De tal modo, la revisión manual que se ejecuta es sobre el 25 por ciento de los datos más alejados, o, las distancias que podrían ser tan grandes debido a errores en los datos. Se procedió a generar un archivo de revisión e inspeccionarlos de manera manual, indicando correcciones que luego fueron ejecutadas en el cuaderno cleanNetworkDev.ipynb. Como una forma de facilitar la revisión manual se crearon las siguientes visualizaciones para explorar los distintos casos de valores atípicos (outliers) detectados.

Figura 12.

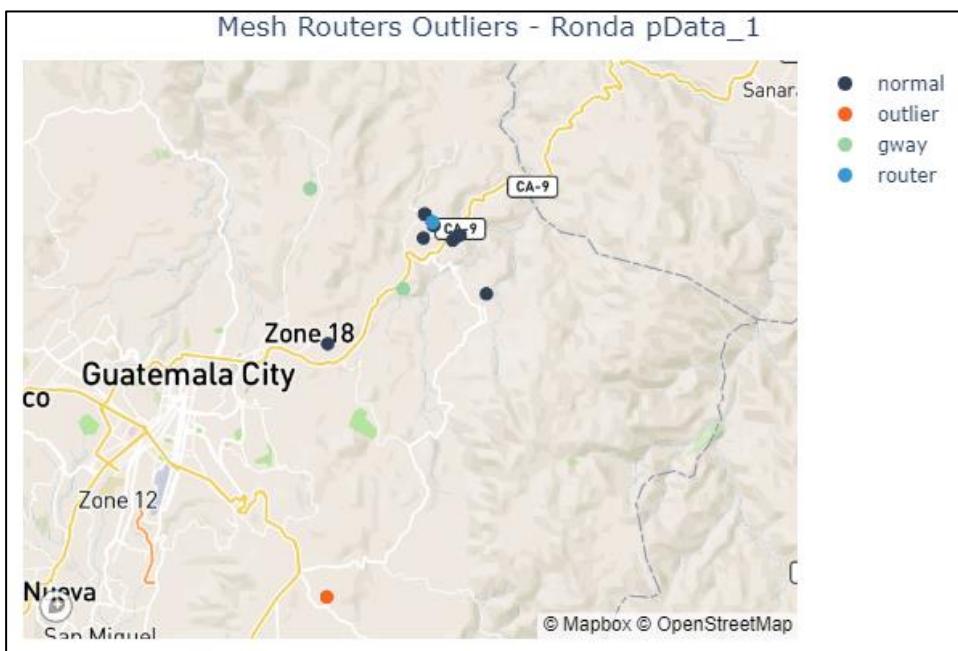
Visualización de outliers por agrupación de MG



Nota. La figura contiene el mapa con los valores atípicos en los *Mesh Routers*. Elaboración propia, realizado con Python.

Figura 13.

Visualización de outliers por agrupación de MR

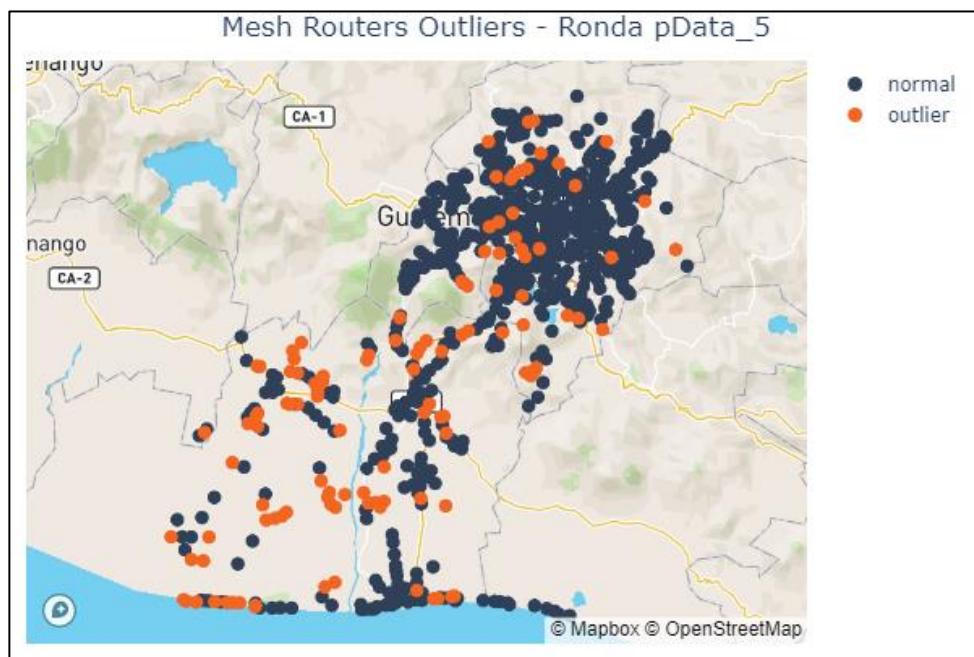


Nota. La figura contiene el mapa con los valores atípicos por agrupación en los *Mesh Routers*.

Elaboración propia, realizado con Python.

Figura 14.

Visualización de totalidad de MR con outliers



Nota. La figura contiene el mapa con los *Mesh Routers* y su indicativo de valor atípico. Elaboración propia, realizado con Python.

Los valores atípicos se observan en las periferias y sectores más aislados de la red. Se utilizó ipywidget de Python para conseguir visualizaciones interactivas en el cuaderno y poder hacer la revisión según las segmentaciones de interés en el mapa. Luego de la revisión manual se consiguió clasificar los distintos outliers, la corrección correspondiente se aplicó y en los casos relacionados con coordenadas erróneas de vecinos de tipo MC, se conservó la etiqueta para después de la limpieza de los *mesh clients*.

Tanto en el caso de los *mesh clients* como en el de los *mesh routers*, luego de la revisión e imputación de medidas correspondientes con esta, se procedió a almacenar la información en la carpeta Base del Mercado de Datos (*data mart*).

3.1.2.3. Limpieza de datos *Mesh Clients*

La relación entre la cantidad de registros de MG y MR es de aproximadamente 1:8, mientras que la relación entre los MG y los MC es de aproximadamente 1:232. Siendo que cada uno de los equipos tiene un registro para los 84 días de la muestra trabajada, la cantidad de filas en el conjunto de datos de los MC supera los 1.7 millones de registros.

Tal situación hace que el enfoque manual con que se limpiaron los MG y el enfoque estadístico con que se limpiaron los MR sea considerablemente más complicado en el caso de los MC. El enfoque utilizado en el caso de la limpieza de este conjunto de datos fue el aprendizaje de máquina (*machine learning*). Al permitir que sea un algoritmo lo que determine la anomalía o no de un MC, se reduce considerablemente la cantidad de análisis mental y de trabajo humanos.

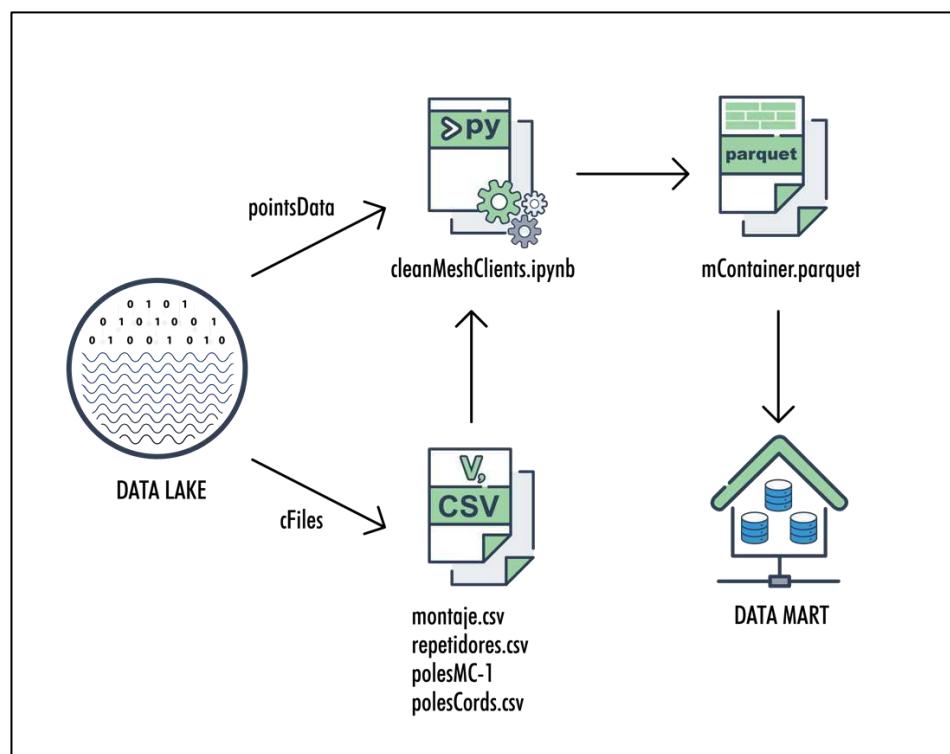
Como el conjunto de datos no cuenta con etiquetas que indiquen anomalía, fue necesario utilizar un enfoque de aprendizaje no supervisado. La elección del algoritmo fue Isolation Forest, por sus bondades a la hora de abordar el problema de detección de anomalías en un conjunto de datos.

El algoritmo Isolation Forest, como su nombre lo indica está enfocado en aislar ciertos registros del conjunto de datos, los cuales se espera sean anómalos. A diferencia de otros enfoques en la detección de anomalías, IF no mapea la normalidad en primera instancia para luego determinar los elementos que no cazan con esa caracterización, sino que, haciendo uso de un bosque de

árboles de decisión, determina como anómalos los elementos del conjunto de datos que requieren una menor cantidad de pasos para ser aislados por completo. El resultado luego del entrenamiento es una generalización de todo el bosque.

Figura 15.

Limpieza de datos Mesh Clients



Nota. La figura esquematiza la limpieza de datos de *Mesh Clients*. Elaboración propia, realizado con Adobe Illustrator.

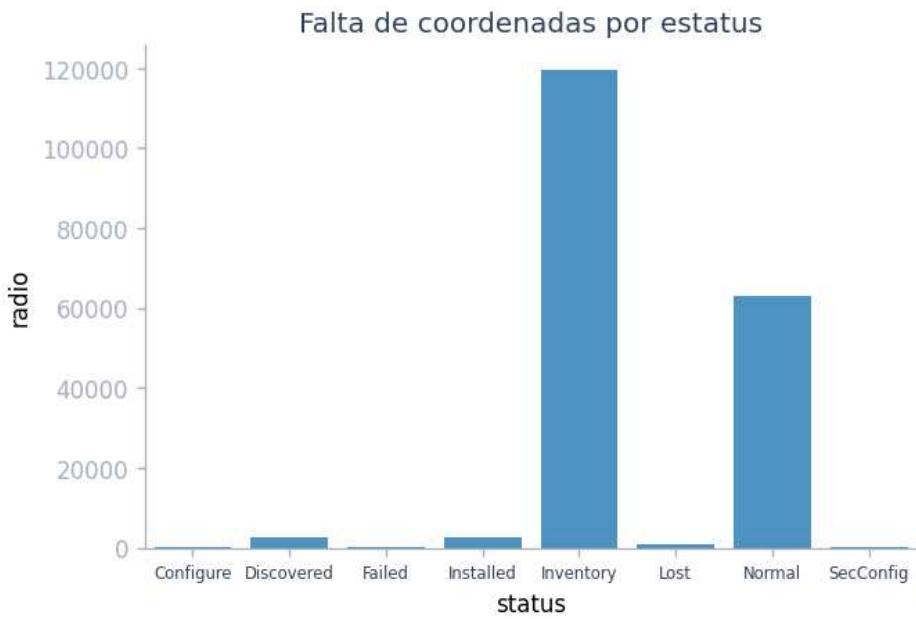
Como se mencionó antes, los MC tienen una mayor propensión a ser removidos del terreno e incluso ser instalados en distintas ubicaciones, pueden estar en almacén y ser instalados dentro del tiempo contemplado en los 84 días de muestra. También pueden cumplir con la función de ser repetidores o de estar

directamente instalados en un polo. Para las situaciones anteriores y algunas otras, fue necesario utilizar información proveniente de fuentes externas y administrativas que permitan determinar su estado de montaje, así como coordenadas de referencia respecto del polo instalado.

Una forma inicial de abordar el problema fue determinando la cantidad de filas en el conjunto de datos resultante de concatenar los archivos en pointsData del Data Lake. La visualización de estos vacíos con respecto a su estado en el sistema gestor de la red dio una primera apreciación de la situación de los MC con falta de coordenadas.

Figura 16.

MC sin coordenadas inicial



Nota. El gráfico muestra la distribución de *Mesh Clients* sin coordenadas según el estatus. Elaboración propia, realizado con Python.

Según la gráfica el estatus con mayor cantidad de vacíos es inventory, o inventario. Lo cual tiene una justificación en que los dispositivos no desplegados en el terreno evidentemente no tendrán coordenadas en el sistema gestor de la red. Por este motivo se utilizó la información de montaje para determinar si un equipo está desplegado en el terreno. Luego de una serie de tratamientos de los datos la gráfica de los restantes queda de la siguiente manera.

La considerable disminución en el total de incidentes, así como la mayor proporción en estados distinto de inventario es un comportamiento esperado luego de realizar el cotejo de los datos contra la información de montaje. Los MC eliminados en este proceso están sin ser desplegados y por lo tanto no son motivo de interés.

Además del análisis de montaje para los MC sin coordenadas, se evaluó cuáles de ellos tenían coordenadas en el conjunto de datos con coordenadas. Es decir, cuáles de los MC que tienen registros en algunas fechas sin coordenadas, tienen coordenadas en fechas diferentes. También se determinó cuáles MC eran aptos para que las coordenadas asignadas fueran las del polo de referencia. Las etiquetas utilizadas para estos dos grupos fueron: Coordenadas polo y Coordenadas parciales. Existe un tercer grupo que no ingresa en ninguna de las siguientes categorías y su distribución por estatus se ve como sigue.

Figura 17.

MC sin categorizar

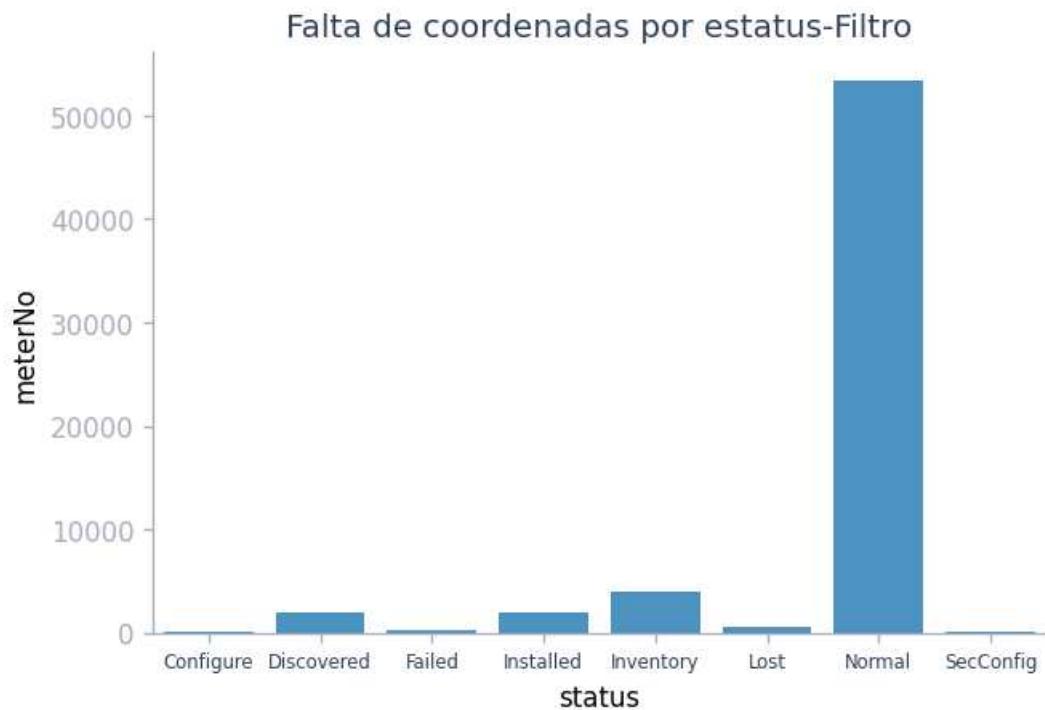


Nota. El gráfico muestra la distribución de *Mesh Clients* sin coordenadas según el estatus, para los registros restantes. Elaboración propia, realizado con Python.

Debido a que existe una diferencia entre la ejecución de una instalación o retiro de equipo en el terreno con respecto a su registro en el correspondiente sistema administrativo, fue necesario integrar una fecha posterior a los 84 días de muestra. De igual forma, un registro de MC en el mismo formato que el Data Lake, siempre de una fecha futura permitió la adjudicación de coordenadas a MC que al momento de recabar la muestra no tenían tal información en el sistema gestor de la red. Como resultado de esto se pasó de 371 MC sin coordenadas a 212, evidentemente los registros pueden ser incluso de 84 por cada MC, por lo que las filas sin coordenadas quedaron como sigue.

Figura 18.

MC sin coordenadas luego de incluir registros futuros



Nota. El gráfico muestra la distribución de *Mesh Clients* sin coordenadas al incluir los registros futuros, según el estatus. Elaboración propia, realizado con Python.

Por último, se procedió a utilizar los grupos mencionados con anterioridad, y cuyas etiquetas fueron: Repetidor y PoloMC-1. Esto es por motivos circunstanciales que implican que estos dos grupos no tienen un marco sistematizado de actualización de su información hacia el sistema gestor de la red, por tal motivo se utiliza la fuente externa disponible con la información real. Esta operación redujo la cantidad de MC sin coordenadas de 212 a 115. Se muestra el resultado final en el conjunto de datos de MC sin coordenadas.

Figura 19.

MC sin coordenadas luego de incluir fuentes externas

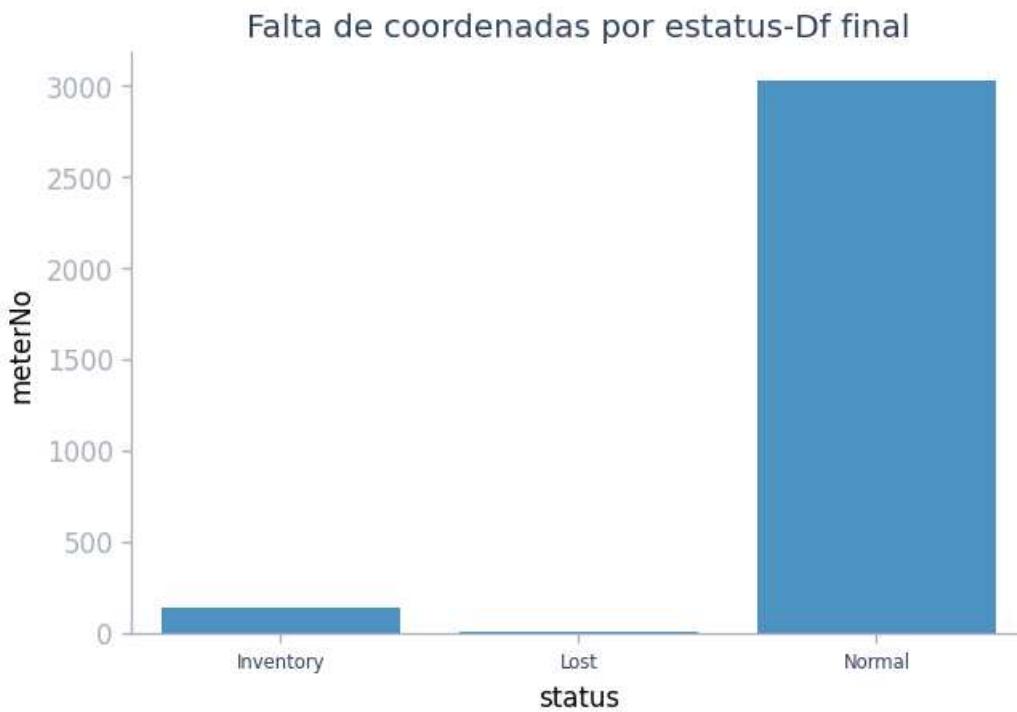


Nota. El gráfico muestra la distribución de *Mesh Clients* sin coordenadas al incluir las fuentes externas, según el estatus. Elaboración propia, realizado con Python.

El siguiente paso fue actualizar el conjunto principal de datos mContainer, según los cambios realizados a mCnCords. Luego fue necesario aplicar el proceso de asignación de montaje a todo el conjunto principal de datos, cuya cantidad de filas es de 1733505. De igual forma, se realizó la asignación de la referencia a los Repetidores y PoloMC-1, en el conjunto principal. Como resultado se tienen las siguientes filas sin coordenadas para MC que están desplegados en el terreno.

Figura 20.

MC sin coordenadas en conjunto de datos principal, final



Nota. El gráfico muestra la distribución de *Mesh Clients* en el conjunto de datos final según el estatus. Elaboración propia, realizado con Python.

Esto termina el tratamiento efectuado con aquellos MC que pese a estar operando en el terreno no tienen una asignación de coordenadas. Dada la importancia que se ha dado en este proyecto a la ubicación geográfica, se vuelve necesario realizar una detección de posibles valores erróneos, para lo cual se utilizará el aprendizaje de máquina.

A continuación, se presenta un análisis somero de las iteraciones más resaltables que se efectuaron durante el entrenamiento del modelo IForest, en el cuaderno de Jupyter de limpieza de MC solo se presenta el modelo elegido luego

de todas las iteraciones. Sin embargo, se llegó a este precisamente por las validaciones elegidas con base únicamente en la experiencia empírica de ubicaciones y distancias.

Como primera instancia fue necesario tratar algunas características para ser procesadas de mejor manera por el algoritmo, así como para mejorar su efectividad según las revisiones respectivas. CollectorFreq fue creada para representar numéricamente a los MG, según su frecuencia en la totalidad de los datos. También se creó la característica Gw_dist, para representar la distancia sobre la superficie de la tierra entre cada MC y su correspondiente MG.

Tabla 3.

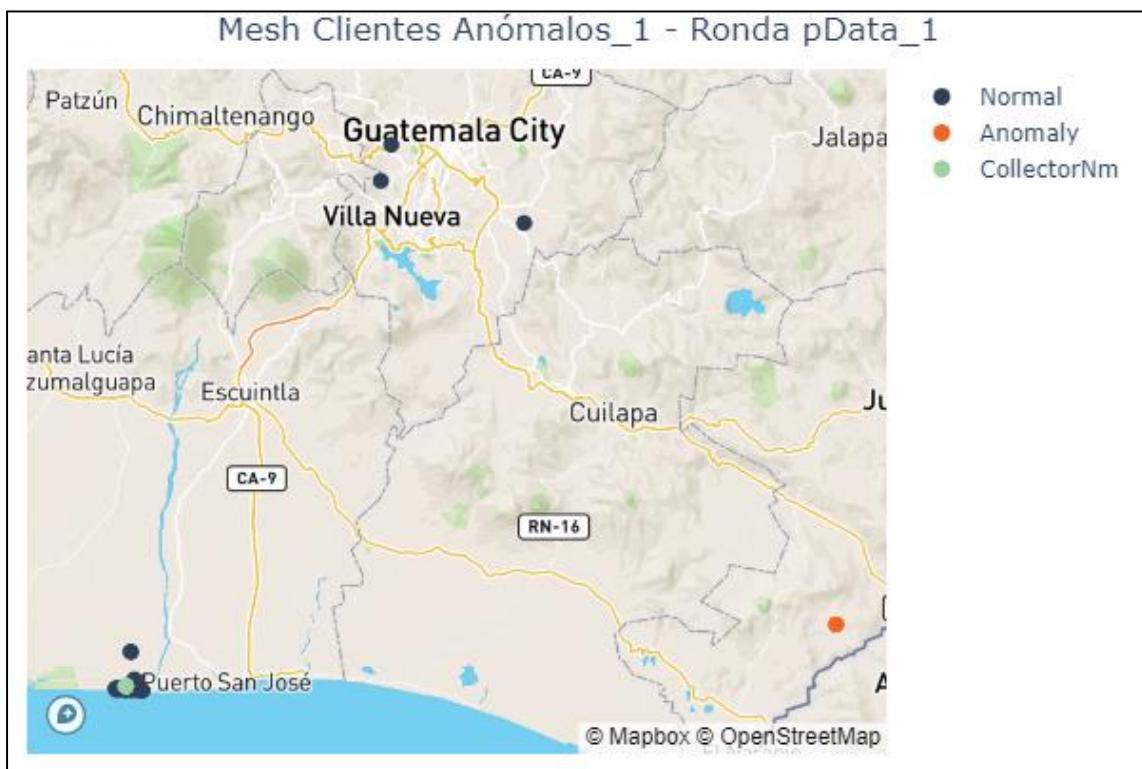
Modelo IF_1

Elemento	Código
Características para entrenar modelo	CollectorFreq, latitude, longitude
Parámetros del modelo	n_estimators=100, contamination=float(0.01), random_state=1010 max_samples='auto',

Nota. La tabla muestra la configuración utilizada en el modelo IF_1. Elaboración propia, realizado con Word.

Figura 21.

Anomalías detectadas por IF_1



Nota. La figura contiene el mapa con las anomalías detectadas por IF_1. Elaboración propia, realizado con Python.

La visualización a la que se acudió fue la de agrupar por MG. Se puede ver en la imagen que el punto a la derecha es detectado como anomalía, lo cual resulta adecuado dado que al revisar los datos se detecta que esta es una anomalía que persiste a lo largo de varios MG, lo que implica varios MC con esas coordenadas completamente aisladas. La falla acá son los puntos que evidentemente están bastante lejanos pero que en el conjunto general de datos están en un área de gran densidad de MC. Por tal motivo se decide agregar la referencia geográfica del MG al conjunto de datos de entrenamiento.

Tabla 4.

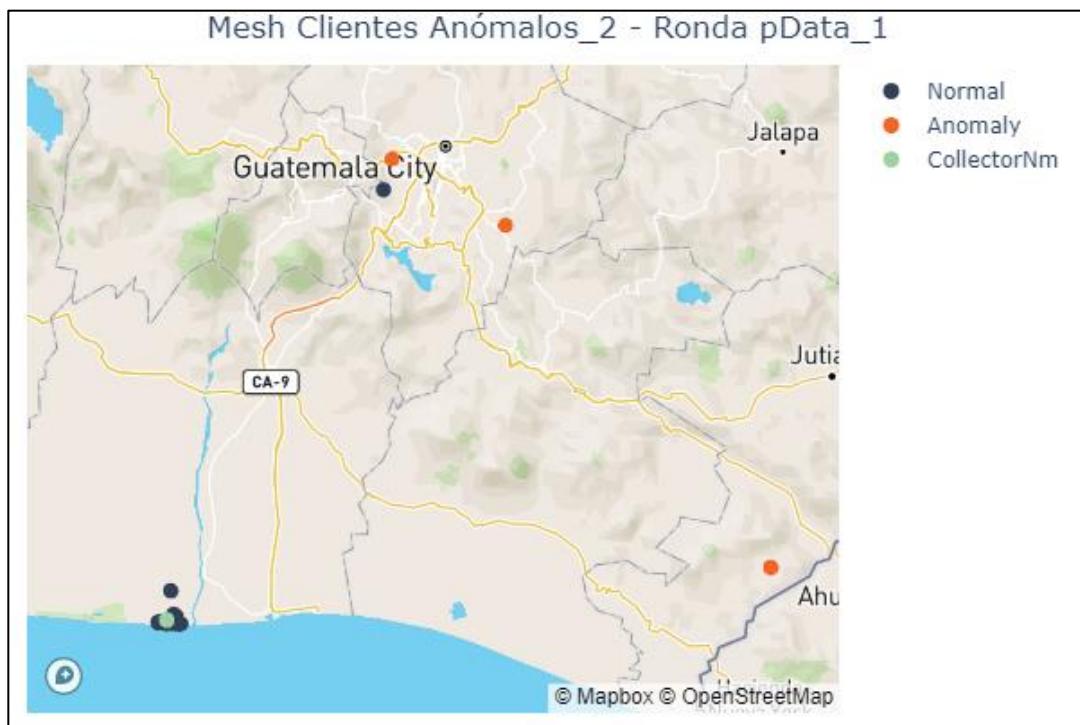
Modelo IF_2

Elemento	Código
Características para entrenar modelo	CollectorFreq, latitude_gw, longitude_gw, latitude, longitude
Parámetros del modelo	n_estimators=100, max_samples='auto', contamination=float(0.01), random_state=1010

Nota. La tabla muestra la configuración utilizada en el modelo IF_2. Elaboración propia, realizado con Word.

Figura 22.

Anomalías detectadas por IF_2.1

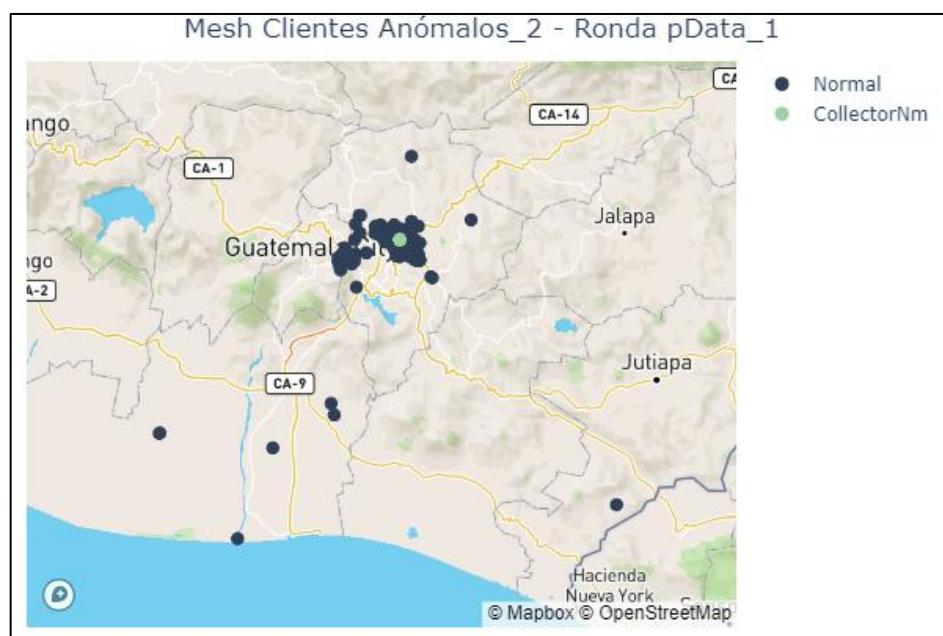


Nota. La figura contiene el mapa con las anomalías detectadas por IF_2.1. Elaboración propia, realizado con Python.

La anterior gráfica muestra que agregando la referencia de donde está ubicado cada MG mejora la detección de anomalías. Esto tiene sentido desde que la lógica que se busca es detectar anomalías respecto de su MG. Sin embargo, parece que hay puntos que a pesar de estar lejos no son detectados como anomalía. Un mejor ejemplo de esta situación es la siguiente gráfica.

Figura 23.

Anomalías detectadas por IF_2.2



Nota. La figura contiene el mapa con las anomalías detectadas por IF_2.2. Elaboración propia, realizado con Python.

Se agrega ahora la característica calculada Gw_dist, haciendo aún más explícita la intención de detectar anomalías en la ubicación de MC con respecto a sus correspondientes GW. Esto es lo que se trabajó con el modelo IF_3. También se determinó la necesidad de aumentar el valor del parámetro contamination en una centésima.

Tabla 5.

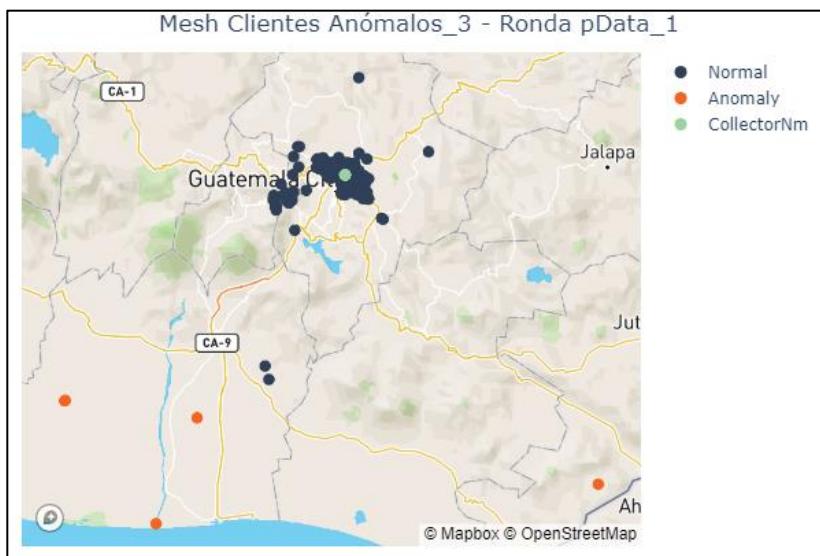
Modelo IF_3

Elemento	Código
Características para entrenar modelo	CollectorFreq, latitude_gw, longitude_gw, latitude, longitude, Gw_dist
Parámetros del modelo	n_estimators=100,max_samples='auto', contamination=float(0.02), random_state=1010

Nota. La tabla muestra la configuración utilizada en el modelo IF_3. Elaboración propia, realizado con Word.

Figura 24.

Anomalías detectadas por IF_3.1



Nota. La figura contiene el mapa con las anomalías detectadas por IF_3.1. Elaboración propia, realizado con Python.

Se aprecia que para el último caso indicado en el modelo IF_2, las anomalías por su lejanía aumentaron según lo esperado. Las siguientes dos imágenes muestran otras anomalías identificadas que coinciden con el análisis humano particular de quien realiza este trabajo.

Figura 25.

Anomalías detectadas por IF_3.2

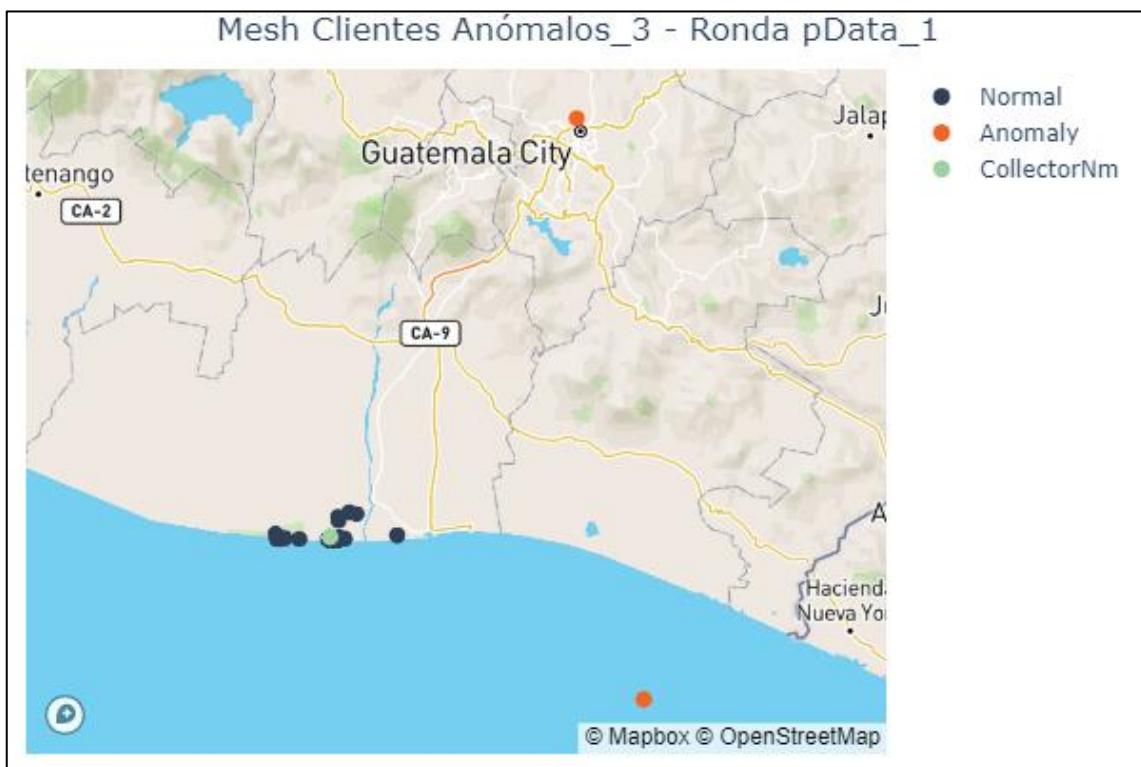


Nota. La figura contiene el mapa con las anomalías detectadas por IF_3.2. Elaboración propia, realizado con Python.

Más allá de la intuición basada en la geometría, es evidente que la siguiente coordenada en medio del mar es una anomalía. Misma que no había sido advertida por las anteriores iteraciones del modelo.

Figura 26.

Anomalías detectadas por IF_3.3



Nota. La figura contiene el mapa con las anomalías detectadas por IF_3.3. Elaboración propia, realizado con Python.

En la figura *Anomalías detectadas por IF_3.1* se aprecian unos puntos que a pesar de la distancia persisten en no ser detectados, luego de algunas pruebas se determinó una mejora al remover la referencia geográfica del MC. Sin embargo, al conservar Gw_dist, se conservan los resultados que en su momento significaron las variables removidas.

Tabla 6.

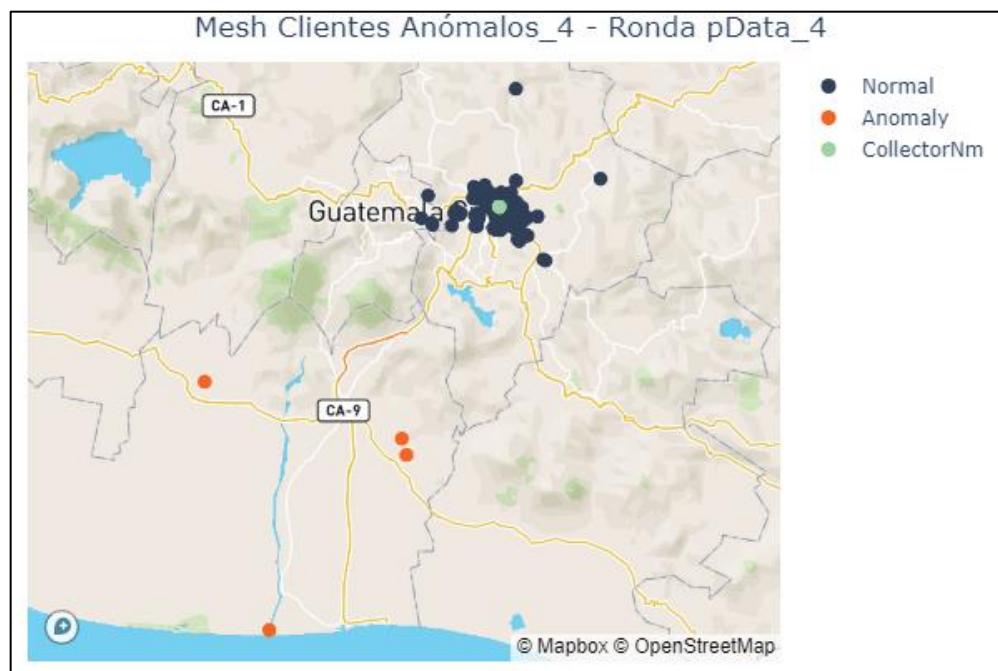
Modelo IF_4

Elemento	Código
Características para entrenar modelo	CollectorFreq, latitude, longitude, Gw_dist
Parámetros del modelo	n_estimators=100, max_samples='auto', contamination=float(0.02), random_state=1010

Nota. La tabla muestra la configuración utilizada en el modelo IF_4. Elaboración propia, realizado con Word.

Figura 27.

Anomalías detectadas por IF_4.1



Nota. La figura contiene el mapa con las anomalías detectadas por IF_4.1. Elaboración propia, realizado con Python.

Los puntos mencionados antes han sido detectados, por lo que para propósitos de lo que se espera en esta etapa de limpieza se puede dar por concluida la detección de anomalías. No sin antes mencionar ciertas particularidades relacionadas con las anomalías identificadas.

Figura 28.

Anomalías detectadas por IF_4.2



Nota. La figura contiene el mapa con las anomalías detectadas por IF_4.2. Elaboración propia, realizado con Python.

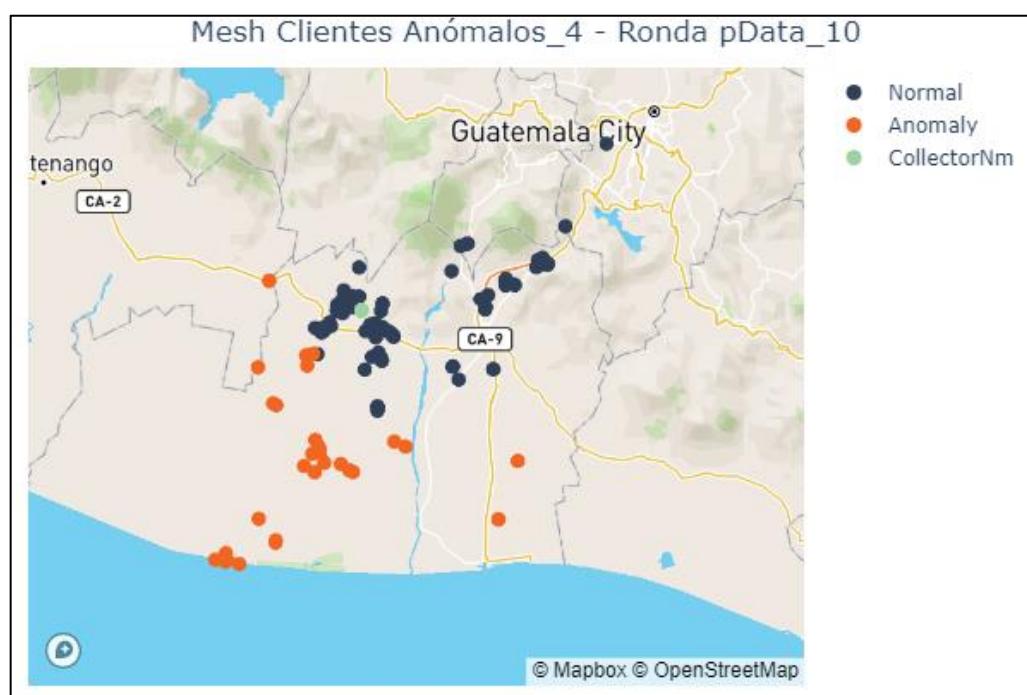
En la imagen se ve el caso recurrente de anomalía geográfica, sin embargo desde la experiencia empírica también parece existir una anomalía no identificada. Esto solo recalca que al utilizar aprendizaje de máquina suceden cosas de las cuales difícilmente se percatará un humano.

Esto puede incluso invalidar los análisis realizados anteriormente, mismos que se indicaron porque efectivamente la idea de relacionar las variables en el modelo fue una decisión completamente de diseño, justificadas en lo que ya se comentó, arrojando resultados similares a lo esperado. Sin embargo, el retiro de la referencia geográfica del MG luego de haber sido agregadas en una etapa anterior, fue una acción tomada sin justificación.

De modo que tampoco existirá una explicación para las incidencias que se describen en lo sucesivo.

Figura 29.

Anomalías detectadas por IF_4.3

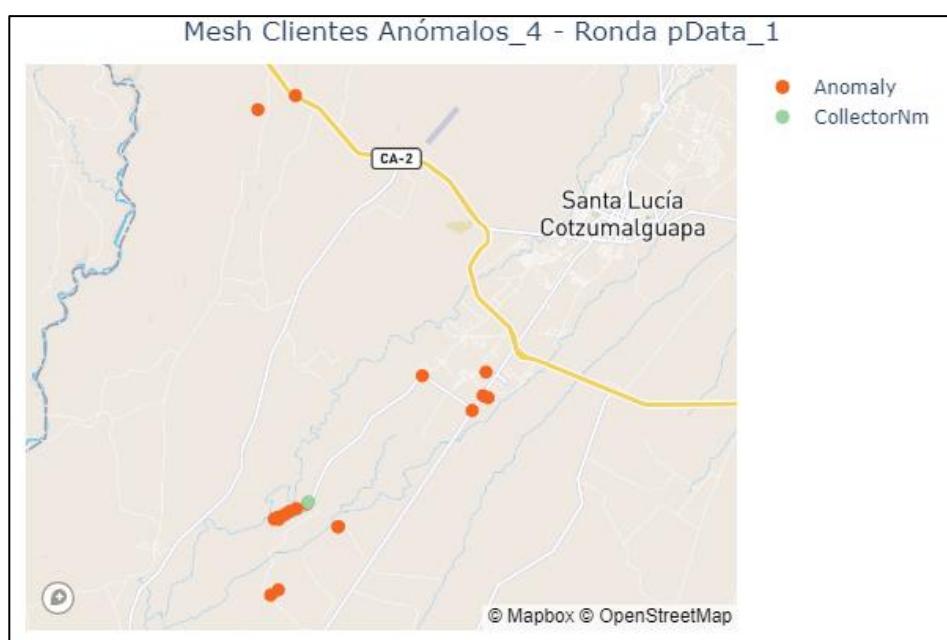


Nota. La figura contiene el mapa con las anomalías detectadas por IF_4.3. Elaboración propia, realizado con Python.

Se aprecia que, para un MG, tanto puntos cercanos como distantes son identificados como anomalía y a la vez no. Al hacer una inspección de los datos varias de las anomalías identificadas de hecho corresponden a información correcta. Al momento de imputar valores se considerará esta situación para evitar una alteración que afecte la precisión de los datos.

Otra incidencia es la de la imagen siguiente, donde para un MG todos sus MC son identificados como anomalía, incluso cuando el análisis humano indique que no existe una gran distancia entre ellos. Luego de revisar los datos podría suceder que la anomalía provenga del MG como tal, al tener pocos MC y estar en un área remota.

Figura 30.
Anomalías detectadas por IF_4.4



Nota. La figura contiene el mapa con las anomalías detectadas por IF_4.4. Elaboración propia, realizado con Python.

Teniendo en mente las situaciones antes descritas se procedió a imputar un valor para corregir la coordenada que es una anomalía. La validación realizada fue respecto a las coordenadas del polo. Siendo que la asignación será la coordenada del polo, se calculó la distancia entre el polo y el correspondiente MG, luego se comparó con la distancia desde el MC al MG. Si la distancia respecto al polo es menor entonces se procede a imputar, de lo contrario las coordenadas permanecen igual.

Figura 31.

Coordenadas corregidas 1



Nota. La figura contiene el mapa con coordenadas corregidas caso de ejemplo 1. Elaboración propia, realizado con Python.

Al parecer la anomalía denominada como recurrente lo era efectivamente y se ha sustituido por una ubicación considerablemente más cercana al MG.

Sin embargo, existen otros casos donde no es tan evidente la corrección. En la siguiente imagen se muestra un escenario donde algunas coordenadas son corregidas y parece ser correcto, mientras que otras que también están distantes no son corregidas. A este punto y dado el enfoque basado en datos, no parece correcta la posición de contradecir a los datos. De igual forma la experiencia empírica indica que entre los sectores donde están las anomalías existen diferencias de altura considerables que bien podrían ser la causa de una comunicación entre equipos tan distantes. Pero, esto será abordado al trabajar con el análisis de las altimetrías.

Figura 32.
Coordenadas corregidas 2

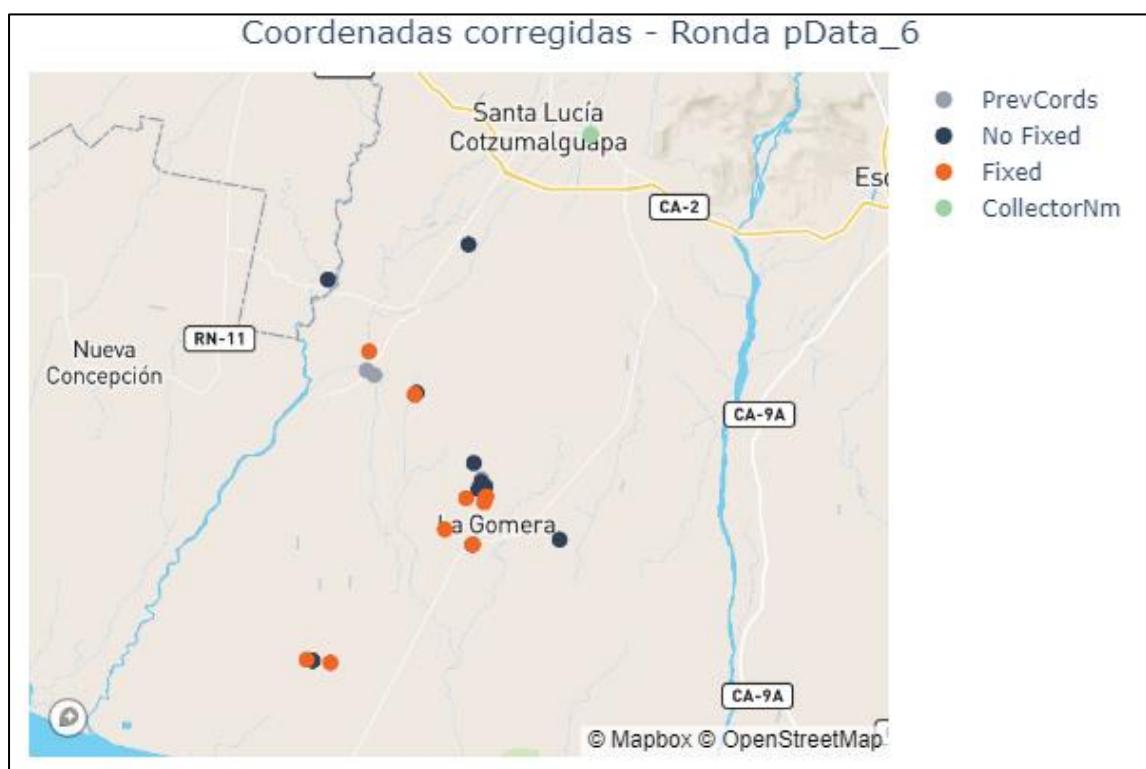


Nota. La figura contiene el mapa con coordenadas corregidas caso de ejemplo 2. Elaboración propia, realizado con Python.

El último ejemplo que se muestra corresponde a un MC con proporción similar de correcciones y no correcciones. Y con esto se da por concluida la limpieza de las coordenadas de los MC. La imputación de valores a las coordenadas vacías corresponderá al siguiente capítulo.

Figura 33.

Coordenadas corregidas 3



Nota. La figura contiene el mapa con coordenadas corregidas caso de ejemplo 3. Elaboración propia, realizado con Python.

Las comunicaciones a través del tiempo van más allá de los alcances del presente trabajo.

3.2. Generación de características

De manera paralela a los procesos de extracción de información de comandos y la validación de la información de los equipos, así como de su carga en el mercado de datos, se produjo la generación de las dos características que dependían de API's externas. Es decir, la característica de Fresnel y la característica de imagen satelital.

Debido a la evolución de la red en el tiempo a través de las rondas incluidas en el proceso de entrenamiento, sucede que la mayoría de los datos geográficos se mantienen constantes. Como la generación de las características referidas requiere de usar servicios externos, uno de los cuales es de pago, se realizó una tarea de selección de coordenadas de equipos y pares de coordenadas de radioenlaces, con el fin de generar solo una vez la característica para cada valor único de esta combinación de campos. El coste computacional también fue un factor de peso para esta decisión, dado que el intervalo de espera en la generación de estas características se contó en días.

Al terminar el proceso de validación de la información geográfica, se procedió a asociar las características con sus correspondientes registros en el resto del conjunto de datos. Luego de considerar las correcciones necesarias se procedió a cargar los conjuntos de datos al mercado de datos.

3.2.1. Generación característica de Fresnel

Es una idea que surge del conocimiento del ámbito del problema y al saber de varias fuentes sobre la existencia de tales herramientas de análisis para radioenlace. El análisis de Fresnel es utilizado para determinar el nivel de obstrucción de una o más zonas de Fresnel. Esta obstrucción puede implicar la

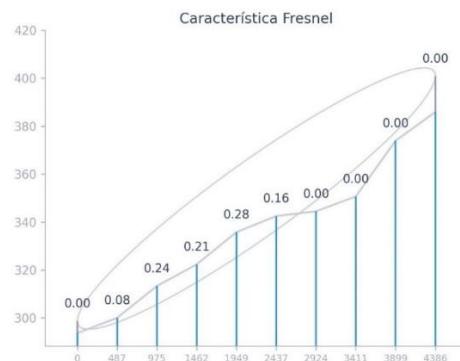
distorsión de la señal que viaja en el espacio entre los dos dispositivos de comunicación.

En un inicio se intentó hacer uso de una librería de Javascript que se integra a una interfaz *web* que permite obtener una figura de la zona de Fresnel y de manera interactiva muestra el porcentaje de obstrucción al posicionar el cursor a lo largo del eje x. Extraer esa información que era visualizada de manera interactiva y calculada con un elemento gráfico de la librería complicó la generación masiva.

La solución fue programar la misma idea en Python, con el fin de obtener los datos en el lenguaje base de todo el trabajo de graduación. En los anexos se explica la forma de obtener las características como tal, y en el cuaderno *FresnelFeature.ipynb* se detalla su implementación en Python y la generación de la gráfica, mucho más modesta que la de la librería original.

Figura 34.

Característica de Fresnel



Nota. La figura muestra una visualización de la característica de Fresnel. El desarrollo de esta característica se muestra en los apéndices. Elaboración propia, realizado con Python.

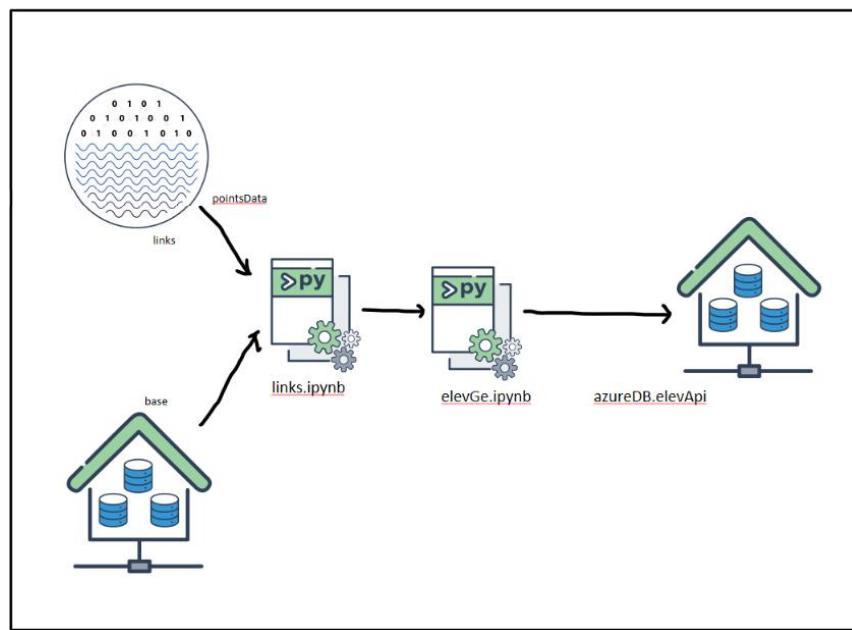
Las líneas celestes indican la altimetría correspondiente a la cantidad horizontal que representa la distancia entre el punto más a la izquierda y los distintos puntos de interés a lo largo de la ruta que une los dos equipos que forman el radioenlace. El número sobre estas líneas corresponde a la característica de Fresnel.

Luego de dar con la función con la capacidad de generar la característica de Fresnel a partir de dos pares de coordenadas, dos alturas y el parámetro de frecuencia de comunicación, se está en posibilidad de una generación masiva de esta característica. Dado que cada consulta realizada a la API de Elevación tiene un costo, se trató de optimizar su uso. Se procedió a trabajar esta generación de características en paralelo con la limpieza de las coordenadas. Pese a la función general creada, fue necesario crear ciertas variantes que permitieran con mayor facilidad la generación masiva.

Dentro del proceso de generación de la característica de Fresnel está la adquisición de la información relativa a las alturas. Debido al valor de esta información se decidió almacenar las alturas en una base de datos en la nube. Por tal motivo se realizó primero la extracción de las altimetrías para todos los puntos de la red. Utilizando la siguiente infraestructura.

Figura 35.

Característica de Fresnel



Nota. La figura esquematiza la generación de la característica Fresnel. Elaboración propia, realizado con Adobe Illustrator.

Es necesario indicar que para los cuadernos —links.ipynb y elevGe.ipynb— existe más de una versión, esto debido a una ejecución antes y una después de la limpieza de las coordenadas, adicional a la variedad de fuentes de donde se obtiene la información de radioenlaces. También que se guardaron archivos en formato parquet como respaldo y debido al tiempo considerable en su carga a la BD, de modo que se pudiera trabajar de manera provisional con los archivos parquet. Por elección de diseño se eligieron 20 puntos a lo largo del radioenlace para la conformación de la altimetría del enlace.

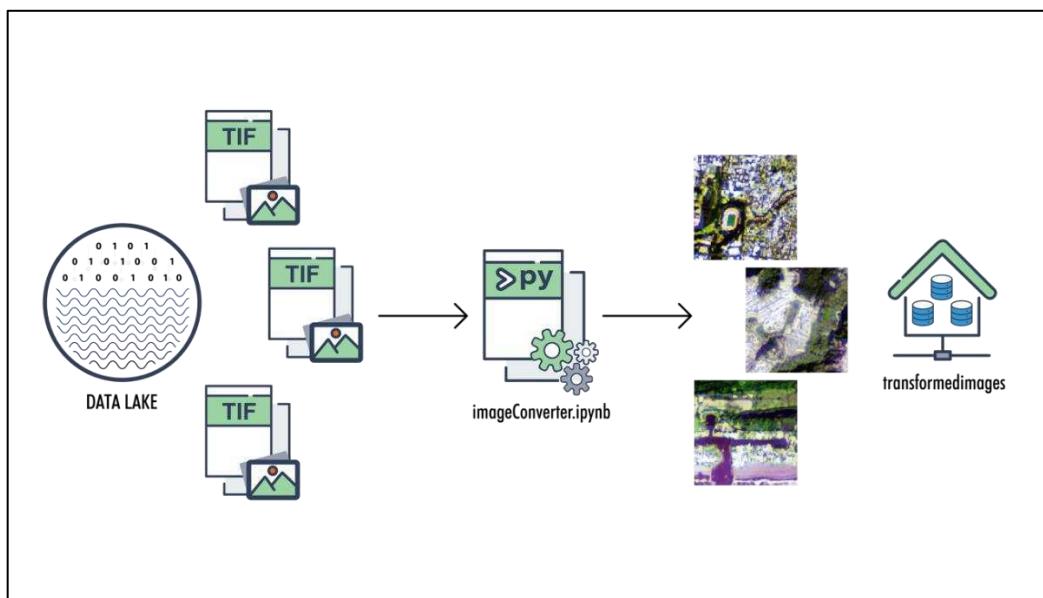
Por último, solo queda la generación masiva de la característica de Fresnel a partir de pares de coordenadas con sus correspondientes alturas, misma que se almacenará para ser utilizado por los algoritmos. Siendo que se requieren las alturas, y estas serán agregadas en el siguiente capítulo, se comentará posteriormente cómo se procedió con esta ejecución.

3.2.2. Generación característica imagen satelital

Al igual que la característica anterior, esta surge del conocimiento del ámbito de la red *mesh* en cuestión y gracias a la experiencia que indica que, a pesar de una altimetría favorable y zonas de Fresnel libres, el entorno juega un papel importante para determinar la comunicación efectiva o no de una determinada región. Como se mencionó antes al discutir la extracción de características se indicó que se hace uso de una API de Google, específicamente la API del Google Earth Engine, la cual permite entre otras cosas la superposición de capas de fotografías en el mapa, la creación de polígonos y la exportación de estos polígonos como imágenes, en este caso en formato tif. Se procedió como sigue.

Figura 36.

Característica de imagen satelital



Nota. La figura esquematiza la generación de la característica de imagen satelital. Elaboración propia, realizado con Adobe Illustrator.

No se consideró siquiera la posibilidad de entrenar los modelos directamente en formato tif. Pareció necesario obtener un formato de archivo que pueda ser asimilado por la mente humana, como una ayuda al momento de trabajar con los datos. Ante la imposibilidad herramientas útiles y confiables para trabajar con la conversión de las imágenes, fue necesario atender el problema realizado con Python, en el cuaderno imageConverter.ipynb se aborda el problema con el uso de la librería Rasterio y manteniendo por el momento las imágenes almacenadas en la nube, con la idea de utilizar Google Colaboratory para ejecutar las extensas jornadas de transformación.

3.3. Diseño e implementación de interfaz de usuario

La interfaz de usuario, comúnmente referida como interfaz gráfica, es la capa que permite la interacción del usuario humano con el sistema computacional. Al menos parece que hasta este momento sigue siendo esa la situación. En el caso particular de este proyecto se utilizaron ciertas herramientas de visualización y de control que permiten a un usuario interactuar con el sistema, principalmente con poder visualizar los datos y solicitar información o ejecución de algún proceso.

Otra interpretación usual para interfaz de usuario es que su enfoque está directamente relacionado con el usuario final del sistema y no precisamente con los usuarios de desarrollo. Pero al igual que el usuario final, la interacción de una persona desde el desarrollo es parte integral del sistema, y esta tarea se facilita al hacer uso de ciertas herramientas que permitan la interacción con el código y con los datos.

3.3.1. Interfaz para desarrollo

Al momento de trabajar en un proyecto basado en datos, como en general es importante tener un control del trabajo realizado para evitar posibles pérdidas o confusiones. Tanto para la parte escrita como para el código resulta de bastante utilidad almacenar versiones históricas que permitan acceder a información que en algún momento pudiese ser necesaria. Desde toda la extracción de la información hasta su depósito en el *data lake* y luego en el *data mart*, hasta su ingestión desde el *data mart* de donde se utilizan los datos para generar modelos, requieren de una interacción del humano con el código, con los datos y con los resultados de los cálculos que correspondan.

Como el enfoque abordado es basado en datos resulta importante que el desarrollador pueda ir visualizando los datos al tiempo que los trabaja, de modo que pueda validar si las operaciones se realizaron correctamente. Esto no se presenta de cara al usuario final, pero disponer de ciertas herramientas que faciliten la tarea es parte de la interacción del sistema con el usuario que trabaja en su implementación. Esto se recalca debido a la cada vez más notoria aparición de un tercer jugador en esta partida, el sistema de IA.

En términos puntuales lo que se hizo fue lo siguiente. Se creó un repositorio de GitHub para alojar el código luego de haber pasado cierta etapa de desarrollo. Se creó el correspondiente ambiente realizado con la interfaz gráfica de Anaconda y se agregó al repositorio para referencia. Se utilizó lenguaje de programación Python y se utilizó el IDE Visual Studio Code, herramienta de gran utilidad dado que permite la utilización de extensiones que fueron efectivas, por ejemplo, para gestionar el repositorio, así como la ejecución de Docker con una interfaz gráfica de apoyo.

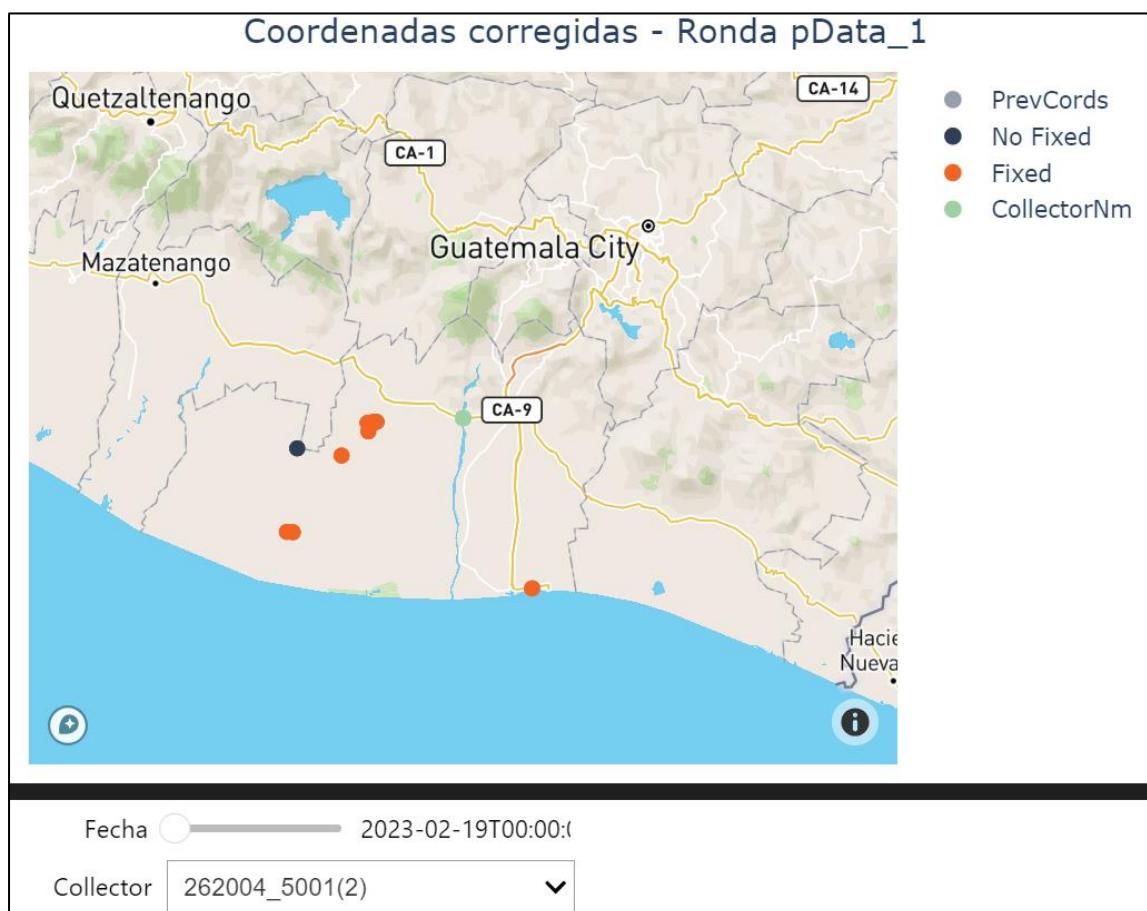
Es momento de hacer la *Nota* aclaratoria sobre el repositorio de desarrollo utilizado durante la implementación y el repositorio adjunto a este trabajo de graduación. Al hacer uso del archivo gitignore, se restringió el acceso en el ámbito del desarrollo tanto a los datos como a las credenciales, que como es evidente no están para ser compartidas. Incluso con esa precaución existía la posibilidad de revelar información menos sutil, como estructuras de bases de datos o nombres de tablas, por tal motivo hubo una revisión de los archivos y fueron seleccionados para publicar junto con el trabajo, aquellos que no comprometen información restringida, y principalmente los que se asocian a las ideas que forman la solución, y con esta selección se creó el segundo repositorio de muestra.

Todas las funciones que pudieron ser almacenadas en el *background* como archivos .py fueron trabajadas de esa forma, muchas por ser procesos que no requerían una visualización en cuaderno y otras para aligerar los cuadernos, guardando las funcionalidades más generales en archivos que acceden los cuadernos de Jupyter.

Al trabajar con tantos datos se vuelve complicada su asimilación y realizar revisiones manuales de la información en alguna herramienta de visualización de mapa pareció ser menos útil que crear visualizaciones dentro de los cuadernos. Se eligió la librería Plotly por su inherente interactividad y se complementó con la librería IpyWidget para agregar una capa adicional de control. Las imágenes mostradas en el inciso anterior fueron extraídas utilizando mapas interactivos que permitían moverse a través de la variable temporal, así como eligiendo la aplicación de ciertos filtros de interés. En el código esta función es definida con el nombre de Rdiscrete_map y sus variantes.

Figura 37.

Mapa interactivo para notebook



Nota. La figura muestra la interfaz de un mapa interactivo dentro de un cuaderno. Elaboración propia, realizado con Python.

La imagen anterior muestra una de las visualizaciones generadas para la interacción del usuario de desarrollo. Las distintas variedades de gráficas permitieron entender de mejor manera los datos y tomar decisiones basadas en este análisis. De este modo, luego del desarrollo de la solución se estuvo en posibilidad de crear una interfaz gráfica para uso del usuario final de la aplicación.

3.3.2. Interfaz para usuario final

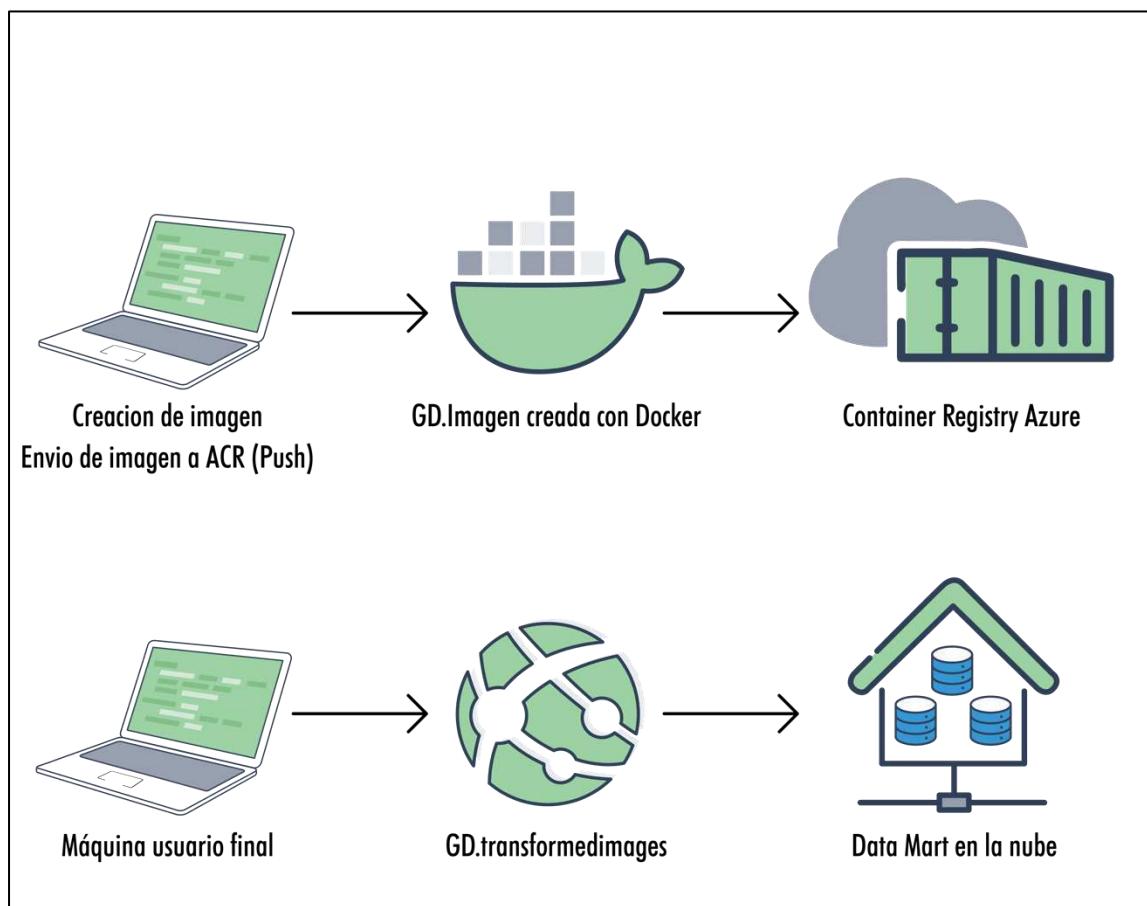
Esta interfaz es el resultado final del proyecto, una forma de interactuar con la solución de modo que sea posible al usuario que lo deseé, obtener información del sistema basado en los datos disponibles, así como solicitar predicciones a los modelos seleccionados por su mejor desempeño.

Debido a que el desarrollo fue trabajado en computadora personal y que no se dispuso de un servidor dedicado para el proyecto, así como a ciertas particularidades que complicaron el uso de la Raspberry para alojar la interfaz de usuario, se tomó la decisión de alojar la interfaz como una aplicación en la nube.

Concretamente, se utilizó la nube de Azure para subir la imagen de un contenedor que contenía la aplicación, este contenedor fue ejecutado y alojado en un dominio público que puso a disposición de los usuarios el acceso desde cualquier computador o dispositivo móvil capaz de ejecutar un explorador de internet. La siguiente imagen muestra la infraestructura creada en la nube para tal propósito.

Figura 38.

Arquitectura en Azure para interfaz gráfica



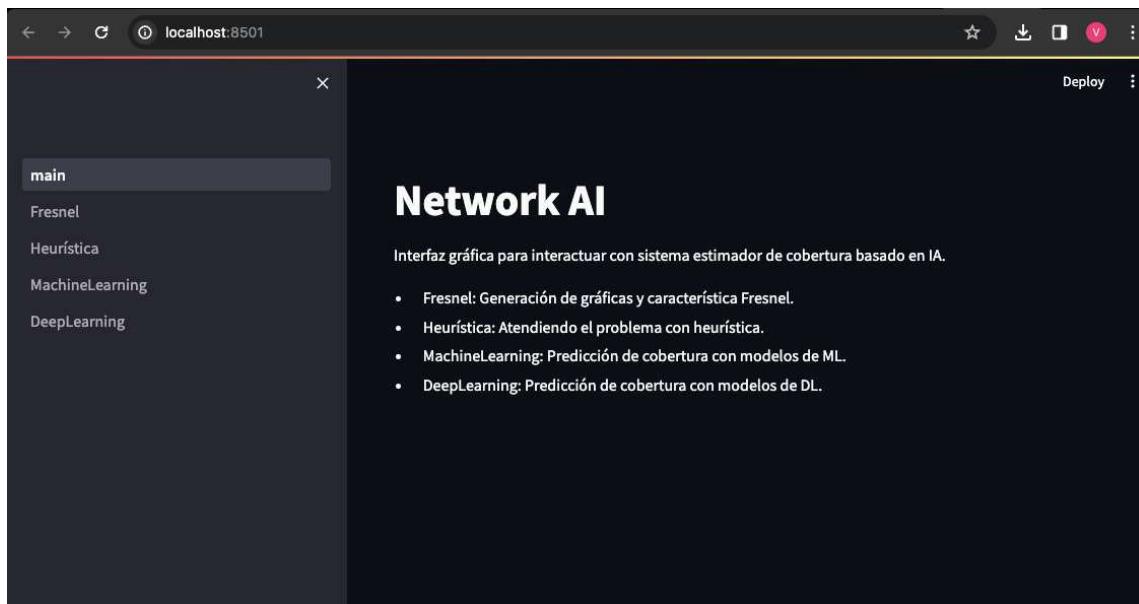
Nota. La figura muestra la arquitectura en la nube de Microsoft Azure para el despliegue de la interfaz gráfica. Elaboración propia realizado con Adobe Illustrator.

En la anterior imagen se muestra el proceso por el cual la aplicación es puesta a disposición del usuario final por medio de la nube de Azure. Inicialmente es necesario crear una imagen que contenga la aplicación que se publicará, esto se hace por medio de un dockerfile, este, al igual del resto de componentes necesarios para la aplicación se encuentran en el repositorio en la carpeta NetworkAI.

En el portal de Azure se pueden acondicionar los recursos necesarios, en este caso es un Container Registry que tiene la capacidad de almacenar imágenes de contenedores, así como una WebApp, que puede publicar aplicaciones en internet a partir de varias fuentes, en este caso se está poniendo a funcionar la aplicación al ejecutar la imagen del Container Registry.

Dado que cierta información del *Data Mart* está almacenada en la nube, se requirió dar acceso a la WebApp. En cuanto a la aplicación publicada, es una interfaz *web* realizada con Streamlit, que es un marco de trabajo en Python que facilita considerablemente la puesta en marcha de una interfaz *web*. En la siguiente imagen se muestra una captura de la aplicación en Streamlit.

Figura 39.
Arquitectura en Azure para interfaz gráfica



Nota. La figura muestra la interfaz gráfica de la solución. Elaboración propia, realizado con Streamlit.

La interfaz gráfica cuenta de 4 pestañas de interacción, Fresnel, la cual permite calcular la característica y su gráfica a petición del usuario para cualquier par de coordenadas válidas, Heurística, que muestra la solución particular al problema basada en experiencia e intuición, *machine learning*, donde se puede solicitar predicciones a los mejores modelos seleccionados de ML, *DeepLearning*, donde se puede solicitar predicciones a los mejores modelos seleccionados de DL.

Lo anterior representa el despliegue final del trabajo de graduación, mismo que se dejó a disposición de los usuarios que puedan requerir información de la red de comunicaciones haciendo uso de la solución propuesta basada en Inteligencia Artificial.

4. DESARROLLO DE MODELOS

Se llega finalmente a la parte que resulta de mayor interés, dado el objetivo del proyecto. Solo es posible resaltar que los modelos no podrían ser entrenados sin el proceso previo de extracción de las características, su transformación y la generación de algunas de estas. Se encuentra en la literatura y parece ser de dominio general el que, en los proyectos de ciencia de datos, la gran mayoría del tiempo es requerido por las etapas previas a la creación de los modelos, en el presente trabajo, seguro lo fue. Dadas las condiciones en que se llega a este desarrollo de los modelos, solo se espera que los algoritmos presentados resulten de interés.

Otra tarea que se aborda antes del entrenamiento de los modelos es la del etiquetado de los datos, el enfoque principal en este capítulo es el aprendizaje supervisado, sin embargo, los datos no vienen explícitamente con la etiqueta a predecir, por tal motivo se vuelve necesario hacer una tarea de etiquetado de los datos para conseguir obtener la variable de interés a predecir. Según se menciona en la literatura consultada, lo que se hará es aprovechar las etiquetas naturales del conjunto de datos, dado que dentro de los datos existentes se encuentra también la clave para la etiquetación.

Las características tal como se entregaron al Mercado de Datos requieren de una etapa previa antes de ingresar a los modelos. Este es un tratamiento de características distinto del que se presentó en el capítulo anterior. Aquí, la fuente de la verdad es el *Data Mart*, no existen fuentes dispersas a las cuales requerir información. Con lo que contenga el directorio se procederá a trabajar hasta el fin del trabajo. Según su tipo los modelos requerirán las características de tal o

cual manera, incluso se dará el caso de que no se utilizarán todas las características.

Las vistas, que serán almacenadas en el *data mart*, serán el resultado del procesamiento previo al entrenamiento de los modelos, una exploración general de los datos, la imputación a los valores faltantes, así como la transformación, selección o generación de características. Estas vistas serán utilizadas para entrenar los modelos, de modo que distintos modelos serán entrenados con distintas vistas.

Los modelos serán de *machine learning* y *deep learning*, de igual forma se presentará una heurística para abordar el problema de manera visual que pareció de utilidad y que es inspirada en proyectos conocidos que abordan de formas distintas la tarea de determinar cobertura.

4.1. Ingeniería de características

Para hacer referencia a lo expuesto en la revisión de la literatura, acá se abordan las siguientes tareas relacionadas con las características: transformación, generación y selección. La generación de características en el capítulo anterior involucró el uso de APIs externas, por lo que es principalmente una tarea de extracción.

Dados los datos existentes en el mercado de datos, se procederá a producir las vistas que serán la materia prima que usarán los algoritmos. La imputación de valores faltantes también es de interés, y se incluye en esta misma sección.

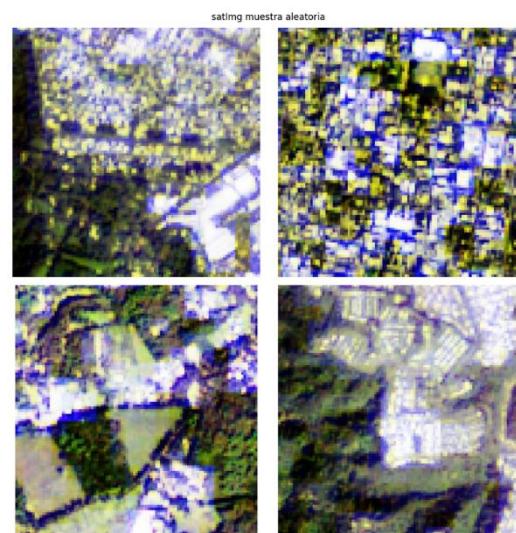
Una vez los datos fueron imputados y las características transformadas para ser utilizadas por los algoritmos, se utilizarán técnicas de selección de características para reducir la cantidad de columnas en los conjuntos de datos de entrenamiento, así como para eliminar posibles perturbaciones debido a interrelaciones entre las características.

4.1.1. Ingeniería de característica imagen satelital

Esta característica consiste en 25600 imágenes satelitales de lado igual a 1 Km, en el punto central se encuentra el par de coordenadas de cada uno de los equipos de red, o de la mayoría, dado que nunca faltan los casos de datos vacíos. En la siguiente imagen se expone una muestra aleatoria tomada tras la lectura del directorio transformedImages en la ubicación en la en el *data mart*.

Figura 40.

Muestra aleatoria de característica imagen satelital



Nota. La figura contiene una muestra aleatoria de la imagen satelital. Elaboración propia, realizado con Python.

Según conocimiento empírico sobre la red de comunicaciones se ha considerado la importancia de la imagen satelital, existen soluciones propietarias o de *software* libre que de alguna manera involucran información referente al terreno, tal como cantidad de vegetación, tipo de clima, entre muchas otras variables. Para el caso de este proyecto, la inserción de tal información se espera realizar mediante la característica imagen satelital, o satImg, que es el nombre con el que se aprecia en los cuadernos y en los nombres asignados a cada imagen.

Una discusión sobre lo que se podría obtener de las imágenes implica el uso de la inteligencia e intuición humana, a mayor cantidad de verde se podría intuir esta característica, a mayor cantidad de cuadros consecutivos de color gris se podría intuir un área urbana, ante la presencia de una franja de color azul se podría intuir la costa, entre otros. Un humano puede distinguir árboles, caminos, casas, edificios, entre otros, luego de inspeccionar las imágenes, lo que puede ayudarle a decidir si es factible que un equipo logre conectarse a la red.

Es posible generar características a la medida, como obtener el promedio del canal verde de la imagen, aplicar una transformada de Fourier, o usar algoritmos de reducción de dimensionalidad que permite tener una representación más ligera de las imágenes para que puedan ser entregadas a los modelos. Sin embargo, esta transformación de la característica imagen satImg, se ve afectada, como cualquier otro tipo de dato, por las anomalías.

Figura 41.

Anomalías detectadas en característica imgSat



Nota. La figura contiene ejemplos de anomalías seleccionadas manualmente en la característica de imagen satelital. Elaboración propia, realizado con Python.

Aquí se resalta la importancia de tener las imágenes almacenadas en un formato ameno para los humanos. Al hacer una inspección en las imágenes se han detectado algunas como las que se muestran en la imagen anterior que, sin lugar a duda, son erróneas y no contienen la información que se pretende extraer de ellas respecto a la geografía del terreno. Luego de comprobar que las coordenadas de algunas de estas anomalías corresponden a lugares reales y con visualización satelital adecuada, fue necesario plantear una forma de seleccionar tales anomalías.

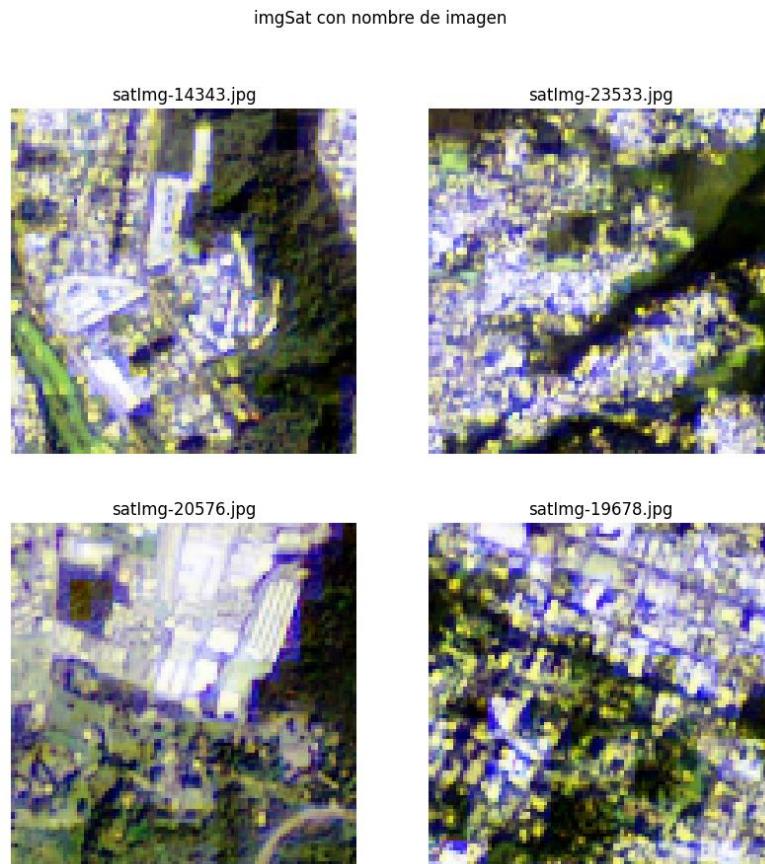
En cierto modo se ejecutó una tarea inversa a la realizada en la detección de características anómalas en el capítulo anterior, la razón es que en esos casos no se tiene una idea precisa de cómo debería —verse— una anomalía, no es fácil generalizar descripción de una característica, de hecho, para describirla se realizó el uso de la estadística y gráficos que resumieran la totalidad de los datos, al hacer una inspección visual de las imágenes de inmediato se reconocen las que están fuera de lugar, sin necesidad de recurrir a la estadística. Sin embargo, es necesario matematizar de alguna manera la detección de las anomalías para la aplicación sobre todo el conjunto de datos.

En la inspección visual también se observaron otros tipos de anomalías, como imágenes con excesivas nubes, por ejemplo. Como al generar la característica se configuró el parámetro para reducir la aparición de nubes, y debido a que siempre es necesario poner una cota a la precisión esperada, las anomalías que serán de interés son las que comprenden la totalidad de la imagen o una considerable porción de ella, con pixeles no correspondientes a una imagen satelital, sino un color aparentemente homogéneo, es decir, las anomalías en imágenes previamente mostradas.

Se accedió al directorio `transformedImages` haciendo uso de la librería Tensorflow, de donde se obtuvo un *batch dataset* (conjunto de datos por lotes) que contiene la fotografía, así como la asociación con su correspondiente ruta, este objeto fue el que se iteró en la transformación de las características. A continuación, una muestra aleatoria tomada de uno de los 256 lotes del conjunto de datos.

Figura 42.

Muestra del conjunto de datos leídos con TF



Nota. La figura contiene una muestra aleatoria del conjunto de imágenes leídas con Tensor Flow y su correspondiente nombre de imagen. Elaboración propia, realizado con Python.

4.1.1.1. Imágenes con modelo pre-entrenado

Cuando las redes convolucionales comenzaron a tener auge, hubo un momento de inflexión y fue en la competencia ImageNet de 2012, con la red convolucional AlexNet mostrando un éxito considerable. A partir de ese punto este tipo de redes se popularizó. Para abordar el problema primero se consideró un modelo previamente entrenado y con resultados satisfactorios, en este caso

primer y segundo lugar en la competencia ImageNet de 2014. No se tomó el modelo como tal sino una de sus capas profundas, por supuesto, posterior a la etapa convolucional.

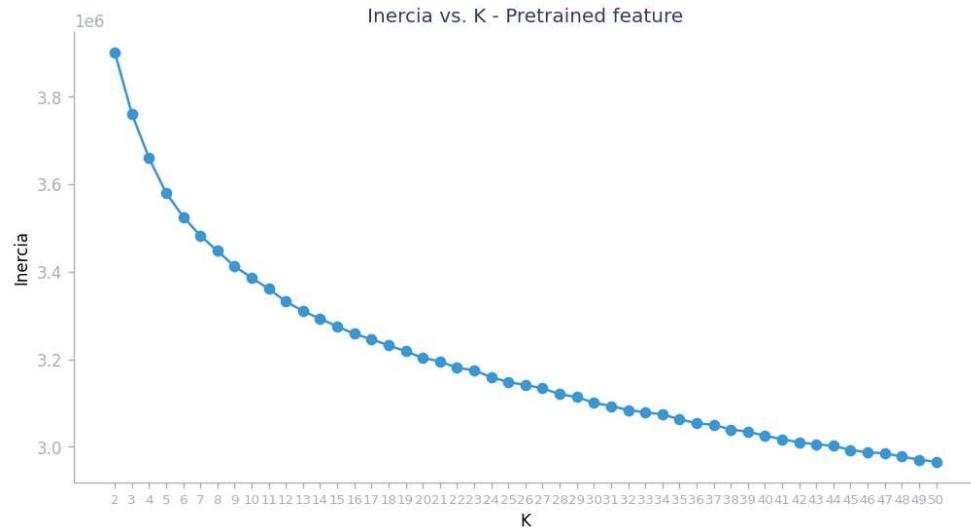
El resultado de aplicar el modelo pre-entrenado es una reducción de la característica de aproximadamente 29403 dimensiones a 4608. Esta corresponde a tomar una de las capas profundas del modelo VGG16 de Tensorflow.

El algoritmo utilizado fue Kmeans, mismo que asume K distribuciones normales para los datos, con K distintas medias, luego utiliza los datos para estimar los valores capaces de representar las distribuciones normales subyacentes y adjudicar a una particular imagen su pertenencia a una u otra distribución. Para el resto del tratamiento de las imágenes se utilizó el mismo modelo y estrategia de agrupación que se explica a continuación.

Las métricas de interés para comparar el modelo Kmeans son la Inercia, suma del cuadrado de la distancia entre un punto y su centro de *cluster* (grupo) más cercano, y la Silueta que compara la media de la distancia entre *clusters* con la distancia media hacia el *cluster* más cercano distinto del *cluster* al que pertenece cada muestra. Se menciona en la literatura que una forma de determinar el valor adecuado de K es utilizando el método del códo (elbow en inglés), el cual consiste en iterar el valor de K y entrenar los correspondientes modelos Kmeans, identificando el valor donde la función de pérdida no mejora significativamente a pesar de aumentar K. A continuación, se muestra cómo se comportaron estas métricas en una ronda que llegó hasta K igual a 50.

Figura 43.

Inercia vs K – modelo pre-entrenado

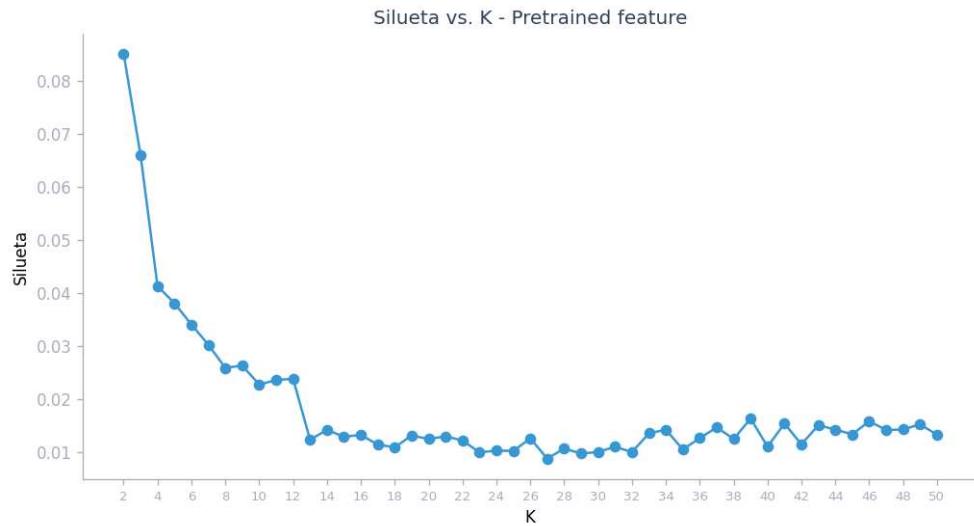


Nota. El gráfico muestra la Inercia vs K para una característica con modelo pre entrenado.
Elaboración propia, realizado con Python.

Se aprecia que el valor de la métrica mejora a medida que aumenta K, pero al llegar a 50 seguía sin identificarse el valor de inflexión en dicha mejora. Como apoyo, la silueta (silhouette) permite identificar la poca conveniencia en seguir aumentando el valor de K.

Figura 44.

Silueta vs K – modelo pre-entrenado



Nota. El gráfico muestra la Silueta vs K para una característica con modelo pre entrenado.
Elaboración propia, realizado con Python.

La silueta mide qué tan similar es un punto respecto a su propio grupo en comparación con los demás grupos. Analizando las dos gráficas anteriores se determina el valor $K=12$ como una decisión de diseño y los resultados en cuanto a agrupación consiguieron el objetivo de aislar las anomalías en un solo grupo, para este caso particular en el grupo número 2.

Figura 45.

Modelo pre-entrenado Kmeans anomalías

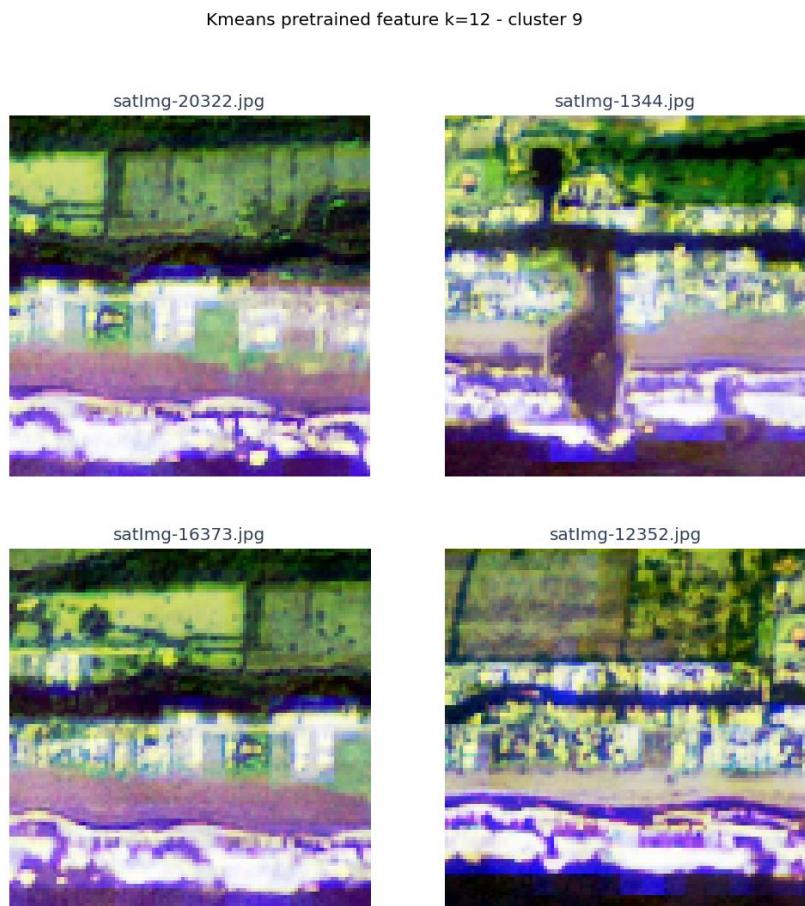


Nota. La figura muestra las anomalías detectadas por el modelo entrenado con la característica del modelo pre entrenado. Elaboración propia, realizado con Python.

Además de conseguir este objetivo el modelo fue capaz de agrupar las imágenes en grupos fácilmente identificables por un ser humano, como pudiese ser un área costera, por ejemplo.

Figura 46.

Imágenes en áreas costeras pre-entrenado Kmeans

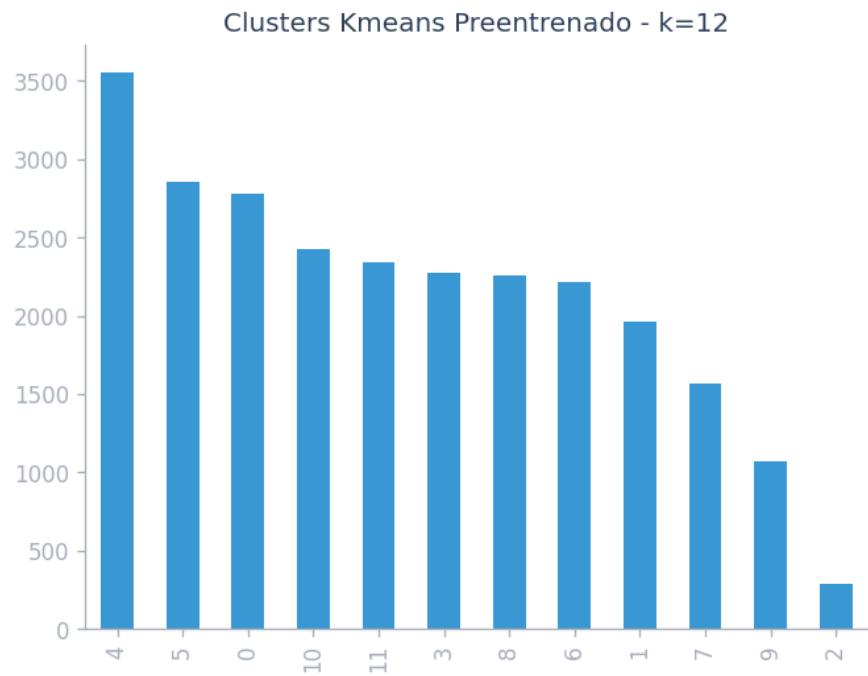


Nota. La figura muestra las áreas costeras detectadas por el modelo entrenado con la característica del modelo pre entrenado. Elaboración propia, realizado con Python.

En los anexos se incluye la visualización de todos los *clusters* para los modelos entrenados en esta parte del trabajo, con el fin de que se pueda apreciar el funcionamiento de la agrupación de imágenes. Solo queda mencionar que la distribución final de los datos redistribuidos a los distintos *cluster* es de interés puesto que podría iniciarse a arrastrar problemas de desbalance de clases.

Figura 47.

Histograma de grupos para característica pre-entrenada



Nota. El gráfico muestra el histograma de los grupos identificados al utilizar la característica de modelo pre entrenado. Elaboración propia, realizado con Python.

Cabe resaltar que el *cluster* de las anomalías es el que tiene menor cantidad de imágenes, lo que solo refuerza la idea de que el modelo está funcionando de manera adecuada. Sin embargo, para ser objetivos es necesario utilizar métricas. Se presentan características más que son transformaciones de la característica imagen satelital, para comparar sus resultados, y más que eso, para disponer de ellas al momento de entrenar los modelos de verdadero interés.

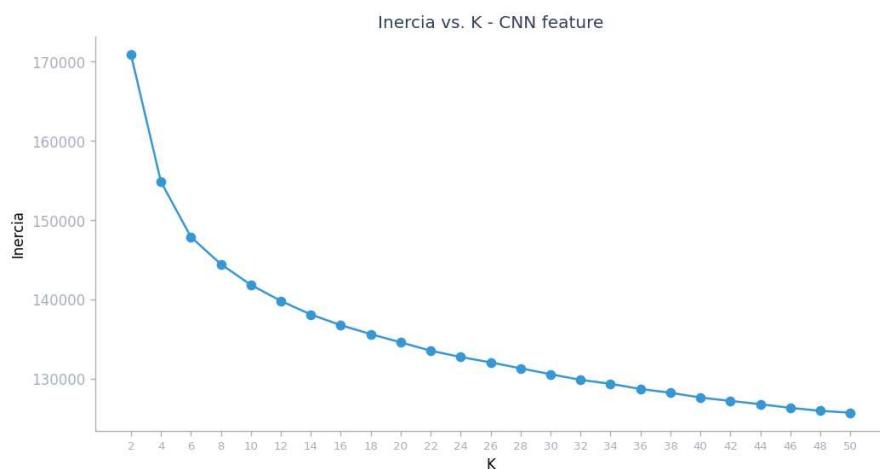
4.1.1.2. Imágenes con modelo CNN

No es lo mismo utilizar una red neuronal entrenada que crear una específica para el problema que se atiende, tanto por la componente subjetiva como por el hecho de que un modelo pre-entrenado usó imágenes que pueden ser muy distintas a las del problema en cuestión. CNN viene del inglés *Convolutional Neural Network*, para este modelo se creó una secuencia de Keras de Tensorflow, que permitió la aplicación de la etapa convolucional de una CNN a la característica de imágenes satelitales, la cual tiene un total de 12800 dimensiones.

Al buscar por el punto de inflexión del algoritmo Kmeans con la característica CNN se procedió igual que en el caso anterior y obteniendo los siguientes valores para las métricas.

Figura 48.

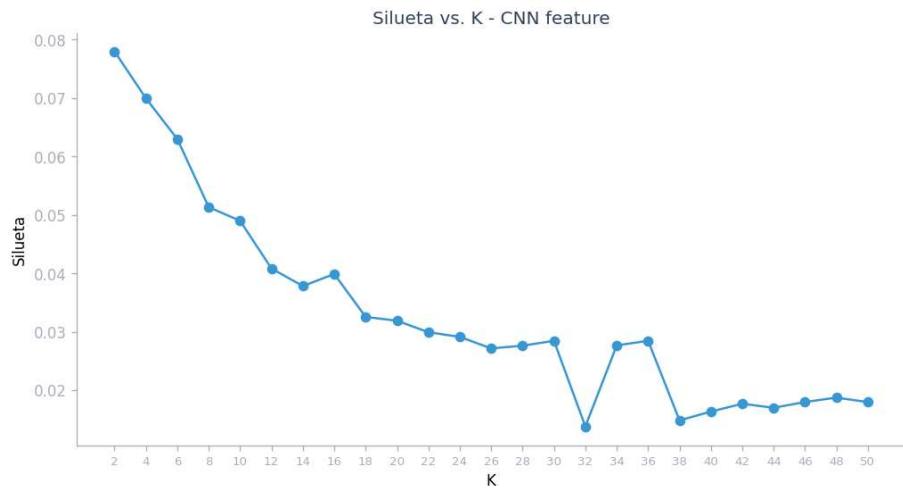
Inercia vs K – modelo CNN



Nota. El gráfico muestra la Inercia vs K para una característica con red neuronal convolucional. Elaboración propia, realizado con Python.

Figura 49.

Silueta vs K – modelo CNN



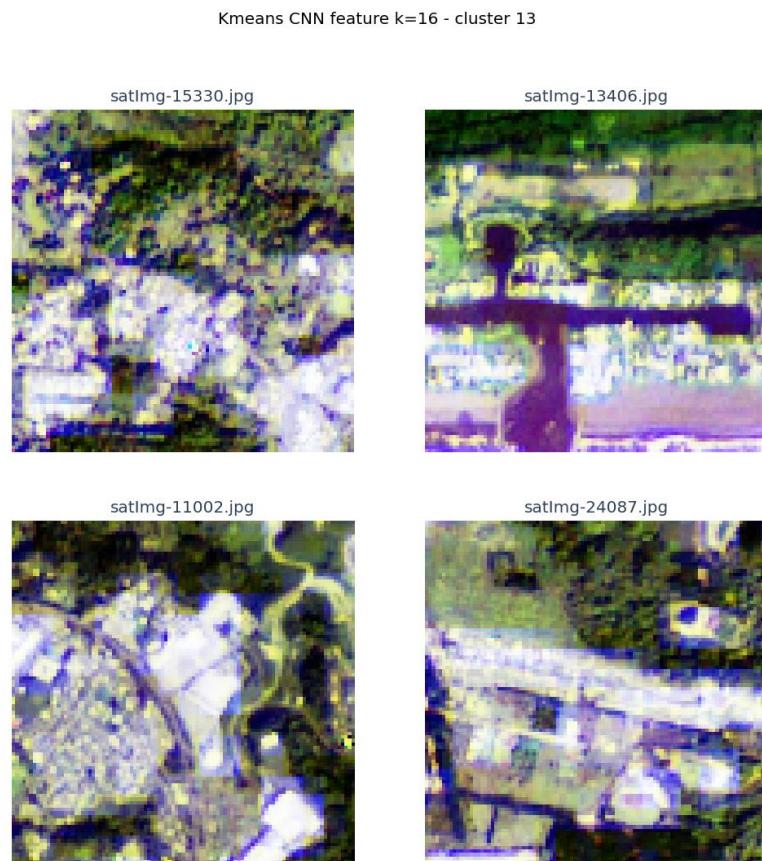
Nota. El gráfico muestra la Silueta vs K para una característica con red neuronal convolucional. Elaboración propia, realizado con Python.

En términos solamente de métrica, se ve una mejora del modelo cnn con respecto al modelo pre-entrenado. Mientras menor el valor de la inercia mejor se comporta el modelo y más cercano sea el valor de la silueta a 1, mejor será la correspondencia de las imágenes con su *cluster* asociado. La decisión de diseño para CNN fue el valor de K =16, lo que repercute en la siguiente distribución.

A pesar de la mejora apreciada en la métrica con respecto al modelo entrenado con la característica pre-entrenada, en la revisión subjetiva de los grupos no se apreció la misma capacidad de segmentación de imágenes. A continuación, se muestran algunos *clusters* que, a pesar de contener imágenes similares, pueden tener también algunas diferentes. Otro factor a resaltar es la incapacidad del modelo para determinar grupos que según el conocimiento del negocio existen, como podría ser el caso de una zona costera.

Figura 50.

Ejemplo de cluster con característica CNN

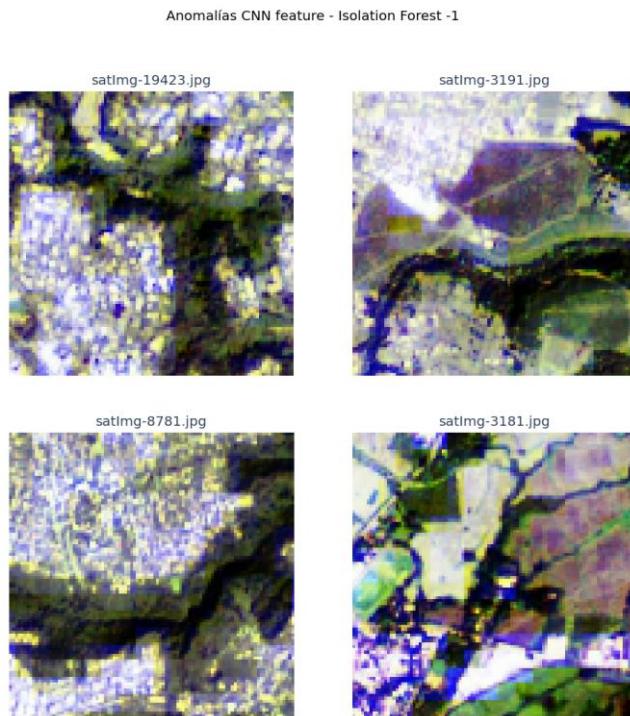


Nota. La figura muestra el ejemplo de las imágenes de un grupo obtenido con el uso de la característica de red neuronal convolucional en el algoritmo K-Means. Elaboración propia, realizado con Python.

Esta característica no fue útil para separar las anomalías con el modelo Kmeans ni con el modelo Isolation Forest, por lo que se descarta el uso de sus grupos y la característica como tal para el entrenamiento de los modelos.

Figura 51.

Anomalías detectadas con característica CNN



Nota. La figura muestra el ejemplo de las anomalías obtenidas con el uso de la característica de red neuronal convolucional en el algoritmo Isolation Forest. Elaboración propia, realizado con Python.

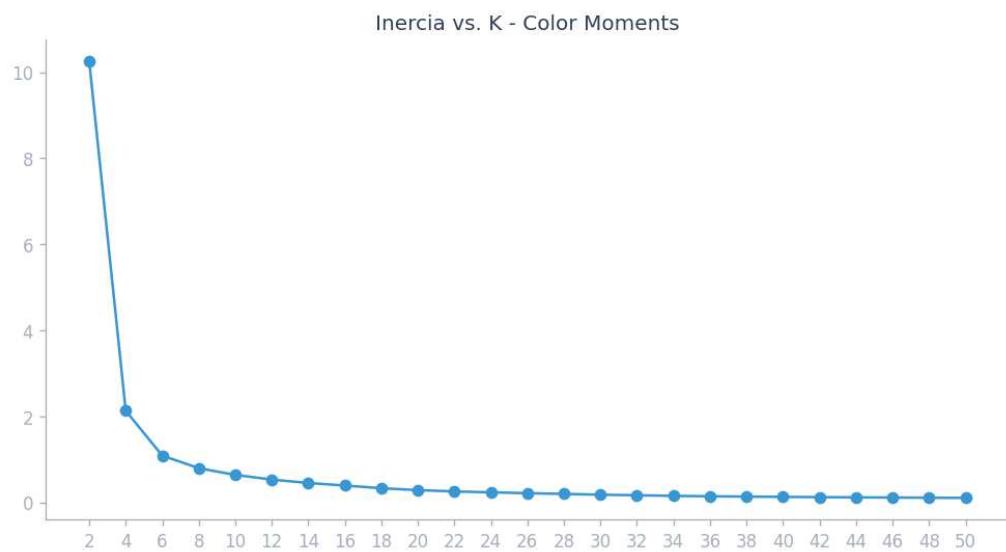
4.1.1.3. Imágenes con momentos de color

Momentos de Color o Color Moments, es el nombre de la característica que implica representar las imágenes con los tres primeros momentos, por cada uno de los canales R (rojo), G (verde) y B (azul). Esta característica fue esencial durante el desarrollo de las dos anteriores características, debido a que es considerablemente más fácil de asimilar sus 9 dimensiones contra las miles de las otras dos.

Se generó la característica y se buscó un valor óptimo de K de la misma manera que se hizo con los modelos previos, iterando el valor de K y obteniendo las métricas de interés.

Figura 52.

Inercia vs K – modelo CM

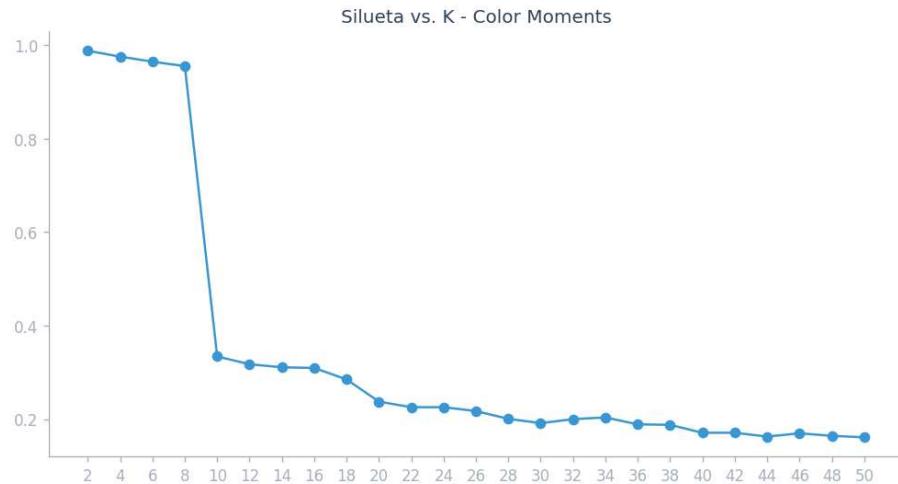


Nota. El gráfico muestra la Inercia vs K para una característica con momentos de color.
Elaboración propia, realizado con Python.

Los valores significativamente menores de la inercia y el aumento en la silueta parece ser consecuencia de la reducción de la dimensionalidad respecto a las dos previas.

Figura 53.

Silueta vs K – modelo CM

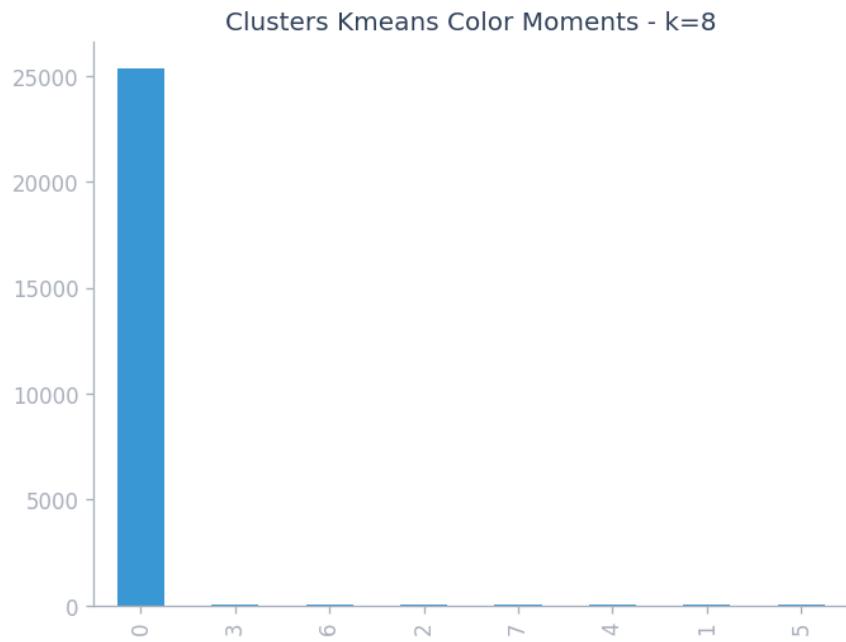


Nota. El gráfico muestra la Silueta vs K para una característica con momentos de color. Elaboración propia, realizado con Python.

Se aprecia una mejora significativa en ambas métricas, y por fin se ve la forma de un codo esperada para determinar el K óptimo. Por lo tanto, se toma la decisión de diseño de utilizar $K = 8$. Al proceder de esa manera se pudo identificar la sensibilidad del modelo ante las anomalías, dado que efectivamente cumple con la tarea de separar las anomalías de los valores normales, pero luego agrupa las anomalías en grupos específicos y todas las imágenes normales las mantiene en un grupo separado, a continuación, se muestra la distribución de los grupos que ilustra lo mencionado.

Figura 54.

Histograma de grupos para característica CM



Nota. El gráfico muestra el histograma de los grupos identificados al utilizar la característica de modelo momentos de color. Elaboración propia, realizado con Python.

Se pueden interpretar los grupos creados por el algoritmo como segmentaciones muy específicas de las anomalías, y la segmentación de las imágenes no anómalas en el *cluster* 0.

Figura 55.

Anomalías agrupadas en cluster 3 con color moments

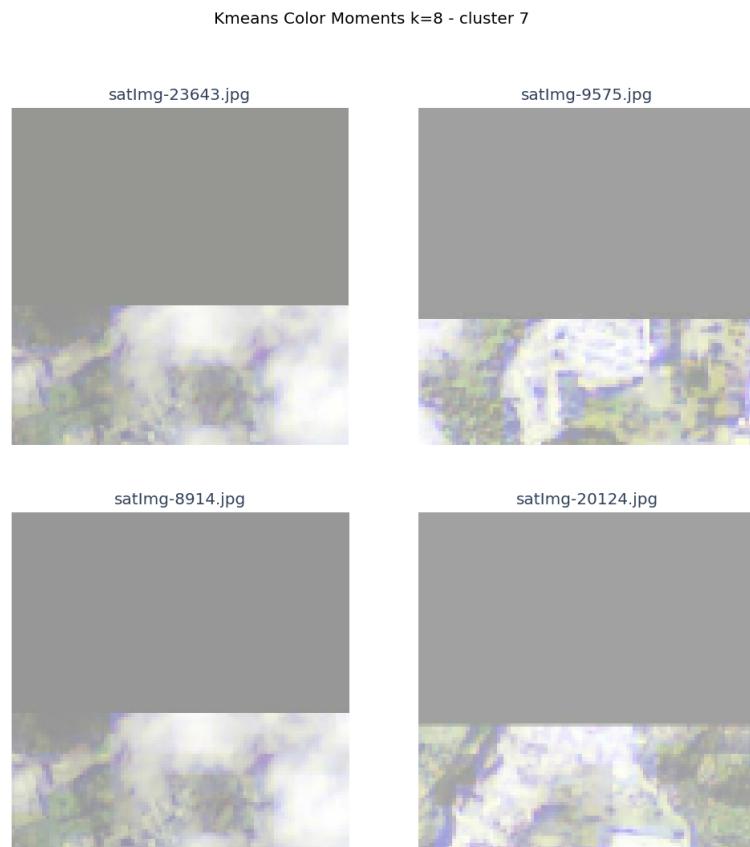


Nota. La figura muestra imágenes agrupadas en el *cluster 3* con la característica de momentos de color. Elaboración propia, realizado con Python.

Se aprecia en la imagen anterior que las características que corresponden a un color fijo y que no contienen información satelital fue segmentada junta, también que los tonos del color fijo son claros.

Figura 56.

Anomalías agrupadas en cluster 7 con color moments



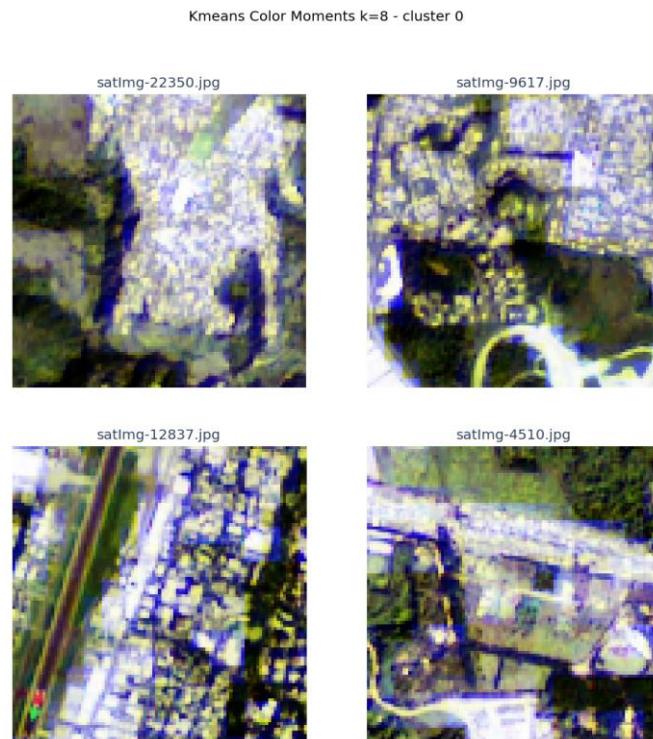
Nota. La figura muestra imágenes agrupadas en el *cluster 7* con la característica de momentos de color. Elaboración propia, realizado con Python.

En la imagen anterior se segmentan imágenes que tienen bloques de un color fijo pero que mantienen una parte de la imagen satelital. La similitud en las imágenes es significativa.

En cuanto a las imágenes que no son anomalías, todas fueron agrupadas juntas, lo que puede ser justificante de los valores tan altos de la silueta en las primeras iteraciones del algoritmo.

Figura 57.

Imágenes no anómalas en cluster 0 con color moments

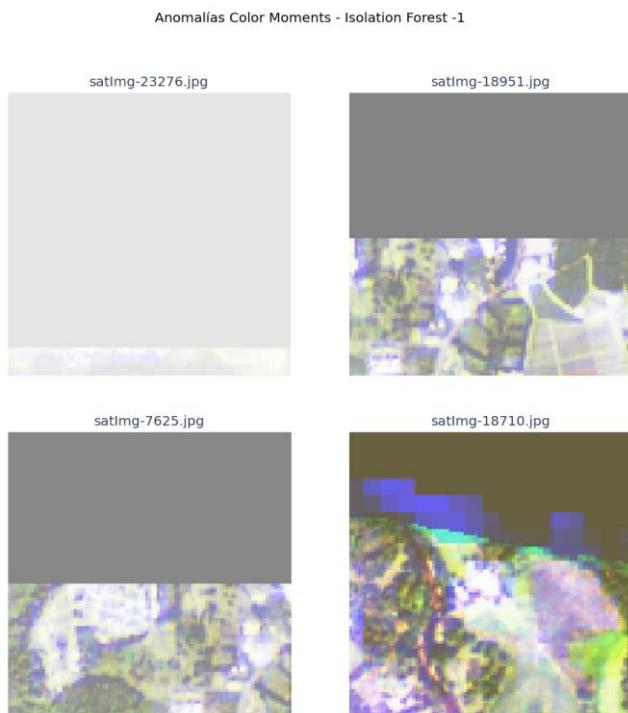


Nota. La figura muestra imágenes no anómalas agrupadas en el *cluster 0* con la característica de momentos de color. Elaboración propia, realizado con Python.

A pesar de que no es lo esperado, la característica Momentos de Color ha demostrado tener la capacidad de representar las imágenes de una forma ligera e informativa a la vez, por tal motivo se procedió a entrenar un modelo de detección de anomalías utilizando la característica *color moments*. El resultado fue exitoso y se muestra a continuación.

Figura 58.

Anomalías isolation forest con color moments



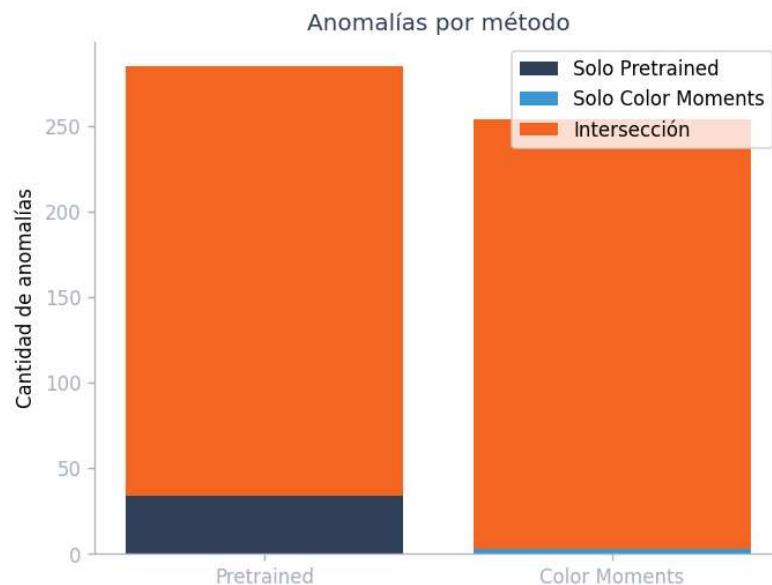
Nota. La figura muestra las anomalías detectadas al utilizar la característica momentos de color al utilizar el algoritmo Isolation Forest. Elaboración propia, realizado con Python.

Un último ejercicio fue el de eliminar las fotos anómalas y volver a entrenar Kmeans con la característica *color moments*, a pesar de que mejoraron las agrupaciones, no hubo un acercamiento real a la calidad obtenida con la característica pre-entrenada.

Para finalizar se comparan las anomalías encontradas por el *cluster 2* del modelo entrenado con pre-entrenada contra las anomalías encontradas por isolation forest con la característica *color moments*. Se decide que el *cluster 2* será el indicativo de anomalía en imágenes, luego de analizar la siguiente gráfica.

Figura 59.

Anomalías isolation forest con color moments



Nota. El gráfico muestra las coincidencias y discrepancias en la detección de anomalías con las características *pretrained* y *color moments*. Elaboración propia, realizado con Python.

4.1.2. Ingeniería de característica datos tabulados

Los conjuntos de datos que se encuentran en el Mercado de Datos han pasado por varias etapas previas, desde transformación hasta limpieza, así como imputaciones basadas en archivos externos. Ahora se trabajará estrictamente con los datos existentes en el *data mart*, de modo que se puedan crear las vistas que permitirán entrenar los modelos.

Se realiza una tarea inicial de imputación de datos faltantes, luego una transformación de las características que lo ameriten y por último una exploración de los datos y selección de características para formar las distintas vistas.

4.1.2.1. Imputación de coordenadas

La tarea en este punto es menos ardua de lo que podría esperarse gracias principalmente al trabajo previo de depuración del conjunto de datos. Ese trabajo anterior hace que el único conjunto de datos con valores faltantes en los campos de la ubicación geográfica sean los *mesh clients*.

La literatura refiere imputar la media como una forma de abordar el problema, en este caso parece no ser la mejor opción, puesto que la media de las coordenadas de todo el conjunto de datos estará en un lugar incorrecto debido a la gran extensión de la red estudiada.

La opción elegida es utilizar una media condicional, de modo que se tomen en cuenta para el cálculo de la media solo aquellas coordenadas que correspondan a un vecino del MC en cuestión. Estos vecinos corresponden a los primeros saltos de la característica *Path*, presente en el conjunto de datos de MC y MR.

Figura 60.

Imputación de coordenadas basada en saltos



Nota. La figura muestra la estrategia de imputación de coordenadas basada en saltos. Elaboración propia, realizado con Python.

La figura anterior muestra el valor de la imputación para un MC particular, de la totalidad de equipos únicamente se calcula la media con aquellos que se registraron como primer salto en el transcurso de los 84 días de toma de datos. A mayor cantidad de ocurrencias de la vecindad, mayor peso tienen las coordenadas en el resultado final de la imputación.

Para las filas que no tenían información de coordenadas ni del salto para obtener una referencia, se decidió eliminar los registros luego de realizar un conteo y determinar que la cantidad de MC a eliminar del conjunto de datos fue mínima.

4.1.2.2. Imputación de *Path*, *Layer* y *CollectorNm*

Al hacer una revisión de los datos se encontró ciertas inconsistencias con la característica *Path*. Esta característica indica el camino que un equipo de la red de comunicaciones utiliza para enviar sus paquetes hasta un MG. El valor en el conjunto de datos es el de una cadena de caracteres que contiene los serialNumber de cada uno de los equipos que conforman el camino. Es necesario recordar que la arquitectura de una red *mesh* se vale de esta capacidad de saltos intermedios para expandir su alcance y crear redundancia.

Una inconsistencia es que hay registros en los conjuntos de datos que no tienen información de *Path*. La otra es que hay valores de *Path* identificados que no cumplen con la estructura esperada para esta característica, específicamente, para una parte de los registros sucedía que el número de serie más a la derecha no correspondía con el del equipo en cuestión, sino que el primer salto. Por lo tanto, esta fue la primera validación necesaria. Se convirtió la cadena a una lista y se agregó como último elemento el número de serie de cada fila, en los casos necesarios.

Luego se abordó la imputación de los valores vacíos. Tanto el *Path*, como la capa o el nombre del MG asociado se resolvieron por criterios de cercanía geográfica. La siguiente gráfica ilustra la idea tras la imputación de estas variables.

Figura 61.

Imputación de datos basada en saltos



Nota. La figura muestra la estrategia de imputación de *Path*, *Layer* y *CollectorNm* basada en cercanía. Elaboración propia, realizado con Python.

Para cada uno de los equipos que carecían de un valor en la característica *Path*, se calculó la distancia sobre la superficie de la tierra con respecto al resto de equipos en la red, de modo que el nuevo valor imputado corresponde al valor de *Path* para este equipo más cercano, con el número de serie del equipo sin *Path* agregado al final de este.

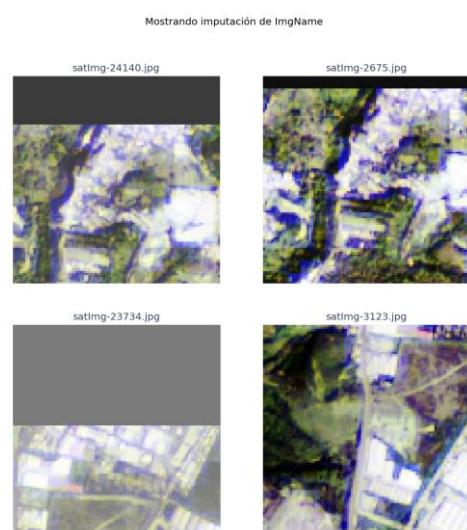
La ventaja de imputar primero *Path* es que esta característica contenía la información para imputar *Layer* y *collectorNm*. El tamaño de *Layer* es la longitud de *Path* -1. El *collectorNm* imputado corresponde al *serialNumber* más a la izquierda de *Path*, dado que toda la comunicación se centraliza en los MG.

4.1.2.3. Imputación de satImg

Dentro de este mismo capítulo se trabajó la identificación de las anomalías en la característica de imagen satelital, que corresponde a la asignación de *cluster 2*. También dentro de la misma imputación de valores faltantes se crearon nuevas coordenadas que no estaban presentes en el conjunto de datos inicial. En ambos casos se realizó una nueva generación de la característica imagen satelital, las nuevas imágenes se reemplazaron y agregaron respectivamente. Al tener el resultado final de todas las imágenes se realizó una agrupación con el modelo Kmeans guardado en el tratamiento específico de esa característica. Se aíslan las imágenes pertenecientes al *cluster 2* dado que son las anomalías. La asignación realizada es, de nuevo, por proximidad geográfica. En la siguiente imagen se ve el resultado de dicha imputación.

Figura 62.

Imputación de imagen satelital



Nota. La figura muestra los resultados de la imputación de la característica imagen satelital basada en cercanía. Elaboración propia, realizado con Python.

4.1.2.4. Depurando columnas

En este punto los conjuntos de datos tienen columnas que serán innecesarias, incluso algunas columnas necesarias que requieren una imputación. La razón de no haber imputado estas columnas es que los valores faltantes bien podrían ser determinantes para la variable dependiente, es decir, la predicción que se quiere hacer sobre la comunicación de determinado equipo. Luego de eliminar las columnas innecesarias, se tienen los conjuntos de datos con las características que se presentan a continuación.

Tabla 7.

Columnas de datos depurados

Característica	MG	MR	MC	Descripción
meterNo	No	Sí	Sí	Identificador de dispositivo.
radio	Sí	Sí	Sí	Identificador de radio en la red mesh.
serialNumber	Sí	Sí	Sí	Identificador de radio en decimal.
CollectorNm	Sí	Sí	Sí	Nombre del MG asociado a un equipo.
status	Sí	Sí	Sí	Estado en el sistema gestor.
latitude	Sí	Sí	Sí	Coordenada geográfica.
longitude	Sí	Sí	Sí	Coordenada geográfica.
LastPacketReceived	Sí	Sí	Sí	Estampa de tiempo de último paquete recibido.
LastStatusChanged	Sí	Sí	Sí	Estampa de tiempo del último cambio de estado.
NoCommunicatingDays	Sí	Sí	Sí	Días sin comunicación.
numberOfNeighbors	Sí	Sí	Sí	Cantidad de vecinos registrados por radio.
fDate	Sí	Sí	Sí	Fecha toma de la muestra.

Continuación de la Tabla 7.

Característica	MG	MR	MC	Descripción
round	Sí	Sí	Sí	Ronda de toma de muestras. De Lunes a Sábado.
fixed_Path	Sí	Sí	Sí	Saltos desde un equipo hasta su MG asociado.
FE-Comments	Sí	Sí	Sí	Comentarios de procesos realizados en ingeniería de características.
Path_length	Sí	Sí	Sí	Longitud de los altos en variable fixed_Path.
ImgName	Sí	Sí	Sí	Nombre de imagen satelital.
cluster	Sí	Sí	Sí	Cluster asignado a imagen satelital.
Layer	Sí	Sí	Sí	Valor de los saltos registrados en la bd.

Nota. La tabla muestra la información de las columnas del conjunto de datos luego de la depuración. Elaboración propia realizado con Word.

4.1.3. Etiquetado de datos

El enfoque escogido para entrenar los modelos es el del aprendizaje supervisado, por lo que es necesario etiquetar los datos con la variable a predecir de modo que el modelo cuente con registros que le permitan ajustar sus parámetros para conseguir que el valor de la etiqueta predicha por el modelo sea lo más cercano posible a la etiqueta verdadera. Esto, por supuesto, no se analiza en el caso de registros particulares sino en conjunto.

La decisión de diseño fue la de utilizar una etiqueta binaria, comunica 1, no comunica 0. Esto dado que los datos no tienen una etiqueta existente para

este propósito. La forma de proceder, haciendo uso del campo NoCommunicatingDays fue estableciendo el umbral de 4 días o menos como Communicating igual a 1, y Communicating igual a 0 para cualquiera cantidad de días sin comunicar mayor a 4. Esto es lo que en la literatura se refiere como una etiqueta natural. La decisión del umbral tampoco fue arbitraria, según el conocimiento del negocio se sabe que, a partir de 4 días sin comunicar, hay información que se pierde irrevocablemente de los equipos.

Como también se contaba con dos características temporales, fDate y round, se consideró necesario hacer una reducción del conjunto de datos al agrupar por la ronda, que como se mencionó en el capítulo 3, corresponde a períodos de una semana. Con la agrupación se utilizó la media para representar varios registros en uno solo. Luego de esta reducción, los conjuntos de datos con valores únicos para las distintas rondas cambiaron como se muestra en la siguiente tabla.

Tabla 8.

Cantidad de registros luego de depuración

Conjunto de datos	Cantidad inicial de registros	Cantidad final de registros
Mesh Gateways	6635	953
Mesh Routers	63202	24103
Mesh Clients	1607261	723807

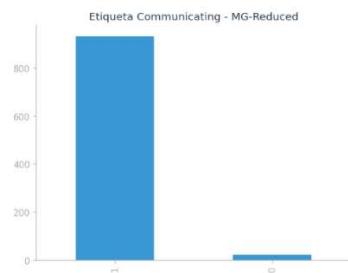
Nota. La tabla muestra la cantidad inicial y final de los registros por tipo de equipo luego de la depuración. Los valores fueron calculados con Python. Elaboración propia realizado con Word.

Al hacer la agrupación y calcular la media, el valor de la etiqueta (Communicating) tuvo algunos valores entre 0 y 1. Para devolver la etiqueta a su condición binaria, se volvió a aplicar un umbral, en este caso 0.9, cuya

justificación está en el objetivo del negocio de mantener niveles aceptables de comunicación, grosso modo, que comuniquen al menos el 90 por ciento de las veces. A continuación, se muestran los histogramas de esta etiqueta para los 3 conjuntos de datos principales.

Figura 63.

Histograma de comunicación MG

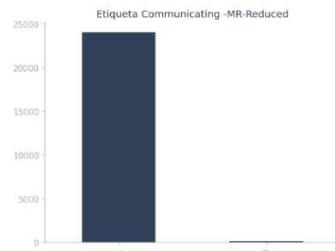


Nota. El gráfico muestra la distribución de los *Mesh Gateways* según su etiqueta de comunicación. Elaboración propia, realizado con Python.

Se aprecia una cantidad pequeña de los registros sin comunicación a los que sí comunican. En el caso específico de los MG esta etiqueta no está para ser predicha, puesto que la comunicación o no de un MG no se da a través de la red mesh, sino que al ser el dispositivo final, su comunicación es IP, ya sea por la red celular 3G o por enlaces de fibra óptica. Sin embargo, esta etiqueta servirá de referencia en los registros de algún otro equipo de la red, MR o MC.

Figura 64.

Histograma de comunicación MR

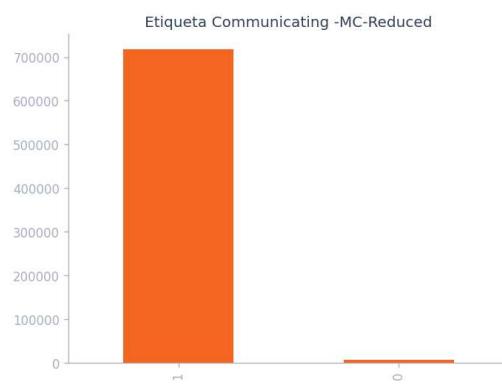


Nota. El gráfico muestra la distribución de los *Mesh Routers* según su etiqueta de comunicación.
Elaboración propia, realizado con Python.

En los MR se aprecia aún una menor proporción de equipos sin comunicar, en este caso la etiqueta sí está para ser predicha, y a la vez será referencia en registros pertenecientes a otro equipo y cuyo vecino sea un MR.

Figura 65.

Histograma de comunicación MC



Nota. El gráfico muestra la distribución de los *Mesh Clients* según su etiqueta de comunicación.
Elaboración propia, realizado con Python.

Es en los *mesh* clientes donde se encuentra el grueso de los datos para el entrenamiento, y como se aprecia, también existe una pequeña porción de equipos sin comunicación. Esto, desde el punto de vista del negocio es deseable, pero desde el punto de vista de conseguir predecir la comunicación exitosa o no, significa tener un conjunto de datos desbalanceado.

Adicional a esta etiqueta se trabajó el resultado de las distintas rondas del comando ping enviado a los equipos, de modo que esta etiqueta complementará a la anterior o simplemente la reemplazará. Luego de agrupar todos los resultados de los comandos y tratar específicamente el comando ping, se determinó lo impráctico de su uso dada la considerable falta de datos al agregar la información del ping a los conjuntos de datos principales. La razón principal de esto puede deberse a fallas en el sistema gestor, a fallas intermitentes en la comunicación y que, en el caso de ciertos estados de equipos en el sistema, este no envía el comando solicitado por el usuario.

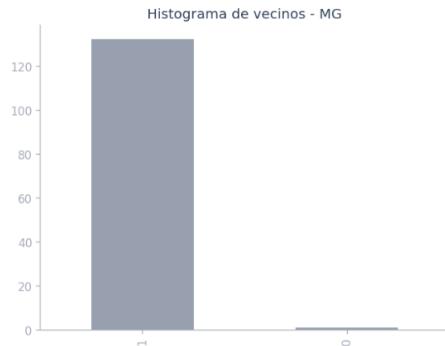
4.1.3.1. Imputación de `numberOfNeighbors`

Se consideró necesario realizar la imputación de la característica `numberOfNeighbors` luego del proceso de etiquetado de datos. La razón es que, parece razonable pensar que un equipo que no comunica tendrá menos equipos registrados como vecinos, más aún, la causa del vacío en esta característica puede ser consecuencia directa de la no comunicación.

Se confirma a continuación el fundamento mencionado, al visualizar los histogramas para la comunicación de los registros con carencia de la característica en cuestión.

Figura 66.

Histograma de comunicación MG sin numberOfNeighbors



Nota. El gráfico muestra la distribución de los *Mesh Gateways* sin registro de *numberOfNeighbors* según la etiqueta de comunicación. Elaboración propia, realizado con Python.

La tendencia parece continuar en el caso de los registros que no tienen el número de vecinos desde la base de datos. Lo que puede significar que no existe una relación directa entre la ausencia del campo y la comunicación o no del equipo, que como ya se dijo, en el caso de los MG no depende de la red Mesh.

Figura 67.

Histograma de comunicación MR sin numberOfNeighbors

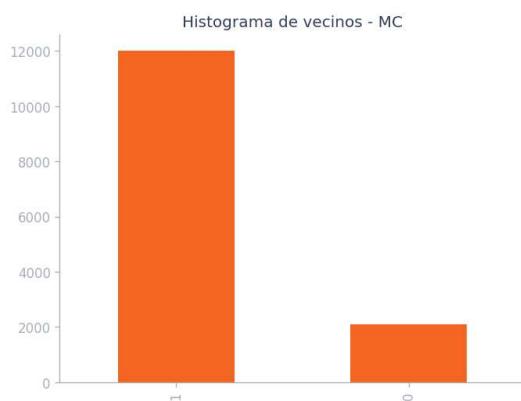


Nota. El gráfico muestra la distribución de los *Mesh Routers* sin registro de *numberOfNeighbors* según la etiqueta de comunicación. Elaboración propia, realizado con Python.

En este caso, se comprueba que no existen registros que no comuniquen sin el valor de la característica `numberOfNeighbors`. Siendo que MR es el conjunto de datos con menor proporción de equipos sin comunicación, no parece significativa tal situación.

Figura 68.

Histograma de comunicación MC sin `numberOfNeighbors`



Nota. El gráfico muestra la distribución de los *Mesh Clients* sin registro de `numberOfNeighbors` según la etiqueta de comunicación. Elaboración propia, realizado con Python.

Es en la gráfica anterior donde se valida hasta cierto punto la idea preconcebida, de que la ausencia de información en varios campos, y específicamente en `numberOfNeighbors`, tiene una relación con la etiqueta de comunicación o no, o viceversa. Esto, por supuesto no significa una causalidad, pero sí se consideró motivo suficiente para la siguiente estrategia de imputación.

Siendo que la etiqueta de comunicación para MG es distinta, que para MR no existen registros sin comunicación para la ausencia de la cantidad de vecinos, y a que el grueso de los datos corresponde a MC. Se utilizaron 3 modelos distintos para la imputación.

Primero se consideró imputar la media, según el conjunto de datos y según la etiqueta de comunicación. Al revisar el principal subconjunto de interés (equipos que no comunican), y generar su gráfica de distribución, se confirmó que la media no era el valor adecuado de imputación.

Figura 69.

Distribución de numberOfNeighbors para no comunica

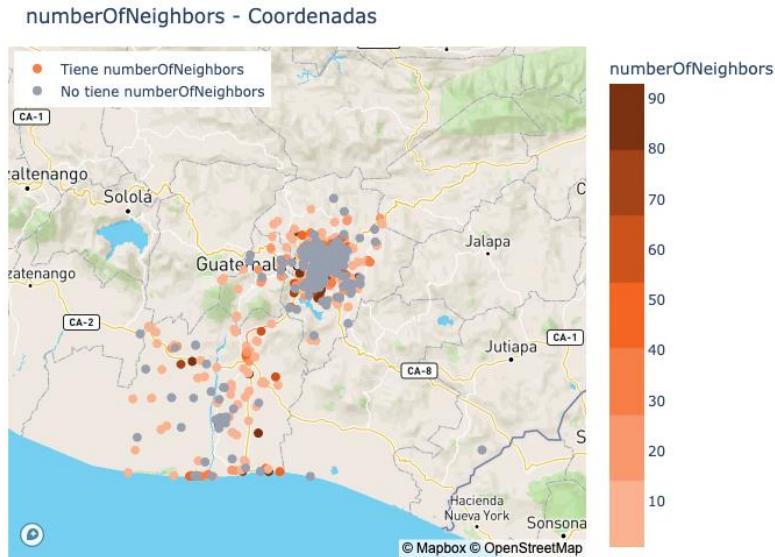


Nota. El gráfico muestra la distribución de `numberOfNeighbors` para los *Mesh Clients* que no comunican. Elaboración propia, realizado con Python.

La visualización anterior arroja luz sobre que existen distintos grupos en los datos con mayor o menor cantidad de vecinos y con mayor o menor proporción del total de los registros. Visualizarlo de la siguiente manera ayudó a asimilar mejor la situación.

Figura 70.

Mapa de `numberOfNeighbors` registros no comunican



Nota. La figura contiene el mapa de la cantidad de vecinos, así como el indicativo de cantidad de vecinos faltantes para los equipos con etiqueta de no comunicación. Elaboración propia, realizado con Python.

Se vislumbra la correspondencia entre la distribución de la cantidad de vecinos y el mapa anterior. La gran mayoría de los equipos que no comunican tienen una cantidad reducida de vecinos, mientras que los que tienen un gran número de vecinos son registros menos frecuentes. En cuanto a los valores en gris, equipos sin registro de vecinos, se ve que están distribuidos a lo largo de toda el área de la red, en correspondencia con la densidad de equipos en determinados territorios.

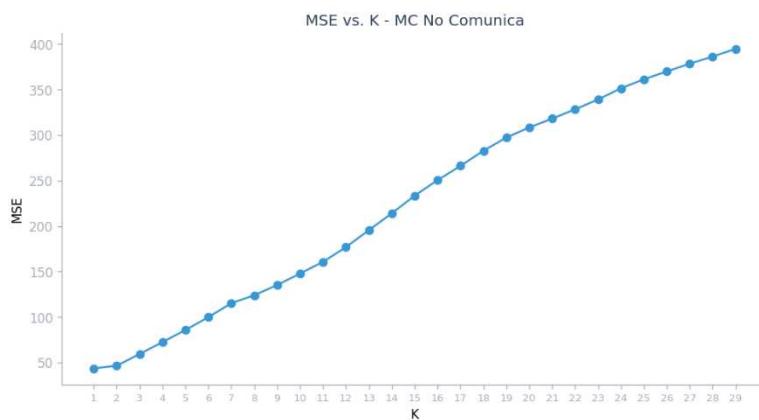
Es notoria e intuitiva la importancia de la información geográfica a través de todo el proyecto. Se decide utilizar el algoritmo KNN (K vecinos más cercanos), que permita realizar la imputación que mejor se acople los datos al

ser validada con un conjunto de datos de prueba. En los siguientes casos se procederá de la siguiente manera: se seleccionará un conjunto de datos y separarán los valores con registro de vecino y los que no lo tienen, luego se separará el conjunto de datos con registro en conjuntos de entrenamiento y testeo, se iterará el valor de K en el rango de 1 a 30 y se entrenarán los modelos, se calculará la métrica para el conjunto de datos de testeo y se elegirá el modelo con menor MSE (error cuadrático promedio).

El conjunto de datos de MC sin comunicación tuvo los resultados siguientes.

Figura 71.

KNN con MC sin comunicación

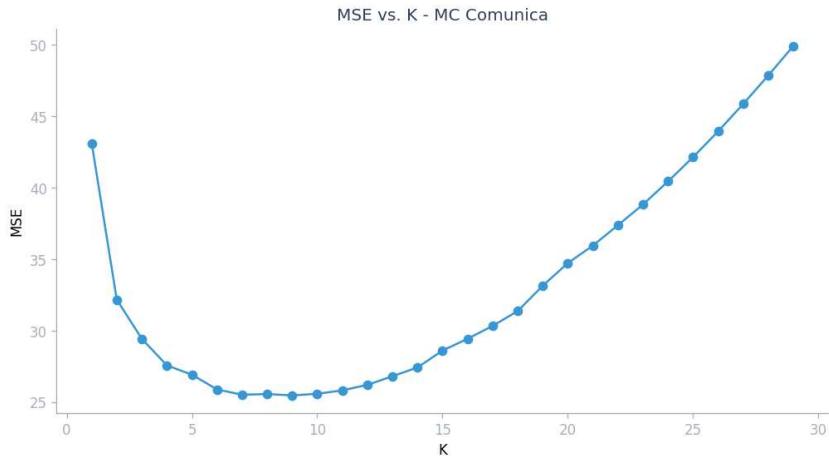


Nota. El gráfico muestra MSE vs K para MC sin comunicación. Elaboración propia, realizado con Python.

El mejor valor para imputar de `numberOfNeighbors`, según la métrica elegida es el que corresponde al vecino más cercanos. Para el caso de los MC que sí comunica, este valor de K varía.

Figura 72.

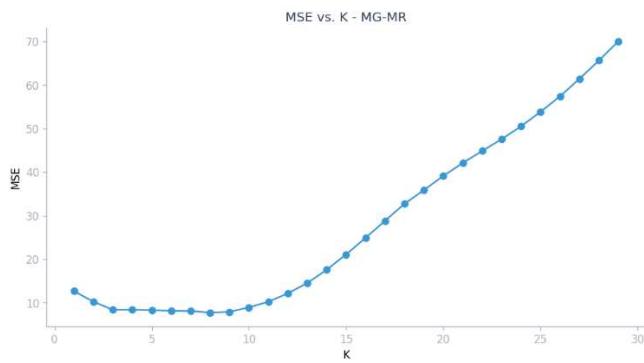
KNN con MC con comunicación



Nota. El gráfico muestra MSE vs K para *Mesh Clients* con comunicación. Elaboración propia, realizado con Python.

En este caso el valor ideal de K fue 9, de modo que el valor imputado fue el promedio de la cantidad de vecinos de los 9 registros más cercanos. Para los MG y MR, se trabajó un solo modelo y no se diferenció según la etiqueta Communicating.

Figura 73.
KNN con MG-MR



Nota. El gráfico muestra MSE vs K para *mesh gateways* y *mesh routers*. Elaboración propia, realizado con Python.

Se aprecia que la tendencia es similar al caso de los MC con comunicación, el mejor valor de K fue 8. Luego de esta imputación se completan todos los campos faltantes en el conjunto de datos de entrenamiento.

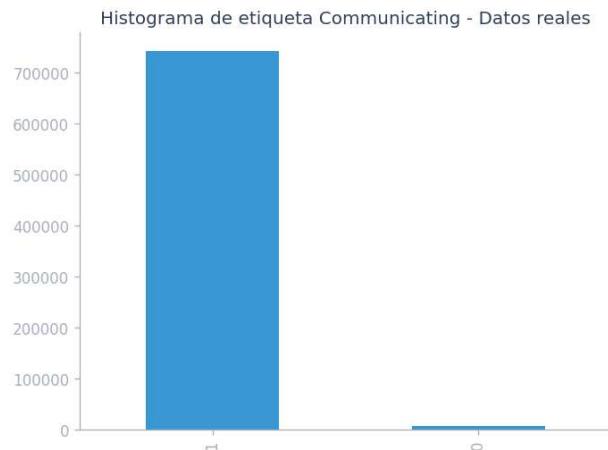
Se hace la aclaración de que el interés de mostrar las gráficas anteriores es para demostrar cierta diferencia según los conjuntos de datos utilizados, sin embargo, no se deben comparar los valores de métricas entre distintos conjuntos de datos.

4.1.4. Aumentación de datos

El desbalance de datos es un conocido antagonista a la hora de entrenar modelos de ML y DL. Existen algoritmos más o menos sensibles al desbalance de clases. En este caso particular, las cotas altas en el nivel de comunicación de la red mesh repercuten en el desbalance significativo que se muestra en la siguiente gráfica.

Figura 74.

Conjunto de datos final con datos reales



Nota. El gráfico muestra la distribución de la etiqueta de comunicación en el conjunto de datos final. Elaboración propia, realizado con Python.

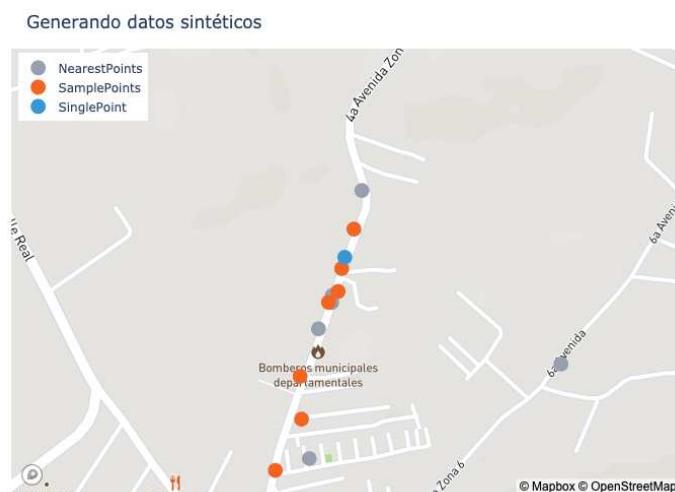
La gráfica anterior es obtenida sobre el conjunto final que se utilizará para el entrenamiento, el cual concatena los datos de MC y MR, que en última instancia son los dispositivos que se comunican a través de la red *mesh*. Los MG, que se comunican en un formato distinto servirán de referencia, pero como tal no se buscará predecir su estado de comunicación.

La idea básica para la creación de los datos sintéticos, de la categoría —no comunica— es que, si un equipo no es capaz de establecer un enlace efectivo con un vecino X, tampoco lo será con algún Y o Z. Es decir, dado que todos y cada uno de los registros en el conjunto de datos está vinculado con su correspondiente vecino, y que para una misma ronda existen más de un registro para un solo equipo de red, se considera viable crear registros sintéticos que correspondan a un equipo que no comunica, con X, con Y y con Z, como una forma de aumentar la cantidad de datos de la etiqueta minoritaria.

Como decisión de diseño y por la costumbre de vida de utilizar el sistema numérico de base 10, se decide que para cada uno de los equipos que nunca comunicó en el tiempo que comprende la muestra, se generarán 10 datos sintéticos con asociaciones a distintos vecinos según sea la ronda. La siguiente imagen ayuda a visualizar tal ideal.

Figura 75.

Generación de datos sintéticos

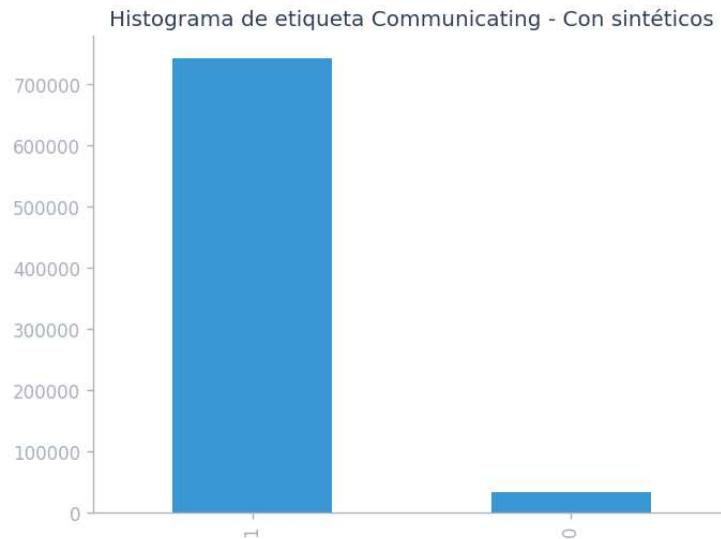


Nota. La figura muestra la estrategia de generación de datos sintéticos. Elaboración propia, realizado con Python.

Para cada punto individual y durante cada una de las rondas, se seleccionaron los 50 equipos más cercanos geográficamente, luego se creó una muestra aleatoria de 10 de estos equipos, los cuales constituyeron cada uno, un registro nuevo en el conjunto de datos. Por supuesto, agregó el comentario —a FE-Comments— indicando el carácter sintético de los datos. Luego de la generación de los datos sintéticos, la proporción de la etiqueta —Communicating— quedó de la siguiente manera.

Figura 76.

Conjunto de datos final con datos sintéticos agregados



Nota. El gráfico muestra la distribución de la etiqueta de comunicación en el conjunto que contiene los datos sintéticos. Elaboración propia, realizado con Python.

Se agregaron un total de 25960 datos sintéticos, correspondientes a los 232 equipos de red que nunca comunicaron en los 84 días que duró la toma de datos. No se generaron datos sintéticos para equipos con etiqueta de no comunicación, pero que en algún momento de la toma de datos tienen registro de comunicación, para evitar confusiones en los algoritmos por añadir posibles etiquetas negativas que resultaran ser positivas en distintas instancias temporales.

4.1.5. Fresnel masivo

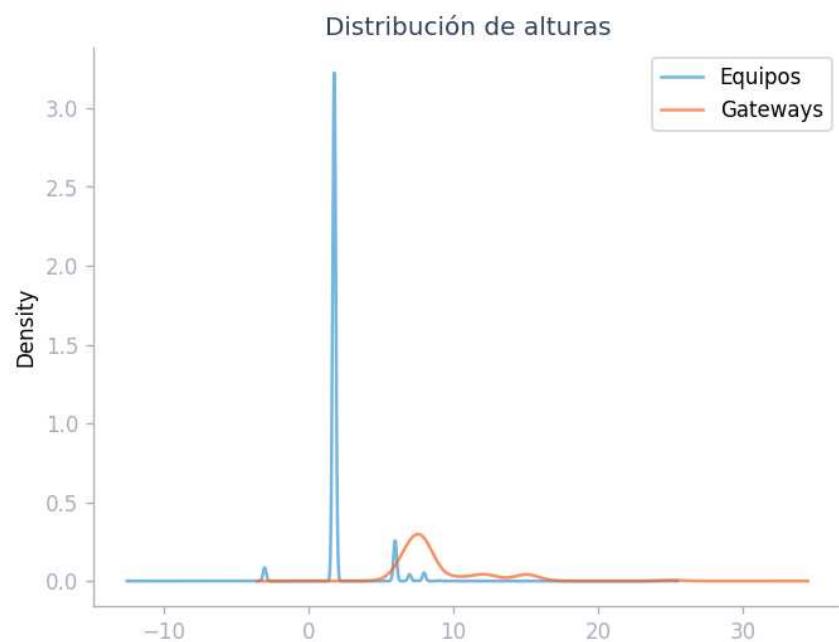
Como se mencionó en el capítulo 3, decidió trabajarse la característica Fresnel para agregar un factor conocido en el mundo de las telecomunicaciones, el hecho de que las ondas electromagnéticas viajen por un medio que no es ideal

implica que puede haber elementos entre las dos antenas que conforman un radioenlace que impidan que la comunicación sea efectiva.

En la etapa anterior de la característica Fresnel lo que se hizo fue calcular los posibles radioenlaces y obtener la altimetría intermedia al calcular un camino entre ambos puntos. Sin embargo, no se calculó la característica completa debido a que no se tenían las alturas, ni se habían depurado los radioenlaces, mucho menos imputado la información faltante. Es en este punto donde se puede calcular la característica sobre todo el conjunto de datos.

Figura 77.

Distribución de alturas para equipos de la red mesh



Nota. El gráfico muestra la distribución de las alturas para todos los equipos de la red mesh.
Elaboración propia, realizado con Python.

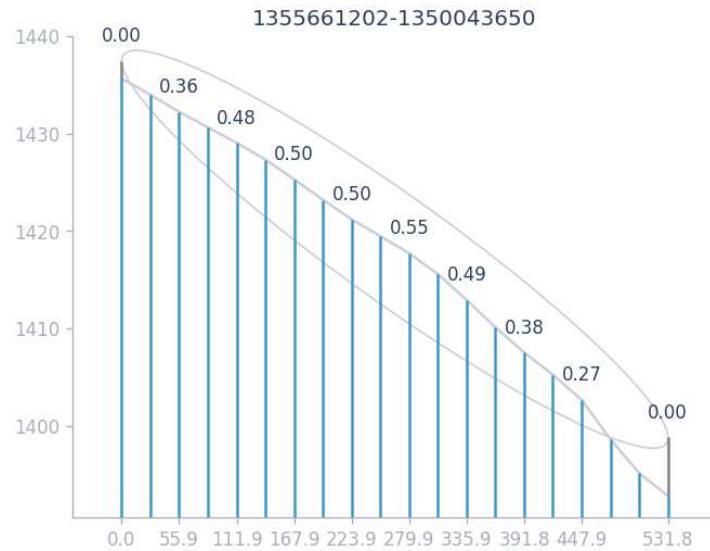
La imagen anterior muestra el resultado de agregar la altura, que es uno de los factores necesarios para el cálculo de la característica Fresnel. Se aprecia una diferencia significativa entre los equipos y los gateways. Esto es justificable desde que los equipos de red (MR y MG) están instalados en altura, ya que su función es la de crear las principales rutas de comunicación. Los valores mayores en la distribución de los equipos son debido a los MR, y los valores menores que 0 corresponden a equipos instalados en sótano.

Luego de las etapas de imputación de datos faltantes y generación de datos sintéticos se agregaron radioenlaces adicionales a los que se tuvieron en cuenta en la generación de la base de datos de altimetría, por lo que fue necesario generar tales altimetrías directamente de la API en el momento de trabajar la generación masiva de Fresnel.

El valor elegido para la cantidad de puntos entre el equipo en cuestión y su vecino fue de 20 puntos equidistantes, para estos se calcularon las 20 alturas correspondientes, adicional se indicó el valor de la frecuencia al momento de generar la característica. El resultado de esto fueron 20 dimensiones más en el conjunto de datos. Para confirmar la validez de los datos se hicieron revisiones al azar con gráficas generadas a partir de los datos almacenados en el conjunto de datos de equipos.

Figura 78.

Comprobación de característica Fresnel con obstrucciones

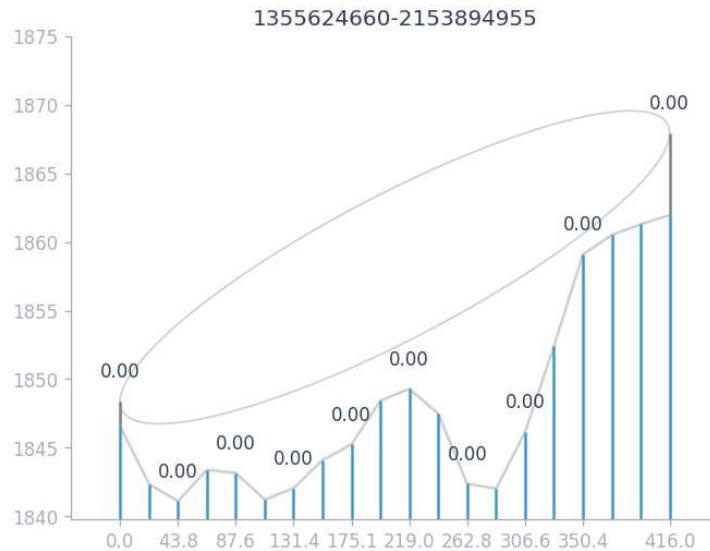


Nota. El gráfico muestra la característica de Fresnel para un ejemplo con obstrucción. La imagen fue generada a partir de la característica generada de manera masiva. Elaboración propia, realizado con Python.

Se visualiza en la imagen anterior que para altimetrías que se interponen entre la primera elipse de Fresnel, se calculó de manera correcta la característica.

Figura 79.

Comprobación de característica Fresnel sin obstrucciones



Nota. El gráfico muestra la característica de Fresnel para un ejemplo sin obstrucción. La imagen fue generada a partir de la característica generada de manera masiva. Elaboración propia, realizado con Python.

En el caso anterior, todos los valores de la característica Fresnel están en 0, debido a que no hay ninguna obstrucción geográfica en el espacio intermedio entre los dos puntos. Es posible, por supuesto, que existan obstrucciones que estén sobre la superficie de la tierra, para tal caso se espera que la característica de imagen satelital arroje información útil, es en este punto también donde se agrega la información de la característica de momentos de color, para tener una representación de las imágenes más allá del *cluster*.

Otra característica que se agregó fue el indicativo de antena, que básicamente aborda una situación del mundo real, que definitivamente es condicionante del estado de comunicación. Para algunos equipos en interior o

especialmente para sótanos, es posible conseguir la comunicación efectiva al instalar una antena adicional, que de manera pasiva traslada la señal de radiofrecuencia fuera de la estructura que la limita. Se considera que sin dicho indicativo, equipos que en condiciones normales no comunicarían, serían catalogados por el algoritmo como Sí comunica sin tomar en cuenta que requirieron de una modificación, en este caso Antena = 1, o Antena = 2, según corresponda al tipo de antena instalada en el terreno.

Una cuestión un poco más sutil respecto al conjunto de datos para el entrenamiento de los modelos es que se puede tener características que afecten el desempeño de los modelos, o algunas que lo mejoren a costa de información redundante con la etiqueta y que significaría una buena métrica más no una solución real al problema.

Por lo anterior, se agregaron referencias del mejor vecino (o primer salto) y del correspondiente MG para cada equipo, esperando con esto que el modelo se base más en el contexto de la red en sus cercanías para predecir el estado de la comunicación, que en variables existentes en el conjunto de datos que en una aplicación real no existirían para un equipo completamente nuevo. Un ejemplo claro de lo anterior es la característica noCommunicatingDays, para la cual bastaría un filtro con umbral en 4, para categorizar si un equipo comunica o no.

La selección final de las características se presentará en los siguientes incisos, donde se divide el conjunto de datos de equipos en conjuntos de datos de entrenamiento y testeo, se seleccionan también las características y se acondicionan para el siguiente paso inmediato, el entrenamiento de los algoritmos.

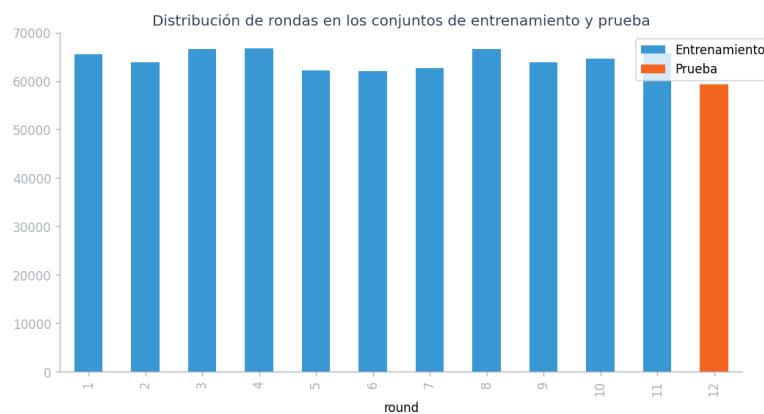
4.1.6. División de datos

Para el entrenamiento de los modelos se usará una porción del conjunto de datos de que se dispone, la otra fracción será para la evaluación de los modelos. Básicamente, la idea es evaluar los modelos con datos desconocidos, evidentemente en una aplicación real, se generarán datos que estarán fuera del conjunto de entrenamiento de los modelos. Esta práctica permite, hasta cierto punto, generalizar el alcance del modelo y evitar una simple adaptación a los datos para minimizar la función de pérdida, lo que se conoce como sobreajuste.

Luego de algunos intentos e indagación en la literatura, se determinó que, dada la existencia de una variable temporal, ronda ('round'), la decisión atinada era evitar un traslape temporal entre el conjunto de datos y el de entrenamiento. Por tal motivo, se decidió apartar la ronda 12, para el conjunto de datos de prueba. A continuación, se presenta gráficamente esta selección.

Figura 80.

Primera división de conjunto de datos



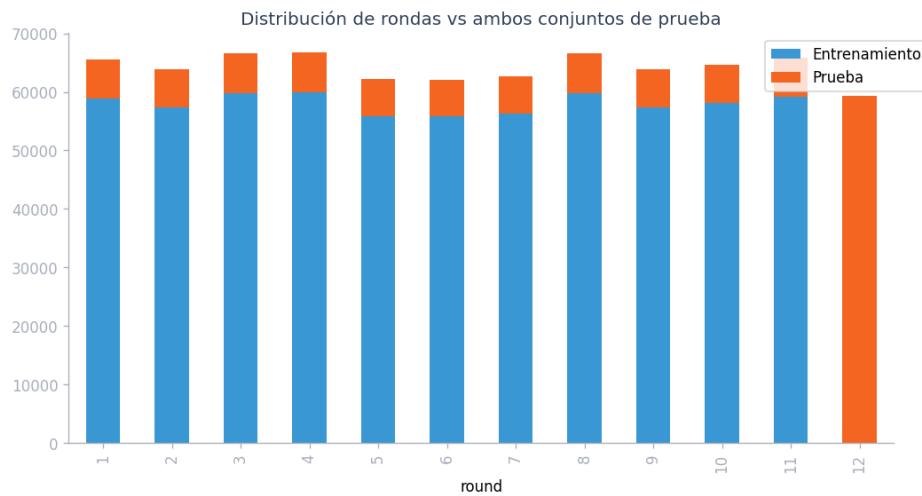
Nota. El gráfico muestra la separación de los datos en entrenamiento y prueba para el problema 1. Elaboración propia, realizado con Python.

La situación ahora es que existe otra fuente de traslape, y es debido a la consistencia de la red a través del tiempo. Un equipo es instalado y en la mayoría de los casos dura años en el mismo lugar, siendo que la muestra solo abarca 84 días, la mayoría de los equipos tendrán registro en la mayoría de las rondas. Esto se apreció en las etapas de limpieza, al hacer conteos por ronda, y obtener demasiados números 84.

Esta situación contraviene al objetivo del modelo de determinar, dada la red existente, su cobertura. Saber si un equipo ya instalado comunica o no, es baladí. Lo interesante es acertar con equipos que aún no existen en la red. Por tal motivo, se retira del conjunto de datos el 10 % de los equipos, para todas las rondas.

Figura 81.

Segunda división de conjunto de datos

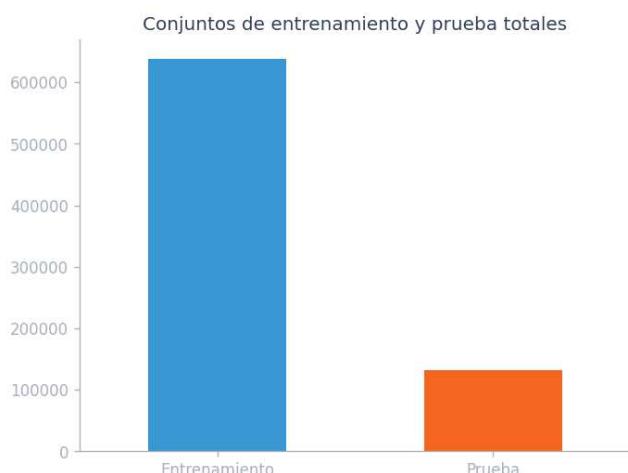


Nota. El gráfico muestra la separación de los datos en entrenamiento y prueba para el problema 2. Elaboración propia, realizado con Python.

Cabe mencionar, que en este conjunto de datos un 10 % de los equipos no representa un 10 % de los registros, esto debido a que para un equipo puede haber varias entradas en una ronda, según sus referencias de equipos asociados. Por otro lado, se indagó en la literatura y diversas fuentes sobre el riesgo de una fuga de datos entre dimensiones, al existir estos equipos retirados, en el conjunto de datos de prueba en forma de vecinos. Para los modelos específicos que se utilizarán, y según las particularidades de los datos, parece no haber un riesgo mayor. Luego de estas divisiones, el porcentaje de datos correspondiente al conjunto de datos de prueba es del 17.12 %.

Figura 82.

División final del conjunto de datos original



Nota. El gráfico muestra la separación general de los datos en entrenamiento y prueba.
Elaboración propia, realizado con Python.

Una vez se han dividido los datos, se puede proceder a la última etapa de transformaciones de características, ahora sí, evitando la fuga de datos entre el conjunto de datos de entrenamiento y el de prueba.

4.1.7. Transformación de características

Los datos que son aptos para entrenar los modelos, como ya se ha dicho, deben ser numéricos, más allá de eso, la escala de estos también resulta de interés puesto que una diferencia significativa entre la escala de distintas variables podría resaltar de manera inadecuada la importancia de unas en detrimento de las otras. Al finalizar este apartado los datos podrán ser procesados por los modelos tanto de *machine learning* como de *deep learning*.

4.1.7.1. Hash Encoding

Para información sensible como lo son los números de serie, se utilizó una codificación basada en un algoritmo Hash, específicamente el algoritmo MD5. Tanto en las características con información de serialNumber, como en los registros de la variable fixed_Path, se aplicó el algoritmo mencionado. En el caso específico de las variables referidas, no tendrán un uso directo en los algoritmos, pero serán necesarias para referenciar los equipos en algún momento del entrenamiento, sin formar parte de las características de entrenamiento.

Como resultado de esta codificación, se mantiene la unicidad en los registros correspondientes a un mismo número de serie, a la vez que se oculta la información real. Cabe mencionar, que este algoritmo no puede ser revertido, más si comprometido haciendo uso de diccionarios Hash y encontrando las colisiones. Como los datos han sido aislados y en los cuadernos sólo se muestran los head de los conjuntos de datos, esto no representa un riesgo de seguridad.

Tabla 9.*Ejemplo de característica codificada con MD5*

serialNumber	fixed_Path
c3ff7d43802c4fcdbcdb4a85344479600	['b1615844d5c5c1d8fcb24ad56a419061', '0c0446825d5e7b506a7fa5aa6ca06448', 'c3ff7d43802c4fcdbcdb4a85344479600']
517e1ad54e07d165c0a95d981fa41300	['b46523072cb39eef309658ad5ce4710d', 'e134455abbc9733a7846f9926fb74931', '517e1ad54e07d165c0a95d981fa41300']
22ee3f1fd01582323867c03b0752829d	['82f518c112aa8141b0282b63ecaaa529', '22ee3f1fd01582323867c03b0752829d']
977728e73369da095cec443ff0d6f821	['fc3ac975c3a24f7413ff8d6788bb920f', '7e2df52ca8e8f9f2c4d8660bda972263', 'c26585d8ffd0dd366af0b39d1db89603', '977728e73369da095cec443ff0d6f821']
991dc605cf605397bf1bd29adc79afa	['1626d8bebb5aa3c04ebd0d509f3bb65e', 'a06ef933611c43236b403774e822f94a', '991dc605cf605397bf1bd29adc79afa']
29de7996f128da1ad079086ae77c777f	['b570a36ae18d7fcf5a9d634cdf153d5a', '75bfb368c412ac021e2de4ba3ae761dc', '76ceea555b5211f262e3985ba957c2be', '29de7996f128da1ad079086ae77c777f']
7cd391c52f5e6554e4fe0812cb2ed565	['c19a7ee65fb33affd848349c78a91a5e', '56de3244cc9539d8892d869601fb91a3', '7cd391c52f5e6554e4fe0812cb2ed565']
e50320878b3b5f60cd2b10f9bdb56e52	['82f518c112aa8141b0282b63ecaaa529', 'eb07f902940c2e2364652bb4848d705b', 'e50320878b3b5f60cd2b10f9bdb56e52']

Continuación de la Tabla 9.

serialNumber	fixed_Path
041106e95bd9ba02a42e717db812313	['cecc13fa9ae872a0c3a664160d3e6003', '0facd81d4ca3c3b15a15f16c30de76d2', '73f38d249ac9255fca05bd6142de51cb', '041106e95bd9ba02a42e717db8123130']
2723dca765b4a1e8aff6d675746812c1	['bdfe2326a4192645b711059c520fb2e4', '760cb96f47725a903420290c1a15a7d0', '2723dca765b4a1e8aff6d675746812c1']

Nota. La tabla muestra las características codificadas con un algoritmo Hash. Elaboración propia, realizado con Python.

4.1.7.2. Binary Encoding

De todas las variables categóricas, aquellas relacionadas con el Mesh Gateway resultaron ser las de mayor cantidad de valores posibles, rondando los 80 valores únicos. Ciertos métodos usualmente utilizados para variables categóricas aumentaban considerablemente la dimensionalidad del conjunto de datos.

Por lo anterior se tomó la decisión de utilizar Binary Encoding. Esta codificación consiste en realizar una codificación ordinal de los registros en las características de interés, en primera instancia, para luego convertir estos valores ordinales a números de base dos y por último, convertir en columna cada uno de los bits del número binario. Con esto se asegura la unicidad a la vez que no se introduce el sesgo de establecer una relación ordinal entre los datos.

Tabla 10.*Ejemplo de característica codificada con Binary Encoding*

Collector Nm_0	Collector Nm_1	Collector Nm_2	Collector Nm_3	Collector Nm_4	Collector Nm_5	Collector Nm_6
0	0	1	0	1	1	1
0	1	1	0	1	1	0
0	1	1	1	1	0	1
1	0	0	1	1	1	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
1	0	1	0	0	0	0
0	1	1	1	1	0	1
0	1	1	1	1	1	0
0	1	1	0	0	0	1

Nota. La tabla muestra las características codificadas con Binary Encoding. Elaboración propia, realizado con Python.

4.1.7.3. One Hot Encoding

Esta técnica es una de las más conocidas para tratar variables categóricas, consiste en crear una columna por cada uno de los valores únicos que puede tener una característica, luego, para cada registro se establece el valor de 1 en su correspondiente columna y para el resto de las columnas el valor de cero. De este modo se garantiza convertir la variable categórica, que como texto no informa al algoritmo, a un vector de ceros y unos.

Una desventaja de esta característica se da a medida que aumenta el número de posibles valores únicos, ya que la dimensionalidad del conjunto de datos crece hasta puntos inmanejables y que podrían perjudicar los algoritmos.

Por tal motivo, las características seleccionadas para esta codificación fueron validadas al respecto.

Tabla 11.

Ejemplo de característica codificada con One Hot Encoding

type_0	type_1	type_neighbor_0	type_neighbor_1	type_neighbor_2
1	0	0	0	1
1	0	1	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	0	0
1	0	1	0	0
0	1	0	0	1
1	0	1	0	0
1	0	1	0	0
1	0	0	0	1

Nota. La tabla muestra las características codificadas con One Hot Encoding. Elaboración propia, realizado con Python.

4.1.7.4. MinMax Encoding

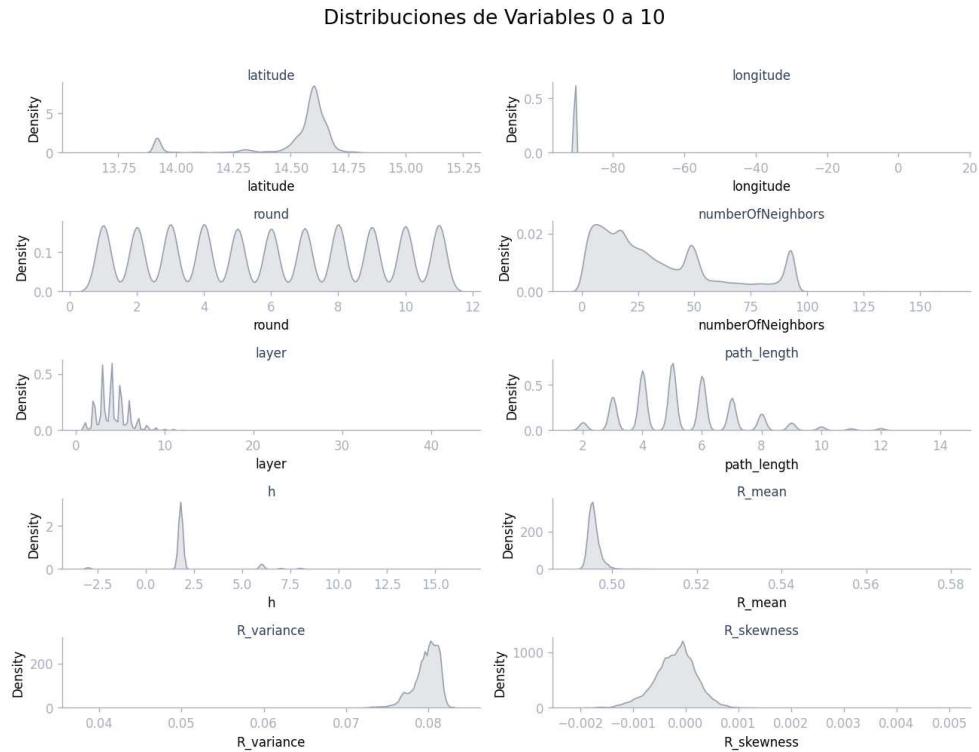
Las características numéricas, a diferencia de los anteriores contienen información codificada en números no sólo limitados a ceros y unos, sino que comprenden el valor de una medida en el sistema decimal. Como se mencionó antes, la escala de los valores puede significar un decremento en el desempeño de los algoritmos si no son tratadas previamente.

Para características como la altimetría, que va desde los aproximadamente 0 metros hasta un par de miles de metros y las obstrucciones de Fresnel que comprenden valores entre 0 y 1, sucede que existe la posibilidad

de que la información útil para aportar de la característica con una escala menor sea anulada por la variable con mayor escala, producto de la operatoria propia de cada algoritmo. Por tal motivo se realizó una normalización para disponer de las 86 variables numéricas en un rango comprendido entre 0 y 1. No sin antes hacer una inspección en las distribuciones de las variables, encontrando con esto la necesidad de actuar como se mencionará en seguida.

Figura 83.

Distribución de variables parte 1 de 9

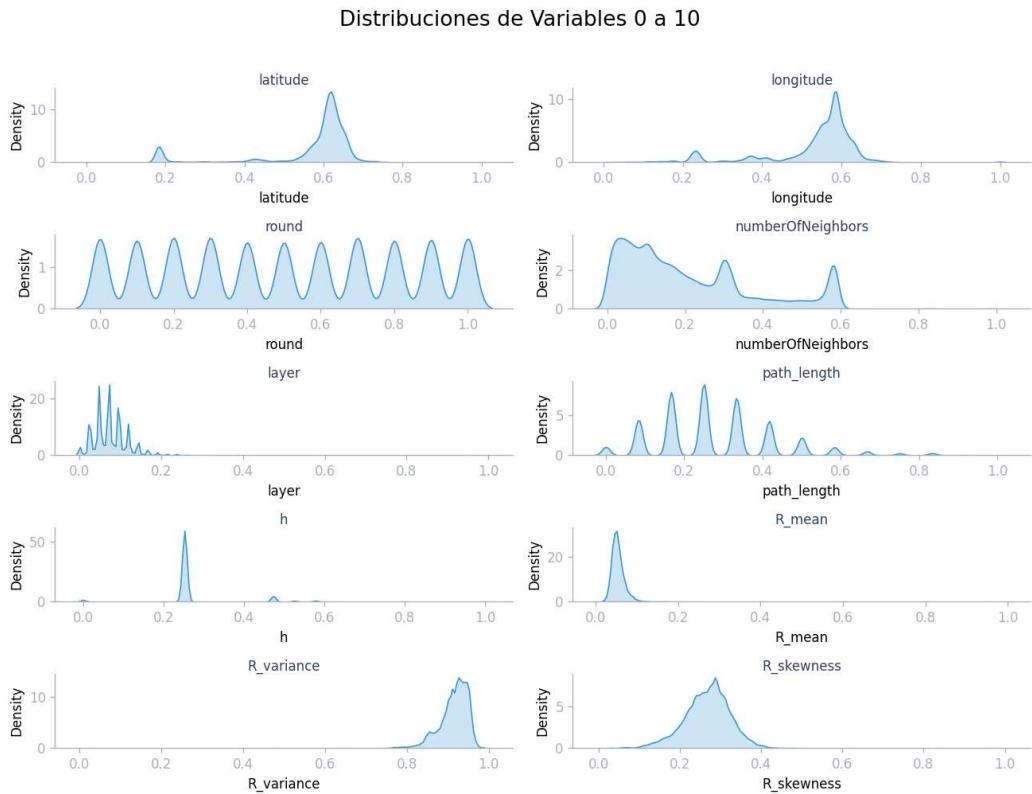


Nota. La tabla muestra la distribución de las variables numéricas en el conjunto de datos. La continuación de esta visualización se encuentra en los apéndices. Elaboración propia, realizado con Python.

Producto del análisis de la totalidad de los gráficos y del cálculo explícito de la variabilidad de las características numéricas, se decidió remover las características f1 y f20 del conjunto de datos, correspondientes a la obstrucción de la zona de Fresnel justo en el foco de la elipse. También fue necesario limpiar los 3 conjuntos de datos al remover no más de una decena de equipos de estos, dado que contenían valores de longitud evidentemente erróneos y que sesgaban considerablemente la distribución de la variable. Luego de esto se procedió a ajustar los escaladores MinMax, y se aplicaron a todos los conjuntos de datos.

Figura 84.

Distribución de variables parte 1 de 9



Nota. El gráfico muestra la distribución de las variables numéricas escaladas entre 0 y 1 en el conjunto de datos. La continuación de esta visualización se encuentra en los apéndices. Elaboración propia, realizado con Python.

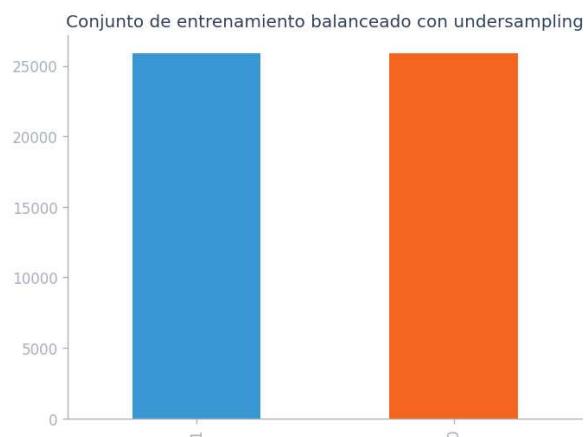
4.1.8. Balanceo de clases

Es un conocido problema en el ámbito de la ciencia de datos y el aprendizaje de máquina, el que el desbalance de clases en un conjunto de datos puede ser motivo para un desempeño inferior del esperado. Esto es todavía más significativo cuando la clase minoritaria, por su misma escasez tiene mayor importancia.

Existen diversas técnicas que propician el balanceo de datos. En este caso particular la etiqueta minoritaria es el valor 0 para la variable independiente (*Communicating*). Una de ellas evita aumentar la etiqueta minoritaria, y es conocida como submuestreo de la clase mayoritaria (*undersampling*). A continuación, se muestra el histograma del conjunto de datos resultante.

Figura 85.

Conjunto de entrenamiento Undersampled

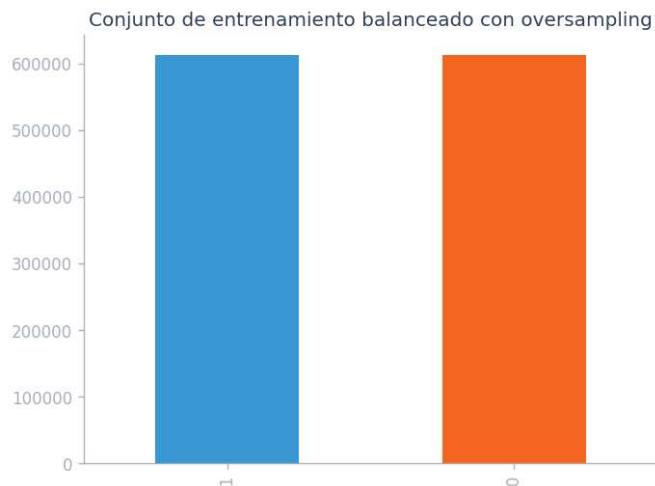


Nota. El gráfico muestra la cantidad de registros por clase al hacer balanceo por submuestreo.
Elaboración propia, realizado con Python.

Otra técnica para lidiar con el desbalance de clases es realizar un sobre muestreo de la clase minoritaria (*Oversampled*), esto por supuesto requiere la creación de variantes de los registros existentes con la etiqueta minoritaria, siempre y cuando se desee obtener una cantidad en el conjunto resultante, mayor que la cantidad de etiquetas 0 en el conjunto de datos original. Se utilizó la biblioteca Imblearn para generar estos datos sintéticos. A continuación, se muestra el histograma del conjunto de datos resultantes.

Figura 86.

Conjunto de entrenamiento Oversampled



Nota. El gráfico muestra la cantidad de registros por clase al hacer balanceo por sobre muestreo.
Elaboración propia, realizado con Python.

4.2. Heurística

Siempre se puede aventurar una solución a un determinado problema, haciendo uso del ingenio, o de que se conoce que alguien lo ha hecho antes. En el caso de este proyecto, cuando inició a buscarse una solución para el problema de la cobertura, evidentemente surgieron varias opciones de soluciones existentes, unas más complejas que otras, y más costosas también.

Sin embargo, dos proyectos de código abierto fueron los más razonables de investigar. Uno es el *software Radio Mobile* y el otro el proyecto *RFAanalysisJS*. No se utilizó ninguno de los dos en este proyecto, pero la heurística que ahora se presenta, se inspira en ambos.

La idea de que la geografía alrededor de un determinado equipo de RF influirá directamente en su cobertura es razonable. Las ondas electromagnéticas se propagan por el medio y las características particulares de la geografía influirán en atenuar la señal. Por supuesto, es imposible evaluar la totalidad de la geografía, pero seleccionando una serie de puntos alrededor del punto central, puede ser un buen inicio.

Figura 87.

Puntos de interés alrededor de equipo



Nota. La figura contiene el mapa que muestra los puntos radiales alrededor de un punto central para la aplicación de la heurística. Elaboración propia, realizado con Python.

Para estos puntos radiales, es posible calcular el nivel de obstrucción de la primera zona de Fresnel, más precisamente, se procedió a calcular la obstrucción de la zona de Fresnel comprendida entre cada una de las líneas radiales, para una altura de prueba elegida por el usuario. Para esto se utilizó la función previamente adaptada para la generación masiva de Fresnel.

Figura 88.

Obstrucción de la zona de Fresnel



Nota. La figura contiene el mapa con la heurística que consiste en la obstrucción de la zona de Fresnel alrededor de un punto central. Elaboración propia, realizado con Python.

La imagen anterior muestra el enfoque en cuestión, puede decirse hasta cierto punto que el mapa de calor arroja información relacionada con la cobertura de un equipo particular sobre su geografía cercana, en este caso específico, un radio de 5 kilómetros alrededor de un GW elegido al azar. Siendo que el mapa de calor muestra la obstrucción, los colores más oscuros corresponden a las áreas con mayor obstrucción en la primera zona de Fresnel. Se considera una mejor visualización, la del complemento de la obstrucción de la primera zona de Fresnel (1 menos la obstrucción de la primera zona de Fresnel).

Figura 89.

Línea vista



Nota. La figura contiene el mapa con la heurística que consiste la línea vista según la zona de Fresnel alrededor de un punto central. Elaboración propia, realizado con Python.

En la gráfica anterior, puede interpretarse las zonas más oscuras como aquellas donde la cobertura es mayor, y las zonas más claras o completamente sin color, como aquellas donde el nivel de cobertura es bajo. Esto, por supuesto, es una simplificación extrema del problema en cuestión, dado que no considera la arquitectura de la red, ni la información adicional disponible en el conjunto de datos.

Es posible agregar niveles de complejidad a esta heurística, tal como se hace en las soluciones referidas. A continuación, se detalla una serie de posibles enfoques para mejorar la precisión de la heurística:

- Realizar el cálculo de las obstrucciones de Fresnel desde el punto central hasta cada uno de los puntos radiales, de modo que la elipse de Fresnel corresponda al radio enlace con cada posición del equipo de prueba.
- Determinar por medio de los datos el radio adecuado para realizar los cálculos de las líneas radiales.
- Incluir la información geográfica adicional a las altimetrías obtenidas con la API de elevación, al hacer uso de las características extraídas de las imágenes satelitales.
- Determinar la cantidad mínima de la arquitectura de la red necesaria para mapear de manera adecuada la cobertura real, utilizando puntos fijos de validación según la etiqueta de Communicating en el conjunto de datos.

La indagación de estas ideas se aleja del objetivo del proyecto, pese a que pudieran arrojar resultados satisfactorios y a que existe, como se mencionó, evidencia del uso del análisis de Fresnel en conjunto con factores adicionales.

Cualquier planteamiento explícito que busque dar solución al problema basándose en estas u otras ideas, dejaría de lado el enfoque del proyecto basado en inteligencia artificial, o más concretamente, basado en datos. Por tal motivo, no se profundiza más en esta heurística, y se cede el lugar al aprendizaje de máquina.

4.3. Aprendizaje de máquina (*machine learning*)

Contrastando con las etapas anteriores, se espera que en el aprendizaje de máquina el humano intervenga lo menos posible, cosa que no se cumple a

cabalidad, evidentemente, puesto que el interés detrás de cada modelo implementado es humano. La postura en este apartado fue la de intervenir lo menos posible, configurando los distintos experimentos que realizarán búsqueda de parámetros aleatorios o con funciones creadas para tal propósito en las librerías de que se hará uso.

Cada uno de los experimentos consistió en el entrenamiento de un grupo de instancias de un modelo particular, del cual se obtuvo un modelo que fue el mejor según una métrica calculada únicamente con datos del conjunto de entrenamiento, también se almacenaron los parámetros configurados a cada experimento, así como el conjunto de datos con el que fue entrenado. Con el modelo obtenido se calcularon las métricas de interés en los dos conjuntos de datos de prueba, y se almacenaron para su posterior análisis. El seguimiento de los experimentos se realizó con la librería Mlflow.

Es importante hacer una distinción referente a los distintos parámetros que tienen los modelos de aprendizaje de máquina, diremos que los hiper parámetros son la configuración general de un algoritmo, cuyos valores no se ven afectados por el proceso de entrenamiento, mientras que los parámetros serán aquellos parámetros internos de los algoritmos que son optimizados en el proceso de entrenamiento.

Es posible decir que existe un solo espacio de parámetros, y que es deseable optimizar también los hiper parámetros. Los alcances de este proyecto no llegan a pretender una optimización de estos parámetros, simplemente se seleccionará el conjunto de hiper parámetros con mejor resultado en la métrica establecida para la selección.

Según lo anterior, lo presentado en este apartado serán los modelos seleccionados y las distintas formas de pretender encontrar el conjunto de hiper parámetros sobresaliente, tomando en cuenta los distintos conjuntos de datos y los límites inherentes al poder computacional disponible. El análisis de los resultados se detalla en el capítulo siguiente.

4.3.1. Validación cruzada

Que tan bien puede generalizar un modelo es un tema de interés en el aprendizaje de máquina. Existe la posibilidad de que un modelo simplemente memorice sus datos de entrenamiento, y con eso consiga tener una métrica muy buena al momento del entrenamiento, sin embargo, al ser expuesto a datos nuevos, falla debido a diferencias entre el conjunto de datos de prueba y de entrenamiento. En tal caso, se dirá que el modelo está sobre ajustado, y su generalización será baja.

Una forma de seleccionar un modelo con una mejor generalización es mediante la validación cruzada. Este enfoque implica hacer divisiones del conjunto de datos de entrenamiento, digamos k de ellas, luego, se entrena el modelo con $k-1$ de las divisiones y se evalúa en la división restante, esto se itera de modo que todas las divisiones se utilicen para evaluar el modelo. Luego, se asigna la métrica del modelo según el desempeño en las k divisiones, representadas con un estadístico, usualmente la media.

4.3.2. Búsqueda aleatoria con validación cruzada

Haciendo uso de la función `RandomizedSearchCV` de la biblioteca Scikit-Learn, se implementó una búsqueda aleatoria dentro de una distribución dada de parámetros, la cual consiste en un diccionario indicando el parámetro del modelo

en cuestión asociado al grupo de posibles valores. La elección de este método de búsqueda fue principalmente debido a la capacidad computacional disponible.

La búsqueda consiste en elegir un número determinado de combinaciones aleatorias de todos los valores posibles para todos los parámetros, luego realizar el entrenamiento aplicando la validación cruzada, eligiendo al final de este proceso el que tenga mejor métrica general dentro de las K particiones de los datos, según se refirió antes.

En cuanto a la validación cruzada, se utilizaron dos variantes, la primera consistió en indicar simplemente el número de particiones, la segunda fue con división de las particiones por agrupación de algún valor, específicamente fue agrupación basada en la característica fuera del conjunto de datos de entrenamiento, serialNumber.

4.3.3. Modelos

Todos los modelos utilizados se enmarcan en el aprendizaje supervisado. Los modelos consisten en algoritmos que cuentan con una función de pérdida, parámetros internos e hiper parámetros. La mayoría de ellos ajustan sus parámetros internos para minimizar la pérdida, dado el conjunto de datos de entrenamiento en la iteración particular de entrenamiento.

Las fronteras en cuanto a la definición de *machine learning*, no están inequívocamente establecidas. Se consideran acá ciertos modelos comunes en el aprendizaje de la Ciencia de Datos. Se intenta con estos dar solución a dos problemas particulares, el conjunto de datos de prueba 1 y el conjunto de datos de prueba 2.

4.3.3.1. K-Nearest Neighbors (KNN)

Este es el único algoritmo de los que se utilizaron, que no tiene un proceso de entrenamiento como tal. Como su nombre lo indica, se basa en los k registros más cercanos para asignar el valor correspondiente a la etiqueta del registro que se quiere predecir. Existe la posibilidad de utilizar KNN para una regresión o para clasificación, debido al tipo de etiqueta de que se dispone, se utilizó el segundo.

Los hiper parámetros configurados para la distribución de parámetros de la búsqueda aleatoria para el algoritmo KNN, fueron los siguientes:

- K: número de vecinos que se considerarán para la asignación de la etiqueta. Los valores deben ser números enteros en el rango $[1, \infty)$.
- Weights: la manera en que se asignarán los pesos a los distintos vecinos para su contribución en la etiqueta predicha. Los valores pueden ser *uniform* o *distance*.

4.3.3.2. Random Forest (RF)

Este algoritmo consiste en entrenar varias instancias de árboles de decisión. Los árboles de decisión a su vez son algoritmos supervisados no paramétricos que pueden ser utilizados para regresión o clasificación, según una serie de umbrales de decisión, de nuevo, se utilizó el algoritmo en su variante para la clasificación binaria de los datos. RF ajusta múltiples clasificadores, lo que constituye un bosque de árboles de decisión, luego realiza la asignación de la predicción según la predicción más común en el bosque.

Los hiper parámetros configurados para la distribución de parámetros de la búsqueda aleatoria para el algoritmo RF, fueron los siguientes:

- N_estimators: número de árboles en el bosque. Los valores deben ser números enteros en el rango $[1, \infty)$.
- Max_features: número de características a considerar para dividir un nodo en los árboles de decisión. Los valores posibles son sqrt, log2, *None*, también pueden ser números enteros en caso se indique un valor específico de características en el rango $[1, n]$, siendo n el número máximo de características del conjunto de datos, o con punto flotante, que indicarán la proporción de características a utilizar (0.0,1.0].
- Max_depth: la profundidad máxima de los árboles en el bosque. Los valores pueden ser enteros, o *None*, en caso de utilizar este último, la profundidad de los árboles dependerá de tener hojas puras (sin diversidad), o el valor indicado en el parámetro ‘min_samples_leaf’.
- Min_samples_split: número mínimo de muestras requeridas para dividir un nodo. Si el valor es entero, debe ser en el rango $(2, \infty]$, y ese será el número de muestras requeridas. Si es un número con punto flotante será la proporción que corresponda dentro del rango (0.0,1.0].
- Min_samples_leaf: mínimo número de muestras en un nodo hoja. Si el valor es entero, debe ser en el rango $[1, \infty)$, ese será el número mínimo de muestras en una hoja. Si es un número con punto flotante será la proporción que corresponda dentro del rango (0.0,1.0].

- Bootstrap: Indica si se utiliza muestreo con reemplazo, o de lo contrario no hay repetición de los registros en los distintos árboles. El tipo de valor aceptado es booleano.
- Criterion: función para medir la calidad de los umbrales de decisión, al haber realizado una división. Los posibles valores son gini, entropy y log_loss.

4.3.3.3. Gradient Boosting Classifier (GBC)

Este algoritmo construye árboles decisión de manera secuencial, de modo que cada uno de los árboles corrija los errores de los anteriores mediante una optimización explícita de una función de pérdida. Esta optimización iterativa lo diferencia de los árboles de decisión y de random forest, a la vez que cumple la función de optimizar la predicción general.

Los hiper parámetros configurados para la distribución de parámetros de la búsqueda aleatoria para el algoritmo GBC, fueron los siguientes:

- N_estimators: número de árboles que construirá el algoritmo. Los valores deben ser enteros en el rango $[1, \infty)$.
- Max_depth: profundidad máxima de los clasificadores individuales. Este parámetro limita el número de nodos en los árboles. Los valores pueden ser enteros en el rango $[1, \infty)$, o *None*, en el caso de este último, los nodos se expanden hasta que las hojas sean puras o hasta que todas las hojas contengan menos registros que lo indicado en el parámetro min_samples_split.

- Learning_rate: controla la contribución de cada árbol al modelo final, este es el valor que cada paso agrega en la dirección contraria al gradiente de la función de pérdida, con el fin de minimizar su valor. Los valores aceptados son datos numéricos con punto flotante en el rango $(0.0, \infty)$.
- Subsample: representa la fracción de las muestras que se utilizará en el ajuste de los modelos base. Los valores aceptados son números con punto flotante en el rango $(0,1.0]$.
- Min_samples_split: el número mínimo de muestras requerido para dividir un nodo interno. Los valores pueden ser números enteros en el rango $[2, \infty)$, o números con punto flotante en el rango $(0,1.0]$.
- Min_samples_leaf: número mínimo de muestras requerido en un nodo hoja. Los valores pueden ser números enteros en el rango $[2, \infty)$, o números de punto flotante en el rango $(0.0,1.0)$.

4.3.3.4. ***Support Vector Machine (SVM)***

Este algoritmo de aprendizaje supervisado puede ser utilizado tanto para la regresión como para la clasificación, en este caso se utilizó su variante SVC (*support vector classification*). Este tipo de máquinas tienen la tarea de encontrar un hiperplano que separa el espacio de características, que es de alta dimensionalidad. La separación de este espacio de características tendrá en cada lado del hiperplano una etiqueta asignada. Los vectores de soporte son los puntos más cercanos al hiperplano y el algoritmo maximiza la distancia entre el hiperplano y estos puntos más cercanos, optimizando así la clasificación y mejorando la generalización.

Los hiper parámetros configurados para la distribución de parámetros de la búsqueda aleatoria para el algoritmo SVM, fueron los siguientes:

- C: es el parámetro de regularización, que cumple la función de evitar el sobreajuste al momento del entrenamiento. El peso de la regularización es inversamente proporcional a C. Los valores que puede tomar este parámetro son números de punto flotante en el rango $(0.0, \infty)$.
- Gamma: este parámetro está relacionado con los kernels aplicados en SVC. Puede decirse que este parámetro controla la influencia de cada muestra particular en la decisión del modelo. Los valores que puede tener este parámetro son números con punto flotante, scale y auto.
- Kernel: indica el tipo de transformación a utilizar para convertir el espacio de características, lo que pretende conseguir que los datos sean linealmente separables en el nuevo espacio. Los kernels posibles son linear, poly, rbf, sigmoid, precomputed.
- Class_weight: asigna un peso a cada una de las clases en la etiqueta del conjunto de datos, permitiendo con eso controlar la importancia que se desea asignar a cada una de ellas. Los posibles valores son un diccionario indicando explícitamente el peso de cada clase, balanced o None.

4.3.4. Experiments de *machine learning*

Grosso modo, los experimentos fueron realizados siguiendo una idea general básica, ya sea la métrica de selección, la estrategia para la validación cruzada, entre otras variaciones. Dentro de cada experimento existen distintas corridas de la búsqueda aleatoria, para distintos modelos y con distintos

conjuntos de datos. De nuevo, la capacidad computacional disponible fue un condicionante, por tal motivo, existen ciertas combinaciones de modelo y datos que no se probaron, debido a una simple inspección, que indicó que tal o cual modelo, o tal o cual conjunto de datos consume considerables recursos computacionales sin arrojar resultados satisfactorios, y en general, son menos eficientes que otras combinaciones.

Tabla 12.

Experimentos machine learning

Experimento	Métrica	Validación Cruzada	Particiones
ML-1	F1-Score	K-Fold Cross-Validation	3
	Avg. Macro		
ML-2	F1-Score Avg. Macro	Group-K-Fold Cross-Validation	3
ML-3	F1-Score Avg. Weighted	Group-K-Fold Cross-Validation	3

Nota. La tabla muestra los criterios generales para los experimentos de *machine learning*. Elaboración propia, realizado con Word.

4.4. Aprendizaje profundo (*deep learning*)

Como se mencionó en la revisión de la literatura, la inclusión de las redes neuronales en el aprendizaje de máquina fue tal que requirió crear una nueva división en esta rama de la inteligencia artificial, el aprendizaje profundo. Su nombre se debe a la existencia de distintas capas profundas con neuronas que tienen la función de transformar el espacio de características con una función no lineal aplicada a una combinación lineal de las capas anteriores.

Se dice que las redes neuronales artificiales están inspiradas en los cerebros biológicos y sus intrincadas conexiones entre neuronas. De alguna

forma, tal interacción de elementos simples consigue la escala de capacidades cognitivas existentes en la naturaleza. Es la apreciable mejora en los resultados que proveen las neuronas artificiales y las redes de estas, lo que las diferencia del resto de modelos de aprendizaje de máquina.

Aunque las redes neuronales no son un modelo precisamente nuevo, la introducción de la retro propagación, el aumento exponencial de la capacidad computacional y la mejora significativa de resultados en problemas de distinta índole, les permitieron ganar su reputación. Hoy día existen distintas arquitecturas y distintos paradigmas sobre cómo utilizar las redes neuronales. En los alcances de este trabajo están únicamente las redes neuronales simples, compuestas de una capa de entrada, cierta cantidad de capas profundas, ciertas capas de dropout (que apagan de manera aleatoria ciertas neuronas para evitar el sobreajuste) y una capa de salida.

Es posible entender una red neuronal como un sistema automático de representación de características en un espacio distinto del espacio original de características, que concluye con un clasificador lineal. Lo resaltable en cuanto a las redes neuronales es que la representación de tales características es ajustada una y otra vez a lo largo del proceso de entrenamiento al variar los pesos en las entradas de las neuronas y el hiperplano de decisión final. Por tal motivo no es de extrañarse que en algunos textos incluyan a las redes neuronales como una técnica de Ingeniería de Características.

4.4.1. Keras Hyperband

Se utilizó la biblioteca Tensorflow, de Google, para entrenar las redes neuronales. Esta biblioteca proporciona toda la operatoria de fondo que permite el proceso de *Forward Propagation* y *Back Propagation*, la predicción de la red

neuronal y el ajuste de sus parámetros. Keras por su parte es una biblioteca de alto nivel que permite construir y entrenar modelos de *Deep Learning*.

Keras cuenta con una función de optimización de hiper parámetros, esta es diferente a la búsqueda utilizada en *machine learning* en que realizará la búsqueda en todo el espacio de parámetros configurado y en que no completará todos los entrenamientos para calcular la métrica y seleccionar el mejor, por el contrario, esta biblioteca optimiza los recursos asignando mayores recursos a aquellas combinaciones de parámetros más prometedoras. Las peores combinaciones de parámetros serán rápidamente descartadas.

4.4.2. Arquitectura de ANN

Con Keras es realmente sencillo configurar una red neuronal. El Hyperband Tuner requiere una clase heredada de Hyperband de la biblioteca Kerastuner. De manera secuencial se van agregando las capas al modelo genérico, indicando los rangos de los parámetros a ser evaluados. A continuación, se indican los tipos de capas agregadas a los modelos. Los rangos y variantes serán indicadas en los experimentos particulares.

- Input: capa de entrada a la red neuronal. Se configura su parámetro `shape`, que debe coincidir con la dimensionalidad del conjunto de datos de entrenamiento.
- Dense: esta capa corresponde tal cual a una capa de neuronas dentro de la red. Estas capas usualmente son llamadas capas profundas. También es utilizada para la capa de salida, con una diferencia en la función de activación. Los parámetros configurados son `units`, `activation` y `kernel_regularizer`.

- Dropout: este tipo de capa elimina ciertas neuronas de la capa anterior. El parámetro que se configura es *rate*, que indica la taza de —apagado— de neuronas.

4.4.3. Experimentos de *deep learning*

Al igual que con ML, acá los experimentos son variaciones generales de la configuración, previo a la aplicación de la función de ajuste de hiperparámetros. En este caso no existe una validación cruzada como tal, en contraposición se utiliza un conjunto de validación general para toda la búsqueda de los parámetros. Por tal motivo, los experimentos acá serán variaciones de este conjunto de validación, de la métrica de selección, entre otras ideas que pretenderán solucionar alguno de los dos problemas que se señalan en breve, en el siguiente capítulo.

Tabla 13.

Experimentos deep learning

Experimento	Métrica de selección	Conjunto de validación	Regularización
DL	Accuracy	División aleatoria por serialNumber	Sí
DL-2	Accuracy	División aleatoria simple	Sí
DL-3	Accuracy	División aleatoria por serialNumber	No
DL-4	Accuracy	División aleatoria simple	No

Nota. La tabla muestra los criterios generales para los experimentos de *Deep Learning*. Elaboración propia, realizado con Word.

5. ANÁLISIS DE RESULTADOS

Es casi una máxima al hablar de datos el que la correlación no indica causalidad, esto es eminentemente cierto, las intrincadas causas que comprenden los fenómenos en lo que llamamos realidad son incomprensibles en última instancia. La recolección y análisis de datos permiten tomar cierta información representativa del mundo real, con una cierta resolución y bajo cierto alfabeto de símbolos. Las interpretaciones aquí dadas sobre lo que puede o no suceder con los algoritmos, y de lo que puede o no significar uno u otro resultado es meramente un ejercicio de buscar cierta verosimilitud en la historia que se cuenta de los datos y los resultados obtenidos.

A los humanos nos gustan las historias, como ya se ha mencionado. La conclusión de este trabajo no podría ser otra cosa que una historia. El marco sobre el que estará contada es el de un conjunto de datos obtenido y tratado según las circunstancias discutidas antes. No se utilizará la formalidad matemática que implica un análisis de causalidad o cualquier otra exquisites estadística, conseguir algo así ahora mismo se ve remoto. Se trata de interpretar lo que aparentemente indican las visualizaciones a las que se recurre, lo que tal métrica podría indicar, cómo un conjunto de datos puede ser visto como un problema. Es decir, qué más se podría esperar de alguien cuya capacidad biológica sólo le permite vislumbrar escenarios en tres dimensiones, frente a las más de cien que los algoritmos digieren para arrojar sus resultados.

5.1. Los problemas atendidos

El conjunto de datos 1, corresponde a la última etapa de las rondas de datos tomadas. Debido a que, de la totalidad de los equipos, la mayoría permanecen a lo largo de las 12 rondas, es posible ver este conjunto de datos como el problema de predecir si un equipo particular va a comunicar en el futuro, tomando en cuenta la información de las once semanas anteriores. Desde el punto de vista del negocio, esto puede ser de interés dado que predecir el nivel de comunicación de la red permite accionar con anticipación ante la caída del nivel general de comunicación. La mayor variabilidad de este conjunto de datos respecto del de entrenamiento está en la característica temporal, esto dado que no existe la ronda en el conjunto de datos.

El conjunto de datos 2, corresponde a ciertos equipos retirados por completo del conjunto de datos de entrenamiento, la intención detrás de esta decisión es evaluar los modelos con equipos nunca vistos. Desde el punto de vista del negocio, esto puede ser de interés dado que brinda una herramienta que permite evaluar el estado de comunicación para equipos nuevos, de modo que se pueda decidir si se brinda el servicio de la red o no. La menor variabilidad de este conjunto de datos respecto del de entrenamiento está en la característica temporal, es decir, los equipos seleccionados se encuentran en promedio en la mayoría de las rondas, al igual que el conjunto de entrenamiento. Sin embargo, estos equipos, con sus respectivos vecinos y toda la serie de características generadas a partir de esta relación, son inexistentes para los algoritmos durante su entrenamiento.

En la práctica fue significativamente más fácil encontrar una solución al primer problema, disponer del pasado de la comunicación de un equipo arroja bastante información sobre si se comunicará en el futuro. Desde el inicio se intuyó

una mayor dificultad en el segundo problema, saber si un equipo nuevo comunicará no es tan implícito como saber si uno ya existente lo hará. Por esto último, las métricas seleccionadas fueron desde el inicio pensadas para resolver el segundo problema, logrando buenos resultados para el primer problema y no precisamente lo esperado para el segundo. En cierto punto se decidió presentar una clara distinción entre ambos problemas en este análisis de resultados.

5.2. Métricas

Una métrica es un valor numérico que se utiliza para tener una noción de cómo se desempeña un algoritmo. Esta es externa al algoritmo, y se calcula sobre los conjuntos de prueba, luego de obtener una predicción del algoritmo que se desea evaluar, y en comparación con la correspondiente etiqueta verdadera del conjunto de datos de prueba. El algoritmo, por sí mismo tiene una función de pérdida, que es calculada únicamente con los datos del conjunto de entrenamiento, y que es la función objetivo optimizada por el algoritmo (no todos los algoritmos utilizados optimizan una función de pérdida).

En ambos problemas la etiqueta a resolver tiene dos valores posibles 0 y 1, es como se conoce, un problema de clasificación binaria. A diferencia de la regresión que asigna el valor de salida en un rango continuo, la clasificación está limitada cierta cantidad específica de clases, en este caso 2. La métrica que principalmente se utilizó fue F1 Score, esta es comúnmente utilizada en la evaluación de algoritmos de clasificación, cuenta también con variantes que se desempeñan bien en conjuntos de datos desbalanceados, como es el caso del conjunto básico de entrenamiento.

F1 Score recurre a otras dos métricas para el cálculo de su valor, *precision* (precisión) y *recall* (sensibilidad). Para plantear sus relaciones matemáticas se requieren las siguientes definiciones.

- Verdaderos positivos (TP): etiqueta predicha como positiva, que efectivamente es positiva en los conjuntos de prueba.
- Falsos positivos (FP): etiqueta predicha como positiva, que en realidad es negativa en los conjuntos de datos de prueba.
- Verdaderos negativos (TN): etiqueta predicha como negativa, que efectivamente es negativa en los conjuntos de prueba.
- Falsos negativos (FN): etiqueta predicha como negativa, que en realidad es positiva en el conjunto de datos de prueba.

Precision (precisión) mide la exactitud de las predicciones positivas. Es la proporción de verdaderos positivos entre todos los resultados positivos predichos por el modelo en cuestión. Matemáticamente su relación es como sigue:

$$precision = \frac{TP}{TP+FP} \quad (\text{Ec. 1})$$

Recall (sensibilidad) mide la capacidad del modelo de encontrar todas las instancias positivas. De todas las etiquetas reales positivas, cuántas se predijeron correctamente como positivas. Matemáticamente su relación es como sigue:

$$recall = \frac{TP}{TP+FN} \quad (\text{Ec. 2})$$

F1 Score toma en cuenta ambas métricas para su cálculo. La media armónica de *precision* y *recall* corresponde al valor de F1 Score. Al utilizar la media armónica, se tiene una forma más efectiva de representar ambas razones. Por lo tanto, la métrica F1 Score está diseñada para representar con un solo número qué tan asertivas son las predicciones positivas del modelo y qué tan sensible es este para identificar todas las etiquetas que en realidad son positivas.

$$F1 = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \quad (\text{Ec. 3})$$

Otra métrica de referencia fue *Accuracy* (exactitud), que es una métrica que indica la asertividad global del algoritmo respecto de la etiqueta verdadera. Esta métrica no diferencia entre clases, de modo que solo importa la cantidad de predicciones correctas respecto de la cantidad total de predicciones realizadas por el algoritmo.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (\text{Ec. 4})$$

Como una forma de buscar el balance en el nivel de acierto de ambas categorías pese al desbalance de clases se utilizó una métrica adicional. La Exactitud Media por Clase (MPCA según sus siglas en inglés). Esta se define como sigue.

$$MPCA = \frac{1}{2} \left(\frac{TN}{TN+FP} + \frac{TP}{TP+FN} \right) \quad (\text{Ec. 5})$$

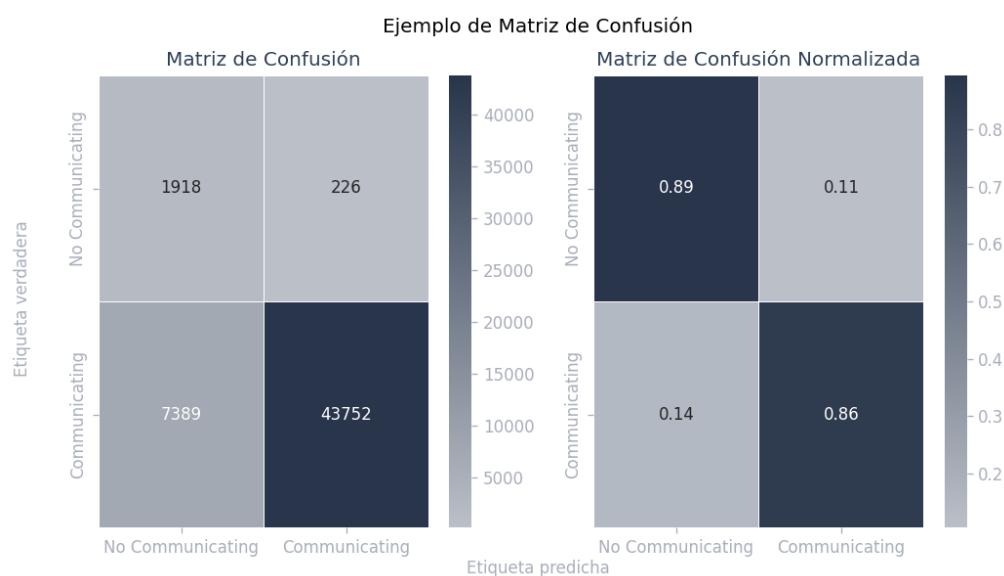
La selección entre los distintos algoritmos y experimentos, así como su comparación será basada en las métricas acá mencionadas, y considerando el particular contexto, según los problemas referidos antes.

5.2.1. Matriz de confusión

La visualización utilizada para asimilar de mejor manera las métricas fue la Matriz de Confusión, en el caso de la clasificación binaria esta matriz tiene dos filas y dos columnas. En el eje vertical se indican las etiquetas verdaderas para el conjunto de datos en cuestión, y en el eje horizontal se indica la clase predicha por el algoritmo que se está evaluando. A continuación, un ejemplo de esta visualización.

Figura 90.

Ejemplo de matriz de confusión



Nota. La figura muestra un ejemplo de la visualización de las matrices de confusión utilizadas para presentar los resultados de los algoritmos. Elaboración propia, realizado con Python.

Se aprecia en la imagen anterior la visualización utilizada para presentar los resultados de la aplicación de un modelo sobre determinado conjunto de datos. En la primera matriz se evidencia que el conjunto de datos de prueba está

desbalanceado, por lo que la escala de colores muestra la proporcionalidad general y la asimilación numérica no es tan directa. En busca de una mejor representación de la información contenida en la gráfica, se normalizó la matriz de confusión (segunda matriz en la imagen anterior), lo que por convención para esta visualización implica normalizar a nivel de filas, de modo que se puede apreciar de manera más directa cómo se comportó la predicción sobre cada etiqueta predicha por el modelo, independientemente del desbalance de clases. Este problema del desbalance también impacta en el cálculo de las métricas, esta es la razón detrás de utilizar varias versiones de la métrica F1-Score para evaluar y comparar los modelos. Las visualizaciones correspondientes a la totalidad de modelos implementados se encuentran en los apéndices.

5.3. Tablas de resultados

Durante la experimentación con los distintos algoritmos se utilizó la librería Mlflow para realizar el seguimiento individual de cada experimento. En la sección de análisis de resultados se agregaron todos los resultados en un solo experimento (por paradigma de aprendizaje de máquina), con la finalidad de comparar de manera más directa el desempeño de los algoritmos. A continuación, las tablas reducidas con los resultados de los respectivos experimentos tanto de *machine learning* como de *deep learning*, para cada uno de los problemas atendidos. Las tablas completas se presentan en la sección de apéndices.

Tabla 14.*Tabulación de resultados ML problema 1*

Algoritmo	DatosEntrenamiento	Accuracy	F1 Score Macro	MPCA	Experiment
GBC	Training_OS	0.9937881204841887	0.9617389312230621	0.9853688137084857	ML
GBC	Training_US	0.9910481373744956	0.9469679113648417	0.9886334609144691	ML
KNN	Training_US	0.857089237121141	0.627467482511822	0.8750533260108795	ML
KNN	Training	0.9810077883081543	0.8742939052926972	0.8661009905085355	ML
RF	Training_OS	0.9974664539739139	0.9836021191319406	0.9837101713889052	ML
RF	Training_US	0.9870507647555598	0.9258237856733549	0.9816354776536744	ML
RF	Training	0.9956272872290514	0.9710511601555225	0.9606322617130807	ML
SVM	Training_US	0.9022238904006756	0.6883754934247245	0.9095149208130171	ML
GBC	Training_OS	0.8757436426761753	0.640237414926814	0.8530442029309078	ML2
GBC	Training_US	0.9512996152763442	0.7919310356318704	0.948710847563088	ML2
KNN	Oversampled	0.9470582715586	0.767465822316771	0.899580536704639	ML2
KNN	Undersampled	0.7653373369616214	0.5505203678938632	0.8393194357011797	ML2
RF	Training_OS	0.8840949610584592	0.651837324454422	0.8614166967269594	ML2
RF	Training_US	0.852115980106972	0.6122676581957869	0.834478998945555	ML2
GBC	Training_US	0.9444871915173125	0.772652093163337	0.9391291702819275	ML3
GBC	Training_OS	0.9599136717650371	0.8173224396755909	0.9549858971990519	ML3
KNN	Oversampled	0.9199962465984799	0.7134187248238428	0.9017928000564432	ML3
KNN	Undersampled	0.7703481279909918	0.5487135642929931	0.815341450086635	ML3
SVM	Training_US	0.8953551656188421	0.6777435694476881	0.9045959861994072	ML3

Nota. La figura muestra la tabulación de los resultados de aprendizaje de máquina para el problema 1 con mapa de color. Elaboración propia, realizado con Python.

Tabla 15.*Tabulación de resultados ML problema 2*

Algoritmo	DatosEntrenamiento	Accuracy	F1 Score Macro	MPCA	Experiment
GBC	Training_OS	0.947563491861324	0.5991315235203688	0.57011890241781	ML
GBC	Training_US	0.9468629798507272	0.5754801496781826	0.5524612248792984	ML
KNN	Training_US	0.823407290419543	0.5413461617601918	0.6160223033361067	ML
KNN	Training	0.9372978067605777	0.588888807995449	0.5714604872996134	ML
RF	Training_OS	0.9502891204116464	0.5409551240325003	0.5284516144744547	ML
RF	Training_US	0.9487352574063225	0.66008536139977463	0.6272203186304477	ML
RF	Training	0.95076037394604780	0.53231015561191230	0.5234896018163148	ML
SVM	Training_US	0.8673994446850243	0.574162833861682	0.6316193303531933	ML
GBC	Training_OS	0.8617189291081845	0.6034448023421664	0.7082018711859823	ML2
GBC	Training_US	0.9147795297653921	0.60978714777164840	0.6246076016433514	ML2
KNN	Oversampled	0.8997376264105764	0.5808792635831252	0.6018814246222389	ML2
KNN	Undersampled	0.7467967496242708	0.5084900473532873	0.6281209213185379	ML2
RF	Training_OS	0.8625085971928573	0.5838714552672037	0.6599491745966424	ML2
RF	Training_US	0.8392133886950098	0.5907170473439616	0.7210944807767052	ML2
GBC	Training_US	0.9168301194691393	0.6346161768861508	0.6595546638992518	ML3
GBC	Training_OS	0.9194029090353313	0.6030260647599679	0.608333631546144	ML3
KNN	Oversampled	0.8751178133836004	0.5656415905762048	0.6052527204463649	ML3
KNN	Undersampled	0.750299309677255	0.5137979992776873	0.6409787394146877	ML3
SVM	Training_US	0.8670173472246988	0.5665698436190046	0.6168529502690376	ML3

Nota. La figura muestra la tabulación de los resultados de aprendizaje de máquina para el problema 2 con mapa de color. Elaboración propia, realizado con Python.

Tabla 16.

Tabulación de resultados DL problema 1

DatosEntrenamiento	Accuracy	F1 Score Macro	MPCA	DeepLayers	Threshold	Experiment	Extra
Oversampled	0.8064933846298208	0.5658744404477993	0.7892618979660272	2	acc	DL	-
Oversampled	0.9576803978605611	0.6780130669038205	0.651965728820131	2	acc	DL	Training
Oversampled	0.7727690717914742	0.5442595858048869	0.7875556070644945	2	acc	DL	Undersampled
Oversampled	0.740502017453317	0.5293276586752351	0.7927780538776756	3	acc	DL	-
Oversampled	0.9545463075912546	0.6775330186148534	0.6621748873541602	3	acc	DL	Training
Oversampled	0.7543774045228488	0.534646521196317	0.7931685275957865	3	acc	DL	Undersampled
Undersampled	0.04081284153138782	0.039307558233116754	0.5003030836315285	2	acc	DL	-
Oversampled	0.8064933846298208	0.5658744404477993	0.7892618979660272	2	f1	DL	-
Oversampled	0.9426104907572488	0.6850102659893833	0.7189646264775145	2	f1	DL	Training
Oversampled	0.7780989021300554	0.5473906860464763	0.7872050890105697	2	f1	DL	Undersampled
Oversampled	0.7922867598761377	0.5583410305341157	0.7960607307438281	3	f1	DL	-
Oversampled	0.943177442056864	0.6815189742073017	0.7081464278084558	3	f1	DL	Training
Oversampled	0.75453774045228488	0.534646521196317	0.7931685275957865	3	f1	DL	Undersampled
Undersampled	0.4885239748520983	0.366018436089824	0.54585766976696627	2	f1	DL	-
Oversampled	0.8064933846298208	0.5658744404477993	0.7892618979660272	2	mpca	DL	-
Oversampled	0.8064933846298208	0.5658744404477993	0.7892618979660272	2	mpca	DL	Training
Oversampled	0.7780989021300554	0.5473906860464763	0.7872050890105697	2	mpca	DL	Undersampled
Oversampled	0.823383691470395	0.5803599440043122	0.8002954208105364	3	mpca	DL	-
Oversampled	0.7922867598761377	0.5583410305341157	0.7960607307438281	3	mpca	DL	Training
Oversampled	0.770742235150652	0.5448887964565914	0.7961081752468373	3	mpca	DL	Undersampled
Undersampled	0.9425354227268462	0.5223562236601433	0.5185069576079828	2	mpca	DL	-
Oversampled	0.9372618935910669	0.7532811527858563	0.9270980807524529	2	acc	DL2	-
Oversampled	0.9589377873698038	0.8102926195629101	0.9386138223136093	3	acc	DL2	-
Undersampled	0.8538237777868301	0.6156440806705343	0.8414013617823362	2	acc	DL2	-
Oversampled	0.9372618935910669	0.7532811527858563	0.9270980807524529	2	f1	DL2	-
Oversampled	0.9589377873698038	0.8102926195629101	0.9386138223136093	3	f1	DL2	-
Oversampled	0.8538237777868301	0.6156440806705343	0.8414013617823362	2	f1	DL2	-
Oversampled	0.9372618935910669	0.7532811527858563	0.9270980807524529	2	mpca	DL2	-
Oversampled	0.9589377873698038	0.8102926195629101	0.9386138223136093	3	mpca	DL2	-
Oversampled	0.8538237777868301	0.6156440806705343	0.8414013617823362	2	mpca	DL2	-
Oversampled	0.9372618935910669	0.7532811527858563	0.9270980807524529	2	mpca	DL2	-
Oversampled	0.9589377873698038	0.8102926195629101	0.9386138223136093	3	mpca	DL2	-
Oversampled	0.8538237777868301	0.6156440806705343	0.8414013617823362	2	mpca	DL2	-
Oversampled	0.9372618935910669	0.7532811527858563	0.9270980807524529	2	mpca	DL2	-
Oversampled	0.9589377873698038	0.8102926195629101	0.9386138223136093	3	mpca	DL2	-
Oversampled	0.8293703668949985	0.5952707550344373	0.8418454641359694	2	mpca	DL2	-
Oversampled	0.76323543211035	0.5397420678473672	0.7928677149026382	2	acc	DL3	-
Oversampled	0.7193769353476586	0.5133166174024189	0.783648475738863	3	acc	DL3	-
Oversampled	0.27518063244815616	0.24355631730203248	0.5643045660701983	2	acc	DL3	-
Oversampled	0.76323543211035	0.5397420678473672	0.7928677149026382	2	f1	DL3	-
Oversampled	0.7193769353476586	0.5133166174024189	0.783648475738863	3	f1	DL3	-
Undersampled	0.7212911701229239	0.478287983487978	0.60679739758036416	2	f1	DL3	-
Oversampled	0.76323543211035	0.5397420678473672	0.7928677149026382	2	mpca	DL3	-
Oversampled	0.7193769353476586	0.5133166174024189	0.783648475738863	3	mpca	DL3	-
Oversampled	0.7212911701229239	0.478287983487978	0.60679739758036416	2	mpca	DL3	-
Oversampled	0.96878878589659378	0.8227749058174627	0.9406669694949317	2	acc	DL4	-
Oversampled	0.9676644459041006	0.8397085647503535	0.945841261531482	3	acc	DL4	-
Undersampled	0.87856122736229708	0.6451812799427283	0.871291793930389	2	acc	DL4	-
Oversampled	0.9676644459041006	0.8397085647503535	0.945841261531482	2	f1	DL4	-
Oversampled	0.82278788539659378	0.8227749058174627	0.9406669694949317	2	mpca	DL4	-
Oversampled	0.9676644459041006	0.8397085647503535	0.945841261531482	3	mpca	DL4	-
Oversampled	0.87189646061180445	0.6411298720938551	0.870925393891987	2	mpca	DL4	-

Nota. La figura muestra la tabulación de los resultados de aprendizaje profundo para el problema 1 con mapa de color. Elaboración propia, realizado con Python.

Tabla 17.*Tabulación de resultados DL Problema 2*

DatosEntrenamiento	Accuracy	F1 Score Macro	MPCA	DeepLayers	Threshold	Experiment	Extra
Oversampled	0.7981506482920243	0.5650442833896664	0.727644973672779	2	acc	DL	-
Oversampled	0.9400988548029651	0.6053183208405144	0.5840676823794997	2	acc	DL	Training
Oversampled	0.7647935400056041	0.5441308620789622	0.7239266326902871	2	acc	DL	Undersampled
Oversampled	0.7317803194334769	0.5243484525120792	0.7174290725680994	3	acc	DL	-
Oversampled	0.93778179687699	0.6129582660397791	0.5948064499454767	3	acc	DL	Training
Oversampled	0.7409888682273225	0.5298705735726051	0.7197935273903496	3	acc	DL	Undersampled
Undersampled	0.0515449473979163	0.049146816903505405	0.5002237838633072	2	acc	DL	-
Oversampled	0.7981506482920243	0.5650442833896664	0.727644973672779	2	f1	DL	-
Oversampled	0.9259240390248873	0.6317612132181696	0.6372283405550535	2	f1	DL	Training
Oversampled	0.7708052067147261	0.5476645569502707	0.7242516846538694	2	f1	DL	Undersampled
Oversampled	0.7791221947678121	0.5528444127686719	0.7254359622780675	3	f1	DL	-
Oversampled	0.9269557021677662	0.6312501217403463	0.6345746193064962	3	f1	DL	Training
Oversampled	0.7409888682273225	0.5298705735726051	0.7197935273903496	3	f1	DL	Undersampled
Undersampled	0.5129913136510686	0.38508006014296464	0.5347900035089708	2	f1	DL	-
Oversampled	0.7981506482920243	0.5650442833896664	0.727644973672779	2	mpca	DL	-
Oversampled	0.7981506482920243	0.5650442833896664	0.727644973672779	2	mpca	DL	Training
Oversampled	0.7708052067147261	0.5476645569502707	0.7242516846538694	2	mpca	DL	Undersampled
Oversampled	0.8107980742288	0.57301587187401	0.727202873181026	3	mpca	DL	-
Oversampled	0.7791221947678121	0.5528444127686719	0.7254359622780675	3	mpca	DL	Training
Oversampled	0.757198005451257	0.5394432866866053	0.7224076111861614	3	mpca	DL	Undersampled
Undersampled	0.938558728379652	0.5036543264340668	0.506048983528712	2	mpca	DL	-
Oversampled	0.8978526122729704	0.6117764080509438	0.6563067536255223	2	acc	DL2	-
Oversampled	0.9108184527600173	0.601694899167575	0.6186131513837572	3	acc	DL2	-
Undersampled	0.8284764500598619	0.5796572498699967	0.7111751771632606	2	acc	DL2	-
Oversampled	0.8978526122729704	0.6117764080509438	0.6563067536255223	2	f1	DL2	-
Oversampled	0.9108184527600173	0.601694899167575	0.6186131513837572	3	f1	DL2	-
Oversampled	0.8284764500598619	0.5796572498699967	0.7111751771632606	3	f1	DL2	-
Oversampled	0.8284764500598619	0.5796572498699967	0.7111751771632606	2	mpca	DL2	-
Oversampled	0.8978526122729704	0.6117764080509438	0.6563067536255223	2	mpca	DL2	-
Oversampled	0.9108184527600173	0.601694899167575	0.6186131513837572	3	mpca	DL2	-
Undersampled	0.8040604223450595	0.5631019646607217	0.7090883087655679	2	mpca	DL2	-
Oversampled	0.7566676006826808	0.5383454217904518	0.7192905417830939	2	acc	DL3	-
Oversampled	0.7045877168403087	0.504426335793626	0.6922094636841409	3	acc	DL3	-
Oversampled	0.299564408895228	0.2624880053494417	0.5080876858921484	2	acc	DL3	-
Oversampled	0.7566676006826808	0.5383454217904518	0.7192905417830939	2	f1	DL3	-
Oversampled	0.7045877168403087	0.504426335793626	0.6922094636841409	3	f1	DL3	-
Undersampled	0.7182668059199633	0.4773816402790984	0.5613434934437913	2	fl	DL3	-
Oversampled	0.7566676006826808	0.5383454217904518	0.7192905417830939	2	mpca	DL3	-
Oversampled	0.7045877168403087	0.504426335793626	0.6922094636841409	3	mpca	DL3	-
Oversampled	0.7182668059199633	0.4773816402790984	0.5613434934437913	2	mpca	DL3	-
Oversampled	0.9139134421886542	0.6043523291192352	0.6179937385250196	2	acc	DL4	-
Oversampled	0.9238225029930968	0.6244157505827241	0.6306741199988469	3	acc	DL4	-
Undersampled	0.8497210688539624	0.5936532055737349	0.7074467714984101	2	acc	DL4	-
Oversampled	0.9139134421886542	0.6043523291192352	0.6179937385250196	2	f1	DL4	-
Oversampled	0.9238225029930968	0.6244157505827241	0.6306741199988469	3	f1	DL4	-
Undersampled	0.8463586112030975	0.5914249954639955	0.708280524814785	2	f1	DL4	-
Oversampled	0.9139134421886542	0.6043523291192352	0.6179937385250196	2	mpca	DL4	-
Oversampled	0.9238225029930968	0.6244157505827241	0.6306741199988469	3	mpca	DL4	-
Undersampled	0.8463586112030975	0.5914249954639955	0.708280524814785	2	mpca	DL4	-

Nota. La figura muestra la tabulación de los resultados de aprendizaje profundo para el problema 2 con mapa de color. Elaboración propia, realizado con Python.

5.4. Comparación de resultados

Luego de la anterior tabulación de los resultados es el momento de seleccionar aquellas variaciones de los algoritmos que de mejor manera dieron solución a los problemas de interés. Como se mencionó antes, se seleccionarán los mejores modelos según su desempeño en cada uno de los conjuntos de prueba. Las tres métricas principales fueron Accuracy (exactitud general), F1 Score Macro y MPCA (exactitud media por clase).

Es importante mencionar que una métrica puede ser mayor para un algoritmo que para otro sin que eso implique necesariamente que proporciona una mejor solución al problema de interés. El desbalance de clases de los conjuntos de prueba se vislumbra como la causa de que esto sea así, pese a eso se tomó la decisión de diseño de no alterar los conjuntos de prueba a modo de tener clases balanceadas, la justificación es que, si las clases están desbalanceadas de manera natural, la utilización de los modelos en un escenario realista será también para predecir más frecuentemente equipos de la etiqueta mayoritaria.

Se busca por lo tanto un valor alto de cada una de las métricas, pero con la particularidad adicional de que brinde un nivel razonable de acierto en la predicción de ambas etiquetas. Idealmente debería poder decirse que un modelo tiene determinado nivel de acierto en general, y que la predicción será confiable tanto en la una como en la otra etiqueta. La visualización de la matriz de confusión normalizada resulta especialmente útil para este cometido, tal que la asignación de colores más oscuros en la diagonal de arriba izquierda a abajo derecha (TN y TP) es señal de un resultado mejor. Por lo anterior, además de la selección de los modelos según su métrica, se escogerán también las instancias de los algoritmos que por inspección constituyan una buena solución, donde se considere necesario.

5.4.1. Comparación de modelos de *machine learning*

Se discutirá en breve, pero parece que los algoritmos de *machine learning* tuvieron cierto nivel de sobreajuste, de modo que una solución adecuada para el problema 1, de manera general arrojó resultados insatisfactorios para el problema 2, y, pese a utilizar métricas con la finalidad de resolver el problema 2, los resultados estuvieron cercanos a los menos sobresalientes en *deep learning*.

Es también necesario recordar que la búsqueda aleatoria con una cantidad limitada de instancias de los modelos no abarcó ni por asomo la totalidad del espacio de hiper parámetros, donde existe la posibilidad de que se encuentren soluciones mejores a cada uno de los dos problemas. Otra situación relacionada con estos modelos de ML es que no se realizaron búsquedas aleatorias para todos los algoritmos dentro de todos los experimentos, ni para todos los conjuntos de datos disponibles de entrenamiento, esto principalmente debido a limitaciones en la capacidad computacional de que se dispuso.

5.4.1.1. Mejores modelos de ML para problema 1

A continuación, se detallan los modelos que mejor resolvieron el problema 1 dentro de los experimentos de *machine learning* realizados. Debido a los buenos resultados en este problema, no fue necesaria la selección por inspección. La selección de estos modelos se encuentra en el cuaderno *summary_ml.ipynb*, disponible en el repositorio del proyecto.

Tabla 18.

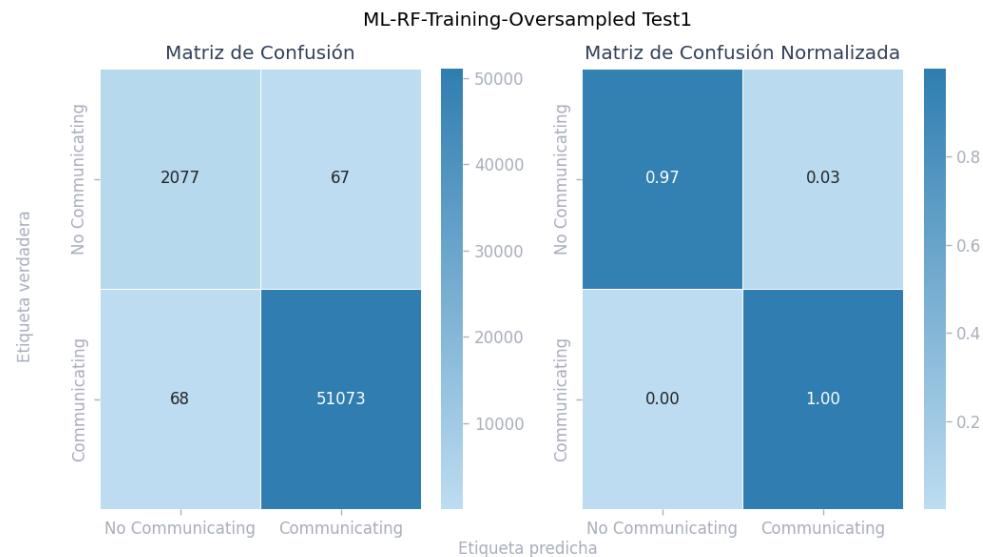
Mejor modelo de ML para P1 según Accuracy

Algoritmo	Datos	Accurac y	F1 Score	MPCA	Experiment o
	Entrenamiento		Macro		
Random Forest	Training Oversampled	0.997466	0.983602	0.9837 1	<i>Machine Learning</i>

Nota. La tabla muestra el mejor modelo de *Machine Learning* según Accuracy para el problema 1. Elaboración propia, realizado con Word.

Figura 91.

Mejor matriz de confusión ML para P1 según Accuracy



Nota. La figura muestra la visualización para el mejor modelo de *Machine Learning* según Accuracy para el problema 1. Elaboración propia, realizado con Python.

Tabla 19.

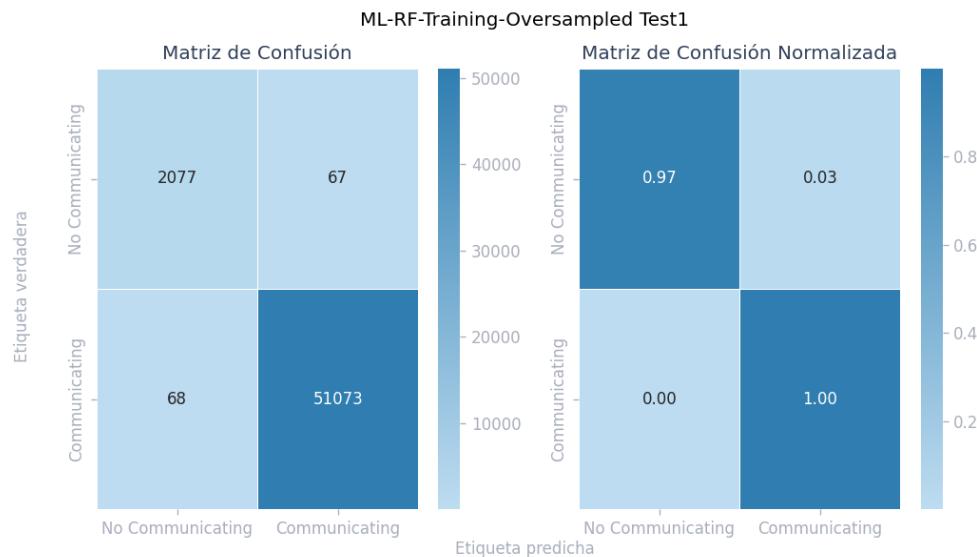
Mejor modelo de ML para P1 según F1 Score Macro

Algoritmo	Datos entrenamiento	Accuracy	F1 Score macro	MPCA	Experimento
Random Forest	Training Oversampled	0.997466	0.983602	0.98371	Machine Learning

Nota. La tabla muestra el mejor modelo de *Machine Learning* según F1 Score Macro para el problema 1. Elaboración propia, realizado con Excel.

Figura 92.

Mejor matriz de confusión ML para P1 según F1 Score Macro



Nota. La figura muestra la visualización para el mejor modelo de *Machine Learning* según F1 Score Macro para el problema 1. Elaboración propia, realizado con Python.

Tabla 20.

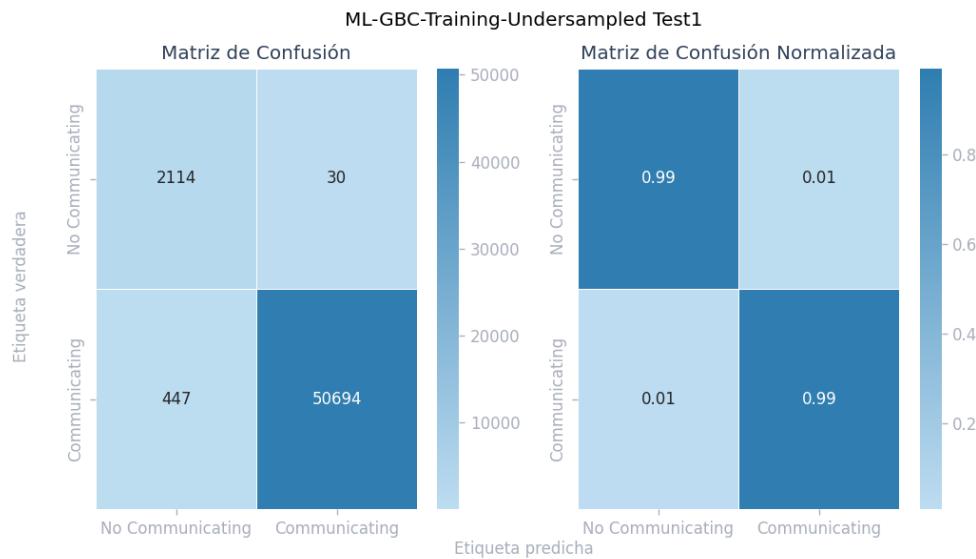
Mejor modelo de ML para P1 según MPCa

Algoritmo	Datos	Accuracy	F1 Score	MPCA	Experimento
					Entrenamiento
Gradient Boost Classifier	Training Undersampled	0.991048	0.946968	0.988633	<i>Machine Learning</i>

Nota. La tabla muestra el mejor modelo de *Machine Learning* según MPCa para el problema 1. Elaboración propia, realizado con Excel.

Figura 93.

Mejor matriz de confusión ML para P1 según MPCA



Nota. La figura muestra la visualización para el mejor modelo de *machine learning* según MPCA para el problema 1. Elaboración propia, realizado con Python.

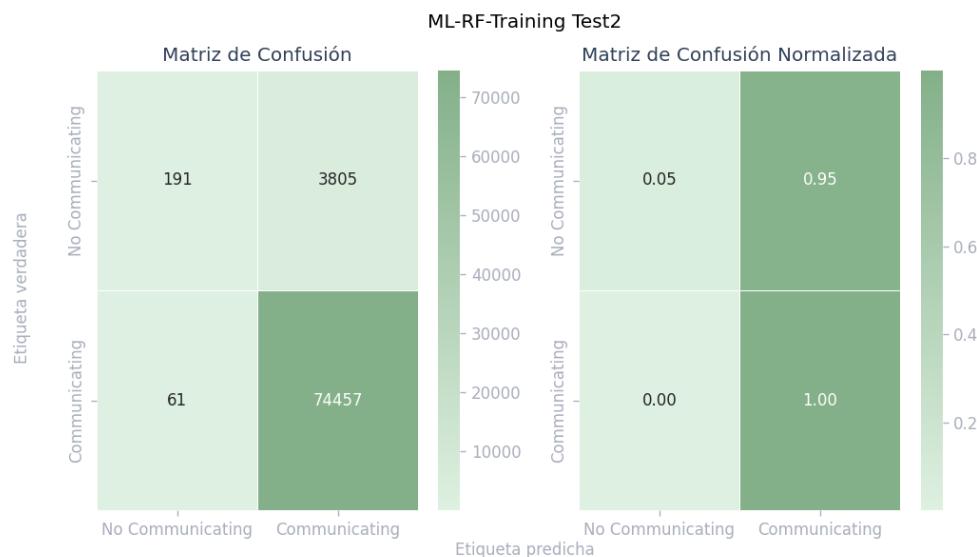
5.4.1.2. Mejores modelos de ML para problema 2

A continuación, se detallan los modelos que mejor resolvieron el problema 2 dentro de los experimentos de *machine learning* realizados. La selección de estos modelos se encuentra en el cuaderno `summary_ml.ipynb`. A pesar de que los resultados no fueron precisamente los esperados, al realizar una inspección se determinó que la selección por MPCA contiene la mejor solución posible al problema en cuestión utilizando el paradigma de *machine learning*.

Tabla 21.*Mejor modelo de ML para P2 según Accuracy*

Algoritmo	Datos	Accuracy	F1 Score	MPCA	Experimento
	Entrenamiento		Macro		
Random Forest	Training	0.95076	0.53231	0.52349	<i>Machine Learning</i>

Nota. La tabla muestra el mejor modelo de *Machine Learning* según Accuracy para el problema 2. Elaboración propia, realizado con Excel.

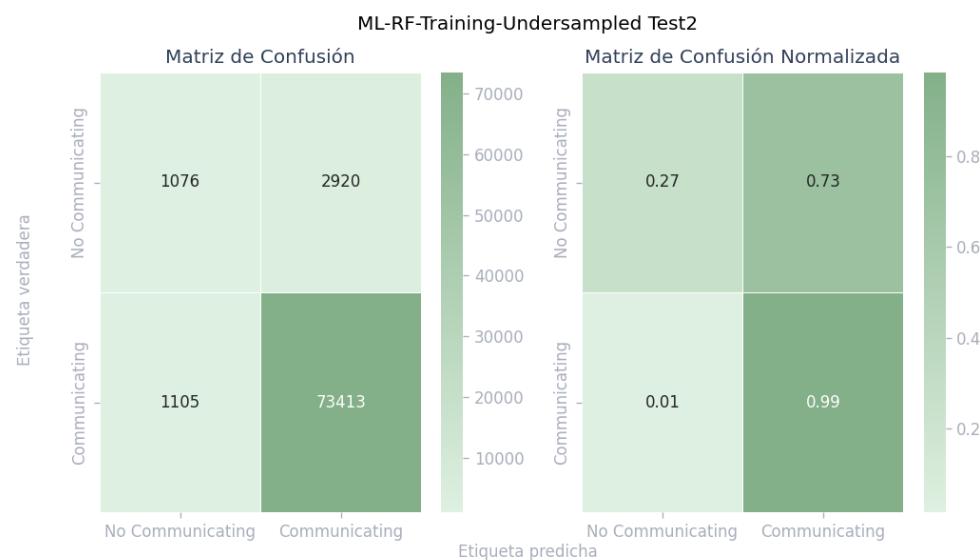
Figura 94.*Mejor matriz de confusión ML para P2 según Accuracy*

Nota. La figura muestra la visualización para el mejor modelo de *Machine Learning* según Accuracy para el problema 2. Elaboración propia, realizado con Python.

Tabla 22.*Mejor modelo de ML para P2 según F1 Score Macro*

Algoritmo	Datos entrenamiento	Accurac y	F1 Score Macro	MPCA	Experiment o
Random Forest	Training	0.94873	0.660854	0.6272	<i>Machine Learning</i>
	Undersampled	5		2	

Nota. La tabla muestra el mejor modelo de *Machine Learning* según F1 Score Macro para el problema 2. Elaboración propia, realizado con Excel.

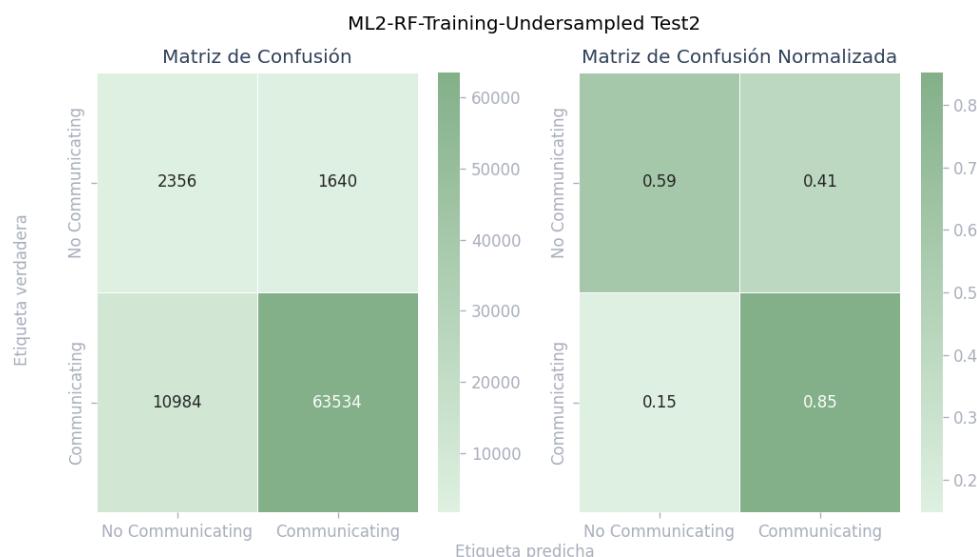
Figura 95.*Mejor matriz de confusión ML para P2 según F1 Score Macro*

Nota. La figura muestra la visualización para el mejor modelo de *machine learning* según F1 Score Macro para el problema 2. Elaboración propia, realizado con Python.

Tabla 23.*Mejor modelo de ML para P2 según MPCA*

Algoritmo	Datos	Accuracy	F1 Score	MPCA	Experimento
	Entrenamiento		Macro		o
Random Forest	Training Undersampled	0.839213	0.590717	0.721094	<i>Machine Learning 2</i>

Nota. La tabla muestra el mejor modelo de *Machine Learning* según MPCA para el problema 2. Elaboración propia.

Figura 96.*Mejor matriz de confusión ML para P2 según MPCA*

Nota. La figura muestra la visualización para el mejor modelo de *Machine Learning* según MPCA para el problema 2. Elaboración propia, realizado con Python.

5.4.2. Comparación de modelos de *Deep Learning*

Se pudo apreciar una mejora en los resultados obtenidos con la implementación de las redes neuronales que comprendieron los distintos experimentos de aprendizaje profundo (DL). Es posible decir que se obtuvo resultados satisfactorios para ambos problemas, lo que confirma la intuición inicial de que este paradigma del aprendizaje de máquina tiene mucho por ofrecer.

Una situación particular que merece ser resaltada es que debido a un error durante la ejecución de los experimentos se encontró una forma de mejorar los resultados de un modelo ya entrenado al momento de encontrar la mejor opción para el umbral de discretización de la etiqueta, dada la predicción continua de una red neuronal. Esto se indica en las tablas y visualizaciones porque el nombre contiene la palabra *extra* seguido del nombre de uno de los conjuntos de datos de entrenamiento. Sucede que, a pesar de entrenar un modelo con un conjunto de datos particular, es posible aumentar el nivel de acierto de la clasificación al realizar la tarea de selección del umbral de discretización con un conjunto de datos de validación distinto del conjunto de datos de validación durante el entrenamiento y selección de hiper parámetros.

Acá también es importante mencionar que la función de selección de hiper parámetros abarcó la totalidad de las posibles iteraciones contenidas en la configuración inicial de la clase heredada del HyperModel de KerasTuner. Esto requirió poder computacional adicional que fue comprado en una nube privada, así como el uso de la función de tuneo que optimiza los recursos, perteneciente a la librería mencionada. En este caso lo que no abarcó la totalidad de posibilidades fue la selección del umbral de discretización. Se procedió eligiendo los mejores modelos según la selección del umbral convencional (mismo

conjunto de validación que en el entrenamiento), y a estos mejores modelos se les aplicó la selección de umbral cruzada (distinto conjunto de validación que en el entrenamiento), mejorando con esto los resultados previos.

5.4.2.1. Mejores modelos de DL para problema 1

A continuación, se detallan los modelos que mejor dieron solución al problema 1 dentro de los experimentos de *deep learning* realizados. La selección de estos modelos se encuentra en el cuaderno summary_dl.ipynb. En este caso particular se tiene que el mismo modelo contuvo el máximo para todas las métricas, más aún, el umbral de discretización fue el mismo para todas las métricas.

Tabla 24.

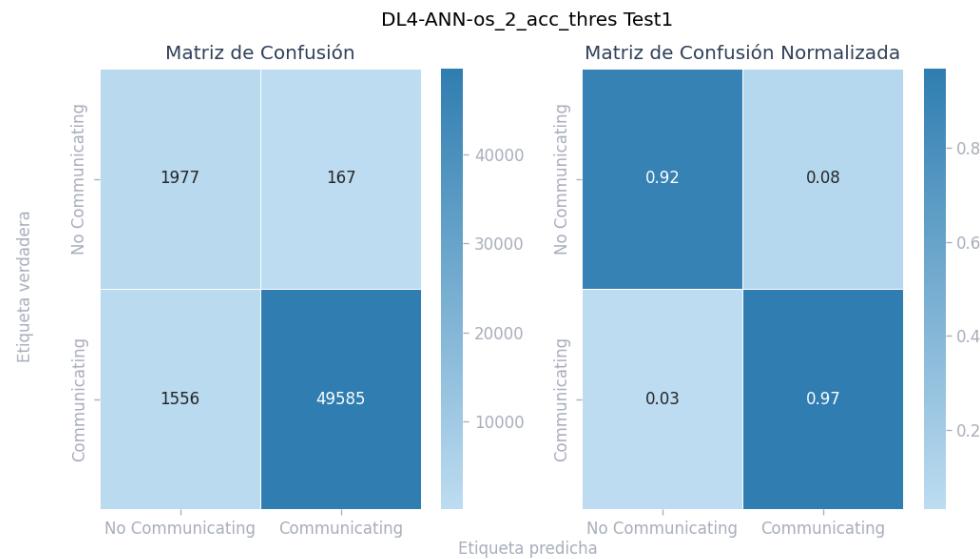
Mejor modelo de DL para P1 según Accuracy

Algoritmo	Datos entrenamiento	Accuracy	F1 Score	MPCA	Experime nto
			Macro		
ANN 3 Layers	Training	0.967664	0.839709	0.945841	<i>Deep</i>
Threshold	Oversampled				<i>Learning 4</i>
ACC,F1,MPCA					

Nota. La tabla muestra el mejor modelo de *deep learning* según Accuracy para el problema 1. Elaboración propia, realizado con Excel.

Figura 97.

Mejor matriz de confusión DL para P1 según Accuracy



Nota. La figura muestra la visualización para el mejor modelo de *deep learning* según Accuracy para el problema 1. Elaboración propia, realizado con Python.

Tabla 25.

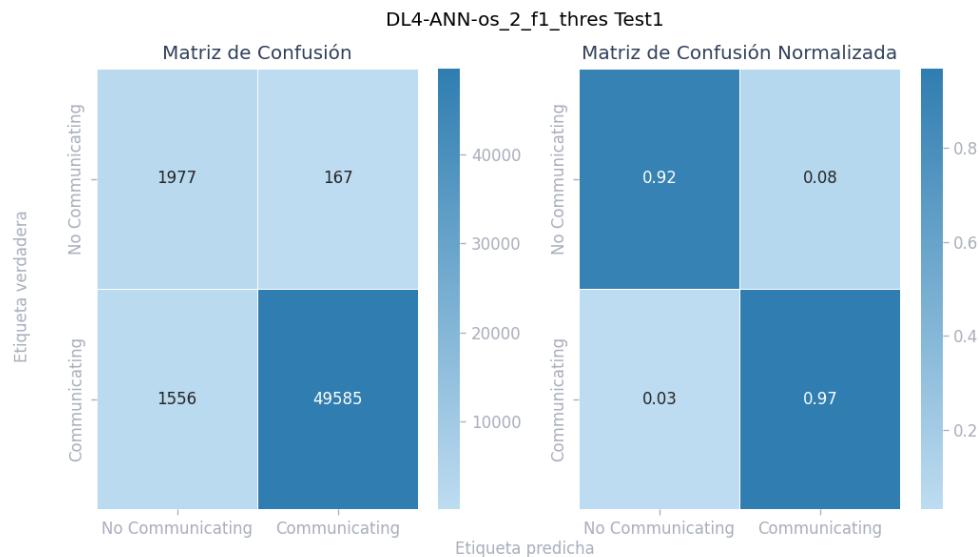
Mejor modelo de DL para P1 según F1 Score Macro

Algoritmo	Datos	Accuracy	F1 Score	MPCA	Experimento
			Macro		
ANN 3 Layers	Training	0.967664	0.839709	0.945841	<i>Deep Learning 4</i>
ANN 3 Layers	Oversampled				
Threshold					
ACC,F1,MPCA					

Nota. La tabla muestra el mejor modelo de *deep learning* según F1 Score Macro para el problema 1. Elaboración propia, realizado con Excel.

Figura 98.

Mejor matriz de confusión DL para P1 según F1 Score Macro



Nota. La figura muestra la visualización para el mejor modelo de *deep learning* según F1 Score Macro para el problema 1. Elaboración propia, realizado con Python.

Tabla 26.

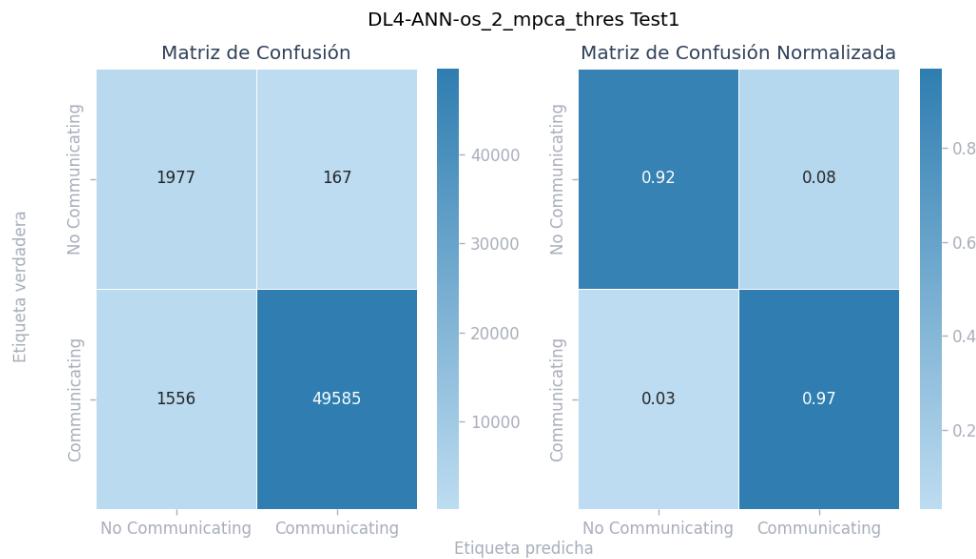
Mejor modelo de DL para P1 según MPCA

Algoritmo	Datos	Accuracy	F1 Score	MPCA	Experimento
Entrenamiento		Macro			
ANN 3 Layers	Training	0.967664	0.839709	0.945841	<i>Deep Learning</i>
Threshold	Oversampled				4
ACC,F1,MPCA					

Nota. La tabla muestra el mejor modelo de *Deep Learning* según MPCA para el problema 1. Elaboración propia, realizado con Excel.

Figura 99.

Mejor matriz de confusión DL para P1 según MPCA



Nota. La figura muestra la visualización para el mejor modelo de *Deep Learning* según MPCA para el problema 1. Elaboración propia, realizado con Python.

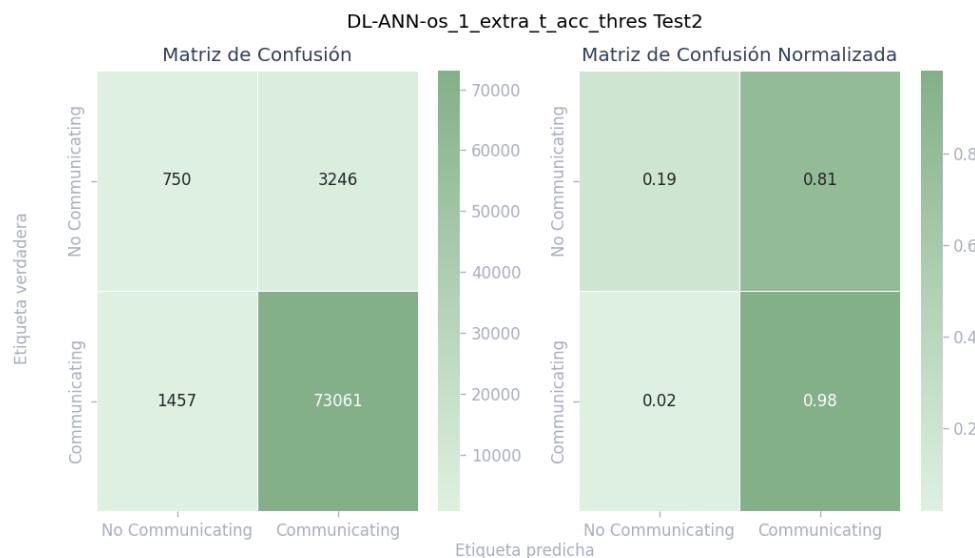
5.4.2.2. Mejores modelos de DL para problema 2

A continuación, se detallan los modelos que mejor dieron solución al problema 2 dentro de los experimentos de *deep learning* realizados. La selección de estos modelos se encuentra en el cuaderno `summary_dl.ipynb`. Es importante mencionar que más allá de las métricas, se consideró primordial que la exactitud fuese similar para las dos etiquetas.

Tabla 27.*Mejor modelo de DL para P2 según Accuracy*

Algoritmo	Datos Entrenamiento	Accuracy	F1 Score Macro	MPCA	Experimento				
					Macro				
ANN 2									
<i>Layers</i>									
Extra Training	Training Oversampled	0.9401	0.605318	0.584068	Deep Learning				
<i>Threshold</i>									
<i>Accuracy</i>									

Nota. La tabla muestra el mejor modelo de *Deep Learning* según Accuracy para el problema 2. Elaboración propia, realizado con Word.

Figura 100.*Mejor matriz de confusión DL para P2 según Accuracy*

Nota. La figura muestra la visualización para el mejor modelo de *Deep Learning* según Accuracy para el problema 2. Elaboración propia, realizado con Python.

Tabla 28.

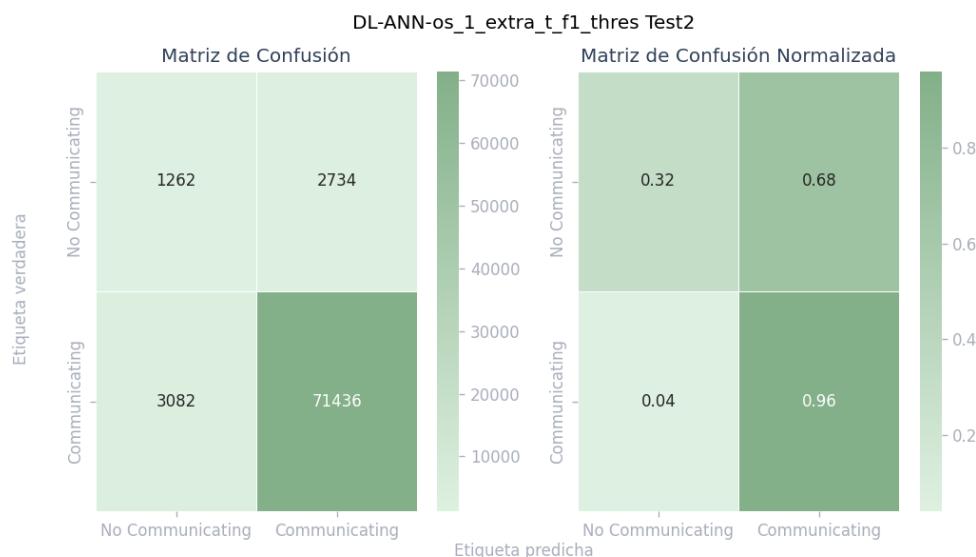
Mejor modelo de DL para P2 según F1 Score Macro

Algoritmo	Datos	Accuracy	F1 Score	MPCA	Experieme
	Entrenamiento		Macro		nto
ANN 2 Layers Extra Training Threshold F1 Score	Training Oversampled	0.925924	0.631761	0.637228	<i>Deep Learning</i>

Nota. La tabla muestra el mejor modelo de *deep learning* según F1 Score Macro para el problema 2. Elaboración propia, realizado con Word.

Figura 101.

Mejor matriz de confusión DL para P2 según F1 Score Macro

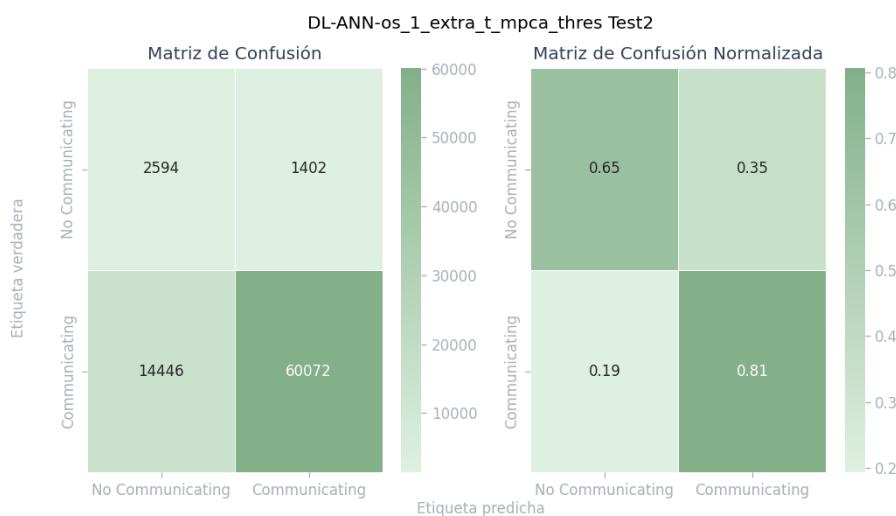


Nota. La figura muestra la visualización para el mejor modelo de *deep learning* según F1 Score Macro para el problema 2. Elaboración propia, realizado con Python.

Tabla 29.*Mejor modelo de DL para P2 según MPCA*

Algoritmo	Datos Entrenamiento	Accuracy	F1 Score	MPCA	Experiemento
			Macro		
ANN 2 Layers					
Extra Training	Training	0.798151	0.565044	0.727645	<i>Deep Learning</i>
Threshold					
MPCA	Oversampled				
ANN 2 Layers					
Threshold	Training	0.798151	0.565044	0.727645	<i>Deep Learning</i>
ACC,F1,MPCA					
	Oversampled				

Nota. La tabla muestra los mejores modelos de *Deep Learning* según MPCA para el problema 2. Elaboración propia, realizado con Word.

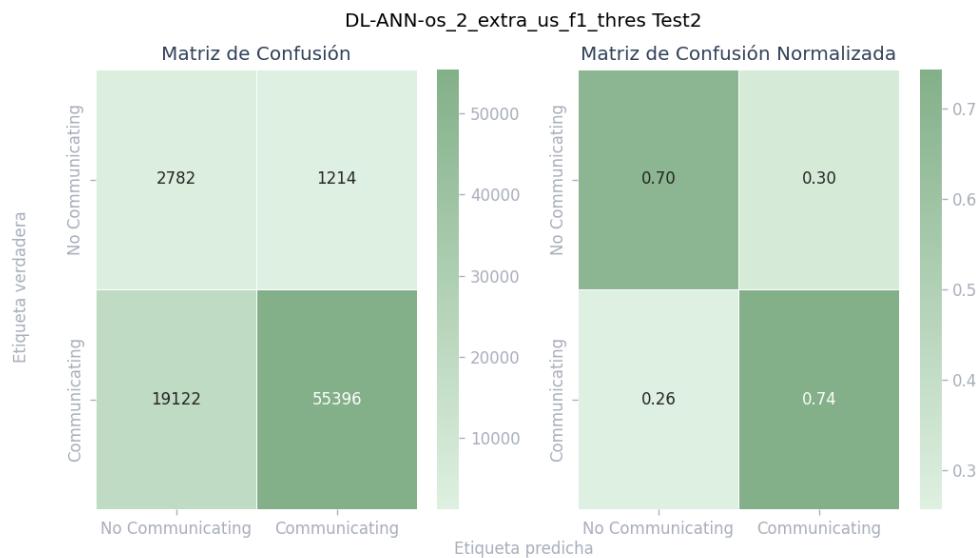
Figura 102.*Mejor matriz de confusión DL para P2 según MPCA*

Nota. La figura muestra la visualización para el mejor modelo de *Deep Learning* según MPCA para el problema 2. Elaboración propia, realizado con Python.

Tabla 30.*Mejor modelo de DL para P2 según inspección*

Algoritmo	Datos	Accuracy	F1 Score	MPCA	Experimento
	Entrenamiento		Macro		
ANN 3					
Layers Extra	Training	0.740989	0.529871	0.719793	<i>Deep Learning</i>
Training					
Threshold F1					

Nota. La tabla muestra el mejor modelo de *deep learning* según inspección para el problema 2. Elaboración propia, realizado con Word.

Figura 103.*Mejor matriz de confusión DL para P2 según inspección*

Nota. La figura muestra la visualización para el mejor modelo de *deep learning* según inspección para el problema 2. Elaboración propia, realizado con Python.

5.4.3. Comparación general

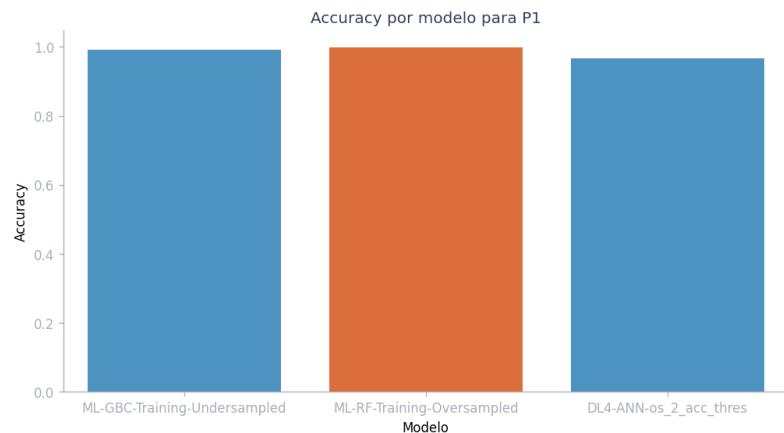
Dentro de la gran cantidad de instancias de modelos que se entrenaron durante los distintos experimentos, existen aquellos que son sobresalientes. Identificar estos en la sección anterior fue la segunda etapa de un reordenamiento y almacenamiento siguiendo ciertos estándares que permitieron una tabulación adecuada. Esto deNota la importancia de lo que se ha venido a nombrar como MLOps, operaciones de *machine learning*. En MLOps existen varias etapas, y en los alcances de este proyecto se llega solo hasta la etapa de revisión, que es previa al despliegue de los modelos en producción, de modo que los usuarios finales puedan hacer uso de estos. Aquí, se comparan de manera general los modelos seleccionados por cada paradigma de aprendizaje de máquina, y se seleccionan los 2 aptos para ser presentados como la solución a los problemas. Esta comparación se encuentra en el repositorio del proyecto en el cuaderno general_comparation.ipynb.

5.4.3.1. Comparación general problema 1

Como se mencionó antes, el problema 1 es el que presentó menor complicación a la hora de encontrar un resultado satisfactorio. Esto debido, según se interpreta, a la ventaja significativa de tener registros históricos sobre los distintos equipos para predecir cuál será su comportamiento futuro. Según las métricas mencionadas de interés, se determinó que en todos los casos los mejores resultados se dieron en los modelos de *Machine Learning*, más precisamente en aquellos que implican el uso de árboles de decisión, como RF y GBC. Para este problema tanto ML como DL arrojaron resultados satisfactorios, y dado que no parece haber una afectación debido al desbalance de clases en el conjunto de datos de prueba 1, la métrica Accuracy parece ser adecuada para la selección del mejor modelo.

Figura 104.

Gráfica Accuracy para problema 1

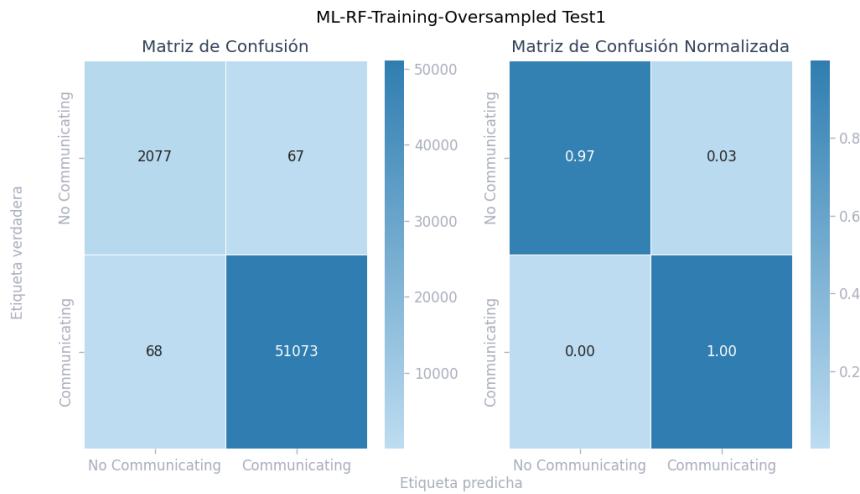


Nota. El gráfico muestra la comparación general según Accuracy para el problema 1. Elaboración propia, realizado con Python.

Aunque los resultados obtenidos con *deep learning* fueron aceptables el mejor modelo para solucionar este problema 1 no implica el uso de redes neuronales. En la sección de apéndices se pueden apreciar las gráficas generales de comparación, siendo que el modelo ML-RF-Training-Oversampled tuvo mejor resultado para las métricas Accuracy y F1 Score Macro, se propone este como solución para el problema de determinar el estado de la comunicación dado el historial de comunicación en instancias temporales previas. Su matriz de confusión se presenta a continuación.

Figura 105.

Solución al problema 1



Nota. El gráfico muestra la visualización de matrices de confusión para el modelo propuesto como solución del problema 1. Elaboración propia, realizado con Python.

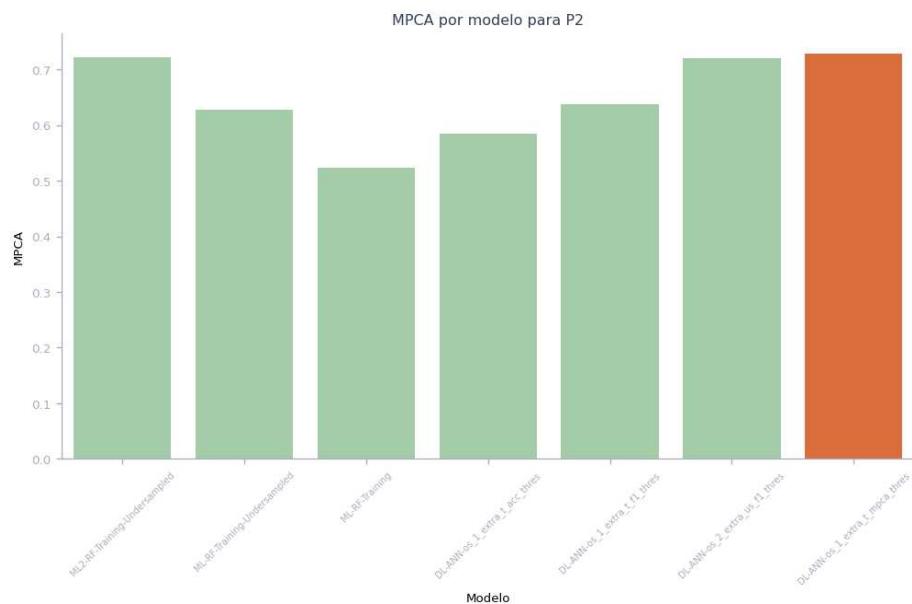
5.4.3.2. Comparación general problema 2

Como ya se ha mencionado, este problema fue el que presentó mayor complicación al momento de conseguir resultados satisfactorios, y la mayoría de los experimentos tanto de ML como de DL se enfocaron en encontrar una solución a este problema. En las gráficas comparativas presentadas en los apéndices se evidencia que fueron modelos de ML los que tuvieron el mayor valor para las métricas Accuracy y F1 Score Macro, esto debido al desbalance en los conjuntos de datos de testeo. Estos modelos, aunque con métricas mayores, carecieron del balance en la exactitud de las predicciones de las dos etiquetas, es decir, predijeron de muy buena manera los equipos que con la etiqueta Sí comunica, con resultados bastante pobres para la etiqueta no comunica.

De cara a presentar una solución se requiere que la predicción tenga buenos resultados para ambas etiquetas. Por tal motivo fue de significativa importancia la selección según la métrica MPCA, que busca que la media de la exactitud de las dos etiquetas sea la máxima. En la siguiente imagen se evidencia un rendimiento significativamente mejor para esta métrica en los modelos de *deep learning*, y verificando minuciosamente las matrices de confusión también se evidencia que los valores altos de esta métrica en *machine learning* fue de nuevo, causado por el desbalance de clases en los conjuntos de prueba.

Figura 106.

Gráfica MPCA para problema 2



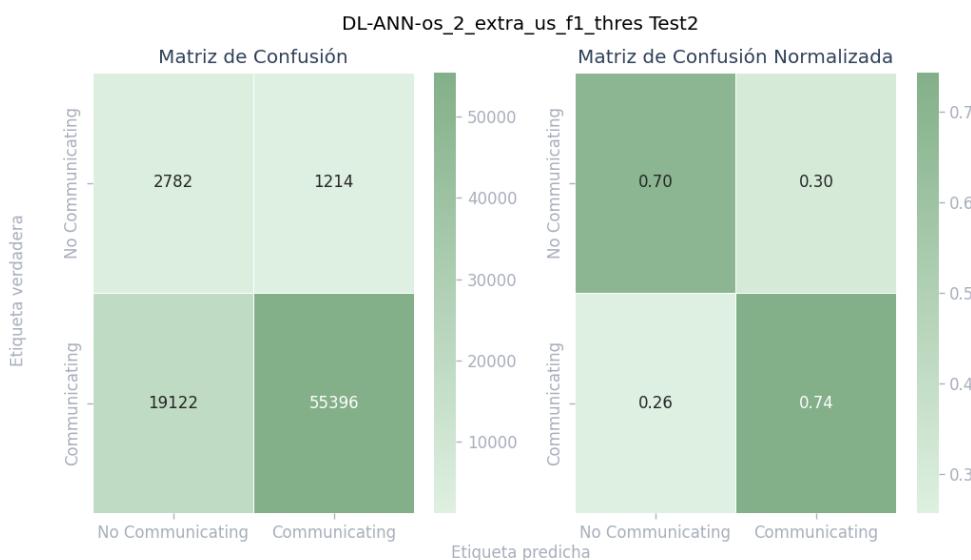
Nota. El gráfico muestra la comparación general según MPCA para el problema 2. Elaboración propia, realizado con Python.

En general los mejores resultados fueron haciendo uso de las Redes Neuronales Artificiales (ANN). Además, aunque el mejor resultado fue con el modelo `DL-ANN-os_1_extra_t_mpca_thres`, el cual brinda una solución

aceptable al problema, se tomó la decisión de presentar como la solución para este segundo problema, el modelo DL-ANN-os_2_extra_us_f1_thres, debido al balance en la exactitud de la predicción de ambas métricas, este fue seleccionado por inspección y no por maximizar una métrica específica. Se presenta a continuación su matriz de confusión.

Figura 107.

Solución al problema 2



Nota. El gráfico muestra la visualización de matrices de confusión para el modelo propuesto como solución del problema 2. Elaboración propia, realizado con Python.

5.4.4. Análisis de resultados

Como se indicó antes, la interpretación de los resultados obtenidos es meramente un ejercicio de búsqueda de verosimilitud y no debe ser tomado en cuenta como una verdadera explicación de lo que sucede dentro de los algoritmos. Sin embargo, es posible aventurarse a un análisis de lo que las

métricas y gráficas pueden significar, dado el constante afán humano de encontrarle un sentido a las cosas.

Resolver el problema 1 dada la existencia de los datos históricos es, podría decirse, un ejercicio de memorización para los modelos, de modo que sepan que determinados equipos en relación con determinados vecinos mantendrán su estado de comunicación a través del tiempo. El hecho de que los experimentos ML y DL4 fuesen los que arrojara los mejores resultados para este primer problema puede deberse a una validación durante el entrenamiento que no discriminó por equipos, de modo que entre los *folds* o el conjunto de validación usados durante el entrenamiento y selección de hiper parámetros, se cumplió la misma situación que entre los conjunto de datos de entrenamiento y el conjunto prueba del problema 1, es decir, había equipos repetidos (aunque en distintas instancias de tiempo). Una situación resaltable es que, tanto para ML como para DL, los mejores modelos para el problema 1, arrojaron sistemáticamente resultados pobres para el problema 2 (véanse los apéndices para comprobarlo).

Encontrar solución al problema 2 significó un reto mayor, dado que este conjunto de datos de prueba tiene equipos nunca vistos por los modelos durante su etapa de entrenamiento y selección de hiper parámetros. Acá la tarea no es solo de memorizar los equipos que comunican y los que no, sino que el modelo debe encontrar patrones generales asociados a cada una de las etiquetas, dado que deberá ser evaluado con equipos nuevos tanto en el espacio como en el tiempo. De nuevo, parece que la estrategia de selección de los *folds* o el conjunto de datos de validación influyó en la solución debido a la similitud de estos con el conjunto de datos de prueba 2, es decir, al discriminar por equipos durante el entrenamiento y selección de hiper parámetros se evaluaron los modelos con equipos nunca vistos. Es importante resaltar que a diferencia de la solución para

el problema 1, las soluciones aventajadas para el problema 2 dieron también una solución aceptable al problema 1 (véanse los apéndices para comprobarlo).

En ambos casos parece ser importante el balanceo de los conjuntos de datos de entrenamiento. A pesar de que en DL solo se utilizaron conjuntos de entrenamiento balanceados, en ML se pudo apreciar una preferencia de los modelos por los conjuntos de datos balanceados, incluso si la evaluación final se realizó sobre conjuntos de prueba desbalanceados. Relacionado también con el balanceo de las clases, durante la selección del umbral de discretización se apreció una mejora con las soluciones mencionadas como —extras—, esto puede deberse a una mayor semejanza de los conjuntos de selección de umbral con los conjuntos de datos de prueba, entiéndase, el conjunto de datos de validación de training que estaba desbalanceado y el conjunto de validación de sub muestreo, que a pesar de estar balanceado, no requirió la necesidad de añadir variantes sintéticas más allá de las primeras generadas, es decir, existía mayor semejanza entre los equipos resultado del sub muestreo que de los equipos fabricados con el sobre muestreo.

Entre los conjuntos de datos de entrenamiento, sin lugar a duda el que tuvo el mejor desempeño durante el entrenamiento y selección de hiper parámetros fue el conjunto de datos de sobre muestreo, recalmando lo que ya es bien conocido de ML y DL, que la cantidad de datos importa, y evidentemente este fue el conjunto de datos que mayor cantidad de registros aportó a los modelos para su entrenamiento y selección de hiper parámetros.

Se planteó este enfoque de solución al problema pertinente como haciendo uso de inteligencia artificial. La realidad es que a pesar del considerable esfuerzo humano que involucró, no fue en ningún momento necesario plantear de manera explícita una solución. La etapa de búsqueda de los hiper parámetros

fue hasta cierto punto a ciegas, dado que la evaluación de los modelos y la visualización de los resultados en una manera entendible (la visualización de la matriz de confusión), fue un proceso que siguió a la ejecución de todos los experimentos, esto se puede apreciar en que en la mayoría de los casos, las soluciones buscadas se encontraron en los primeros experimentos planteados, de modo que bien pudieron haber sido omitidos los demás experimentos. Pero, como la solución en última instancia no fue vislumbrada por el humano, esto hace sentido y confirma el paradigma de este trabajo de graduación, de que una máquina está en condición de resolver un problema, o incluso dos, como en este trabajo.

Como un comentario final se menciona la significativa diferencia de enfrentarse con un conjunto de datos extraídos directamente de un problema del mundo real en contraposición a los conjuntos de datos ad hoc que se utilizan en el aprendizaje de la ciencia de datos. La primera situación a diferencia de la segunda requiere librarse del primer obstáculo de la extracción y limpieza de los datos, así como el segundo obstáculo de estar trabajando con datos que no tienen por qué y definitivamente no son a propósito amigables con los algoritmos de aprendizaje de máquina.

CONCLUSIONES

1. La inteligencia artificial es tanto un concepto actual como un mito antiguo. No parece clara, ni siquiera hoy, dónde está la barrera entre lo uno y lo otro. A lo largo de la literatura referida en este trabajo se evidencian distintas ideas que giran alrededor de una inteligencia creada por otra, así como distintas especulaciones al respecto. Haciendo uso también del lenguaje, cada vez más formal, se ha desarrollado el marco teórico y las técnicas que fueron la base para el desarrollo de la solución creada. Otra idea encontrada y ratificada durante la revisión de la literatura realizada, es que hoy en día el uso del término Inteligencia Artificial se refiere al Aprendizaje de Máquina.
2. Se considera que la solución presentada en este trabajo es de utilidad no solo de cara al negocio y para la red particular, sino que explora un uso del aprendizaje de máquina en relación con las redes malla, que, según se evidencia en la literatura referida en este trabajo, no se ha publicado de manera explícita el atender al problema de la cobertura para una red inalámbrica con topología malla que ya esté desplegada.
3. El sistema de obtención de datos implementado consiguió su cometido y automatizó gran parte de la labor de extracción de la información necesaria para implementar el resto de la solución. El uso de un computador de placa reducida permitió total disponibilidad del equipo para las tareas del todo automatizadas.

4. Se dice que los algoritmos de Inteligencia Artificial son los que resuelven en última instancia el problema de interés. Sin embargo, existe una cantidad significativa de labor humana. En este caso y como también se confirma en la literatura referida en este trabajo, la extracción y tratamiento de los datos requieren eminentemente la intervención humana. Estos pasos referidos son los que involucran la creación del Mercado de Datos. Toda esta labor previa permitió simplificar las etapas posteriores donde la fuente de la verdad fue el *Data Mart*.
5. Dados los datos depurados en el Mercado de Datos es posible comenzar el entrenamiento. Pero reiteradamente en la literatura referida en este trabajo y según la manera en que operan los datos los algoritmos, es preferible realizar una etapa adicional de Ingeniería de Características. En este caso se consiguió generar la característica de altimetrías, Fresnel e imágenes satelitales. Luego de esto, tanto las características originales como las agregadas fueron escaladas y transformadas en números entre 0 y 1. Esta etapa permitió añadir los conjuntos de datos enriquecidos y transformados al Mercado de Datos.
6. Aunque algunas de las ideas concebidas y comentadas en los inicios de este trabajo se mantuvieron válidas, hubo varios factores no considerados y que implicaron trabajo de experimentación. Esto sucedió en mayor medida durante la implementación de los distintos modelos. Es acá donde comienza a dispensarse de la labor humana en favor del cálculo aritmético de las máquinas. Los resultados satisfactorios obtenidos en la solución de los dos problemas planteados puede ser un inicio para soluciones cada vez más completas y que utilicen el aprendizaje de máquina para el tipo específico de redes que se atendió.

7. Para el problema 1, enfocado en la predicción del estado de comunicación de equipos según su registro histórico el algoritmo resultante arrojó una precisión media por clase rondando el 99 %. Para el problema 2, enfocado en la predicción del estado de comunicación para equipos totalmente nuevos el algoritmo resultante arrojó una precisión media por clase que ronda el 72 %. Una buena solución para el problema 2 es también una buena solución para el problema 1. Esta situación no sucede en la dirección contraria.

RECOMENDACIONES

1. Abordar la literatura referida en este trabajo para propiciar un mejor entendimiento de las ideas planteadas y las técnicas utilizadas.
2. Aumentar el tamaño de la muestra de modo que se incluyan registros más allá de los 84 días para conseguir mejores resultados.
3. Utilizar técnicas de reducción de la dimensionalidad para lidiar con la alta dimensionalidad y propiciar mejores resultados.
4. Utilizar métodos de ingeniería de características que estén específicamente enfocados en series temporales y en grafos, de modo que se consiga una mejor representación de la red con los datos.
5. Explorar distintos paradigmas del aprendizaje profundo, como con una red neuronal recurrente, para aprovechar la naturaleza secuencial del conjunto de datos.

REFERENCIAS

- Agrawal, A., Gans, J. & Goldfarb, A. (2017). What to expect from artificial intelligence [Qué esperar de la inteligencia artificial]. *MIT Sloan Management Review; Cambridge* 58(3), 23-27. <https://www.proquest.com/docview/1885859665?sourceType=Scholarly%20Journals>
- Asimov, I. (2009). *Yo, Robot*. Edhasa.
- Asimov, I. (2019). *Asimov: cuentos completos 1* [Asimov: cuentos completos 1]. Debolsillo.
- Bartra, A. (2022). *La epopeya de Gilgamesh*. La Guillotina.
- Benyamina, D., Hafid, A. & Gendreau, M. (2011). Wireless mesh networks design—A survey [Diseño de redes inalámbricas en malla: una encuesta]. *IEEE Communications Surveys & Tutorials*, 14(2), 299-310.
- Borges, L. (1964). *El otro, el mismo*. Ediciones Neperús.
- Chafii, M., Bader, F. & Palicot, J. (2018). Enhancing coverage in narrow band-IoT using *Machine Learning* [Mejora de la cobertura en IoT de banda estrecha mediante aprendizaje automático]. *IEEE Wireless Communications and Networking Conference (WCNC)*, 1-6.
- Cixin, L. (2016). *El problema de los tres cuerpos*. Nova.

Cixin, L. (2017). *El bosque oscuro*. Nova.

Cixin, L. (2018). *El fin de la muerte*. Nova.

Côté, D. (2018). Using *Machine Learning* in communication networks [Uso del Machine Learning en redes de comunicación]. *Journal of Optical Communications and Networking*, 10(10), D100-D109.

Dick, P. (2019). *¿Sueñan los androides con ovejas eléctricas?* MiNotauro.

Dong, G. & Liu, H. (2020). *Feature engineering for Machine Learning and data analytics* [Ingeniería de funciones para Machine Learning y análisis de datos]. CRC Press.

Doxiadis, A., & Papadimitriou, C. (2015). *An epic search for truth* [Una búsqueda épica de la verdad]. Bloomsbury.

El Hammouti, H., Ghogho, M., & Zaidi, S. (2018). A Machine Learning approach to predicting coverage in random wireless networks [Un enfoque de aprendizaje automático para predecir la cobertura en redes inalámbricas aleatorias]. *IEEE Globecom Workshops (GC Wkshps)*, 1-6.

Fernandes, D., Raimundo, A., Cercas, F., Sebastião, P., Dinis, R. y Ferreira, L. (2020). Comparison of Artificial Intelligence and Semi-Empirical Methodologies for Estimation of Coverage in Mobile Networks [Comparación de Inteligencia Artificial y Metodologías Semiempíricas para la Estimación de Cobertura en Redes Móviles]. *IEEE Access*, 8, 139803-139812.

Funabiki, N. (2011). *Wireless mesh networks* [Redes de malla inalámbricas]. BoD–Books on Demand.

Gibson, W. (1984). *Neuromancer*. Berkley.

Harari, Y. (2014). *Sapiens: de animales a dioses. Breve historia de la humanidad* [Sapiens: de animales a dioses. Breve historia de la humanidad]. Debate.

Hata, M. (1980). Empirical formula for propagation loss in land mobile radio services [Fórmula empírica para las pérdidas de propagación en los servicios de radiocomunicaciones móviles terrestres]. *IEEE Transactions on Vehicular Technology*, 29(3), 317-325.

Hesíodo. (2017). *Trabajos y días*. Alianza Editorial.

Hufford, G. (1952). An integral equation approach to the problem of wave propagation over an irregular surface [Un enfoque de ecuación integral al problema de la propagación de ondas sobre una superficie irregular]. *Quarterly of Applied Mathematics*, 9(4), 391-404.

Huyen, C. (2022). *Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications* [Diseño de sistemas de aprendizaje automático: un proceso iterativo para aplicaciones listas para producción]. O'Reilly Media.

Karunaratne, S. & Gacanin, H. (2019). An overview of Machine Learning approaches in wireless mesh networks [Una descripción general de los enfoques de aprendizaje automático en redes de malla inalámbricas]. *IEEE Communications Magazine*, 57(4), 102-108.

Knaflic, C. (2015). *Storytelling with data* [Narrar historias con datos]. Wiley.

Krishen, K. (1994). Future trends in antennas and propagation for the U.S. space program [Tendencias futuras en antenas y propagación para el programa espacial de EE. UU.]. *IEEE Antennas and Propagation Magazine*, 36, 31-35.

Luebbers, R. (1984). Propagation prediction for hilly terrain using GTD wedge diffraction [Predicción de propagación para terreno montañoso mediante difracción de cuña GTD]. *IEEE Transactions on Antennas and Propagation*, 32(9), 951-955.

Madrid, C. (2013). *Las bases de la matemática: Hilbert, en el principio fue el axioma*. RBA Coleccionables.

Meyrink, G. (2019). *El Golem*. Mirlo.

Mohtashami, V. & Shishegar, A. (2012). Modified wavefront decomposition method for fast and accurate ray-tracing simulation [Método de descomposición del frente de onda modificado para una simulación de trazado de rayos rápida y precisa]. *IET Microwaves, Antennas & Propagation*, 6(3), 295-304.

Ottaviani, J. (2016). *The Imitation Game: Alan Turing Decoded* [El juego de la imitación: Alan Turing decodificado]. Abrams ComicArts.

Pickover, C. (2021). *Inteligencia Artificial: una historia ilustrada* [Inteligencia Artificial: una historia ilustrada]. Librero.

Piñeiro, G. (2012). *Los teoremas de incompletitud: Gödel, la intuición tiene su lógica* [Los teoremas de incompletitud: Gödel, la intuición tiene su lógica]. RBA Coleccionables.

Platón. (2020). *Crátilo*. Greenbooks editore.

Popoola, S. & Oseni, O. (2014). Empirical path loss models for GSM network deployment in Makurdi, Nigeria [Modelos empíricos de pérdida de ruta para el despliegue de redes GSM en Makurdi, Nigeria]. *International Refereed Journal of Engineering and Science*, 3(6), 85-94.

Popoola, S., Misra, S. & Atayero, A. (2018). Outdoor path loss predictions based on extreme learning machine [Predicciones de pérdida de ruta al aire libre basadas en una máquina de aprendizaje extremo]. *Wireless Personal Communications*, 99, 441-460.

Recinos. A. (1992). *Popol Vuh: las antiguas historias del Quiché*. Piedra Santa.

Roy, D., Mukherjee, T. & Chatterjee, M. (2019). Machine Learning in adversarial RF environments [Aprendizaje automático en entornos de RF adversaries]. *IEEE Communications Magazine*, 57(5), 82-87.

Santa Biblia. (1960). *Génesis. Reina Valera*. Sociedades Bíblicas Unidas.

Shelley, M. (2015). *Frankenstein*. Nórdica Libros.

Simu, C., & Vesa, A. (2020). First fresnel ellipse evaluation [Primera evaluación de la elipse de Fresnel]. *International Symposium on Electronics and Telecommunications (ISETC)*, 1-5.
DOI:[10.1109/ISETC50328.2020.9301068](https://doi.org/10.1109/ISETC50328.2020.9301068)

Turing, A. (1950). Computing machinery and intelligence [Maquinaria informática e inteligencia]. *MIND, LIX*(236), 433-460.

Vallejo, I. (2019). *El infinito en un junco: la invención de los libros en el mundo antiguo*. Siruela.

Zeng, G., Wang, B., Ding, Y., Xiao, L. & Mutka, M. (2007). Multicast algorithms for multi-channel wireless mesh networks [Algoritmos de multidifusión para redes de malla inalámbricas multicanal]. *IEEE International Conference on Network Protocols*, 1-10.

APÉNDICES

Apéndice 1.

Cálculo de característica de Fresnel

A continuación, se muestra el desarrollo matemático utilizado para la generación de la característica de Fresnel, su aplicación se encuentra en el cuaderno Característica Fresnel en el repositorio de este trabajo y en el script de Python fresnelFeature.py.

En primera instancia se sitúan las dos fuentes emisoras y receptoras de la radiación en los focos de la elipse de Fresnel. Haciendo uso de la fórmula de Haversine se puede obtener la distancia sobre la superficie de la tierra entre estos dos puntos a partir de las coordenadas en radianes. Siendo $dlat = lat1 - lat2$ y $dlon = lon1 - lon2$, y siendo:

$$a = \sin\left(\frac{dlat}{2}\right)^2 + \cos(lat2) * \cos(lat1) * \sin\left(\frac{dlon}{2}\right)^2$$

Dado que se conoce el radio de la tierra R , la distancia sobre la superficie de la tierra entre los dos puntos del radioenlace, en metros es:

$$d_0 = 2R * \text{atan2}(\sqrt{a}, \sqrt{1-a})$$

Donde atan2 es una función que devuelve un ángulo en radianes en el rango $[-\pi, \pi]$.

Continuación del Apéndice 1.

Una primera aproximación para calcular la zona de Fresnel sería utilizar d_0 como la distancia entre los puntos, por lo tanto, según el análisis de Fresnel, y conociendo la frecuencia f y la velocidad de la luz c_l , es posible determinar los parámetros que permiten obtener una primera elipse.

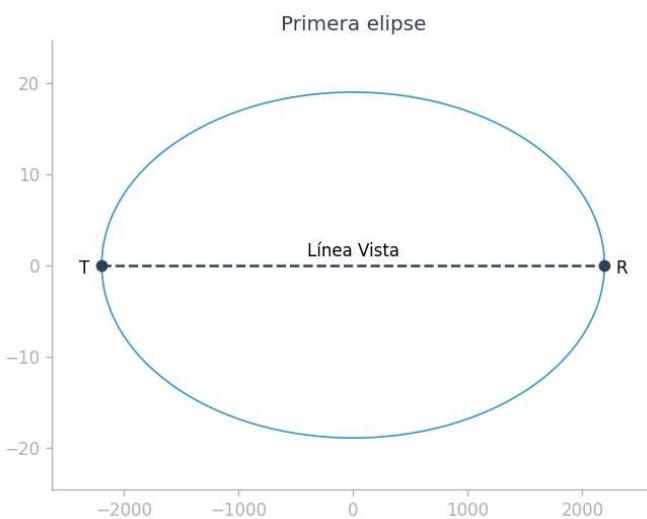
$$c = \frac{d_0}{2}$$

$$b = \frac{1}{2} \sqrt{\frac{d_0 c_l}{f}}$$

$$a = \sqrt{c^2 + b^2}$$

Siendo esta la elipse obtenida, haciendo uso de un conjunto de coordenadas arbitrarias.

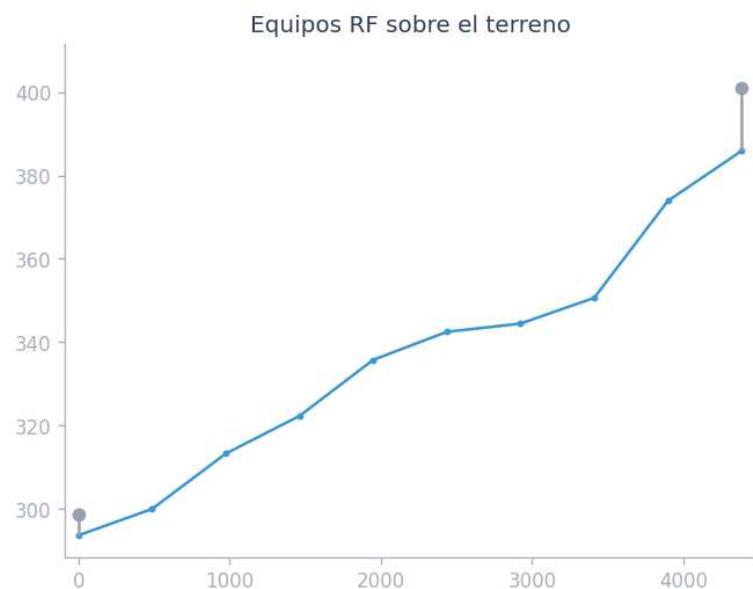
Primer intento de Elipse de primera zona de Fresnel



Continuación del Apéndice 1.

Evidentemente, esta aproximación no contempla la diferencia de alturas debido a la geografía, ni debido a la instalación particular de cada equipo, por lo cual es necesario modificar ligeramente la distancia entre los dos equipos. Para esto se utiliza la información de altimetría obtenida con la API de elevación de Google. Al incluir la geografía para el mismo conjunto arbitrario de coordenadas, se hace evidente que la distancia sobre la superficie de la esfera terrestre no corresponde a cabalidad con la realidad.

Distribución de variables numéricas no escaladas 2



Continuación del Apéndice 1.

Siendo ***path*** el vector que contiene una cantidad n de puntos equidistantes en la línea entre los dos equipos, ***evec*** el vector que contiene n valores de altura sobre el nivel del mar para los puntos en ***path***, obtenidos con la API de elevación de Google y ***distVec*** el vector que contiene las distancias sobre la superficie de la tierra a partir de las coordenadas del equipo 1, obtenidas con la fórmula de Haversine. Es posible obtener la elipse de la primera zona de Fresnel que contemple la geografía y la altura de cada equipo ($h_{equipo1}, h_{equipo2}$), utilizando las mismas ecuaciones que antes, pero reubicando los focos según la elevación y altura de los equipos, y recalculando la distancia del radioenlace D_e de la siguiente manera:

$$h_1 = h_{equipo1} + \mathbf{evec}_1$$

$$h_2 = h_{equipo2} + \mathbf{evec}_n$$

$$h_{delta} = h_1 - h_2$$

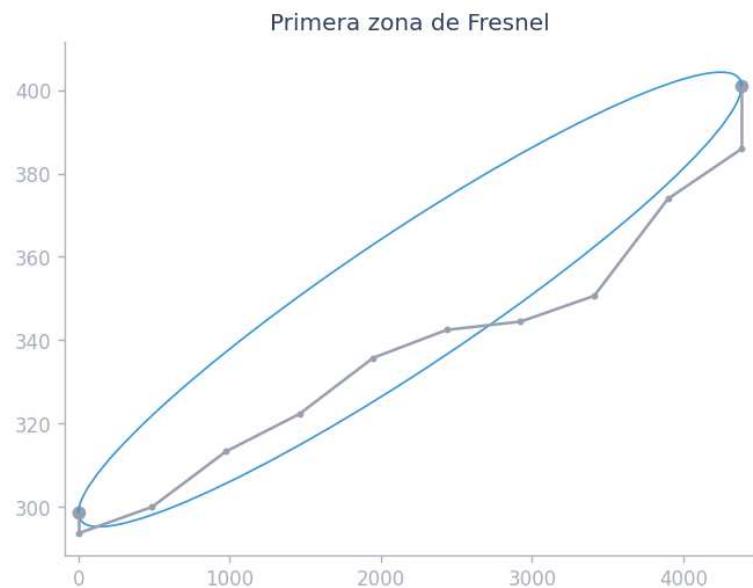
$$D_e = \sqrt{h_{delta}^2 + d_0^2}$$

$$\beta = \tan^{-1}\left(\frac{-h_{delta}}{d_0}\right)$$

Con el nuevo cálculo de la distancia del radioenlace se pueden utilizar las relaciones previamente indicadas del análisis de Fresnel y las relaciones de una elipse de modo que la primera zona de Fresnel es:

Continuación del Apéndice 1.

Distribución de variables numéricas no escaladas 2



Nota. El gráfico muestra la primera elipse de Fresnel para un radioenlace puesta sobre la geografía. Elaboración propia, realizado con Python.

La característica deseada corresponde a la obstrucción de la primera elipse de Fresnel debido a la geografía entre los dos equipos. Por lo que el objetivo es calcular la porción de la componente vertical de las $n - 2$ cuerdas perpendiculares al eje mayor correspondientes a los $n - 2$ puntos entre los focos.

En primera instancia se parametriza el eje mayor para poder obtener las distintas distancias sobre la línea vista en función de las distancias sobre la superficie de la tierra:

Continuación del Apéndice 1.

$$b_0 = h1$$

$$y_{los} = \frac{-h_{delta}}{d_0} x + b_0$$

$$d_i = \sqrt{x_i^2 + (b_0 - y_{los}(x_i))^2}$$

Con lo anterior es posible obtener la longitud de la mitad de la cuerda perpendicular al eje mayor.

$$h_{normal}(d_i) = \sqrt{\frac{d_i(D_e - d_i)c_l}{f * D_e}}$$

Siendo esta componente normal al plano inclinado de la línea vista, es necesario obtener su componente vertical.

$$h_{y_i} = h_{normal}(d_i) * \cos(\beta)$$

Luego, dado que el centro de las cuerdas coincide con la línea vista, es posible determinar el punto inferior y superior de las cuerdas perpendiculares al eje horizontal.

$$downPoint_i = y_{los}(x_i) - h_{y_i}$$

$$upPoint_i = y_{los}(x_i) + h_{y_i}$$

Continuación del Apéndice 1.

Primera zona de Fresnel con correspondientes cuerdas



Nota. El gráfico muestra la primera elipse de Fresnel con sus correspondientes cuerdas a distancias horizontales equidistantes. Elaboración propia, realizado con Python.

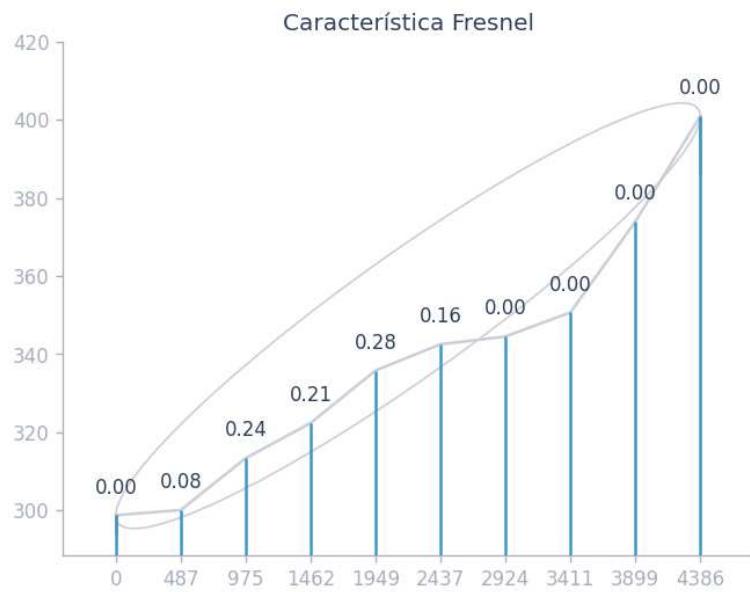
Por último, se calcula la obstrucción de la componente vertical de las cuerdas calculadas, lo que constituye la característica de Fresnel.

$$fresnel_obstruction_i = \begin{cases} 0, & \text{si } evec_i < dp_i \\ \frac{evec_i - dp_i}{2 * h_{y_i}}, & \text{si } dp_i \leq evec_i \leq up_i \\ 1, & \text{si } evec_i > up_i \end{cases}$$

Donde $dp_i = downPoint_i$ y $up_i = upPoint_i$.

Continuación del Apéndice 1.

Característica Fresnel

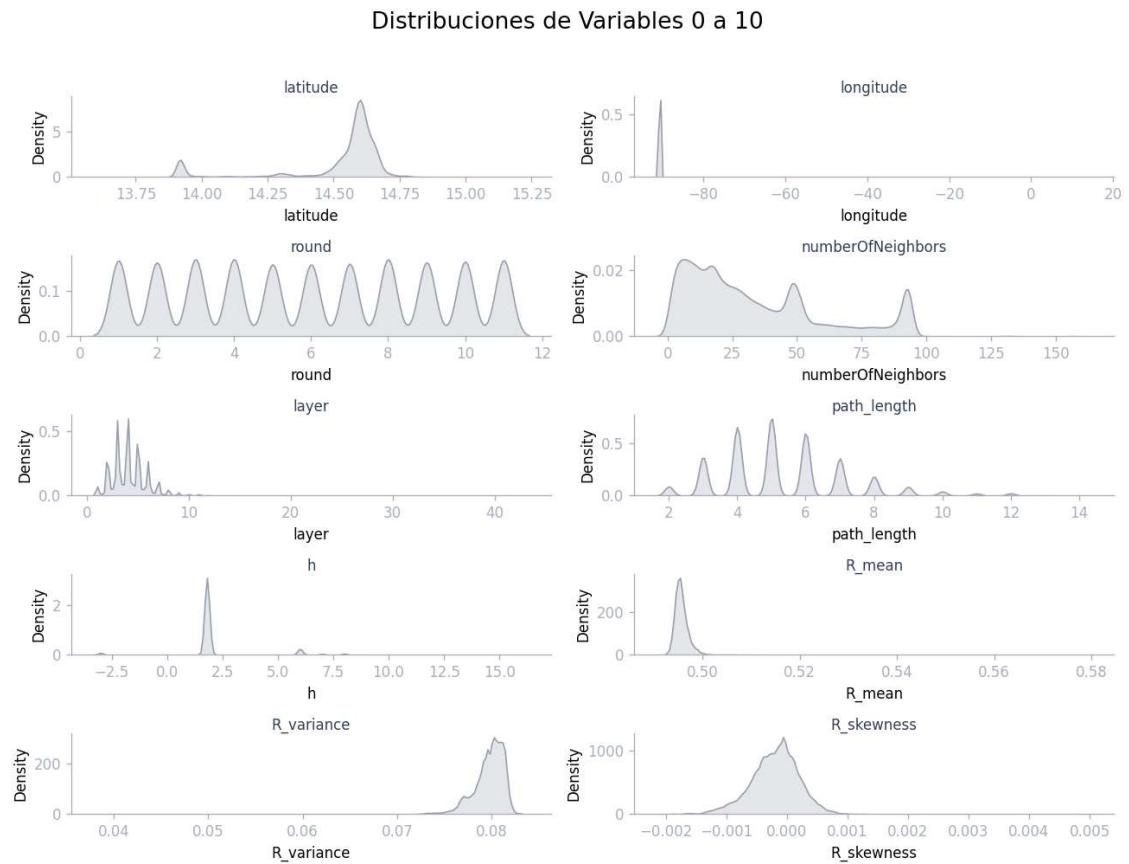


Nota. La figura muestra el resultado final del desarrollo teórico para la generación de la característica de Fresnel. Elaboración propia, realizado con Python.

Nota. Cálculo de característica. Elaboración propia, realizado con Excel y Word.

Apéndice 2.

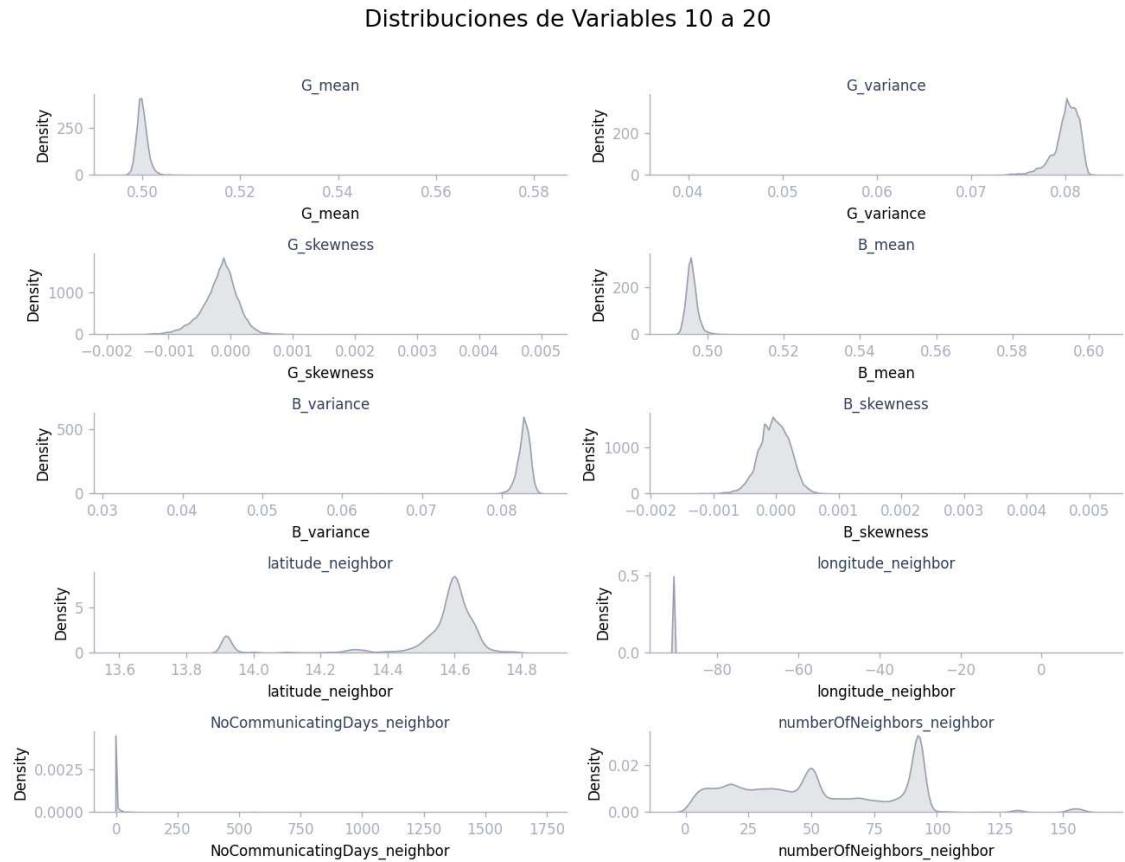
Distribución de variables numéricas no escaladas 1



Nota. El gráfico muestra la distribución de las variables numéricas no escaladas. Elaboración propia, realizado con Python.

Apéndice 3.

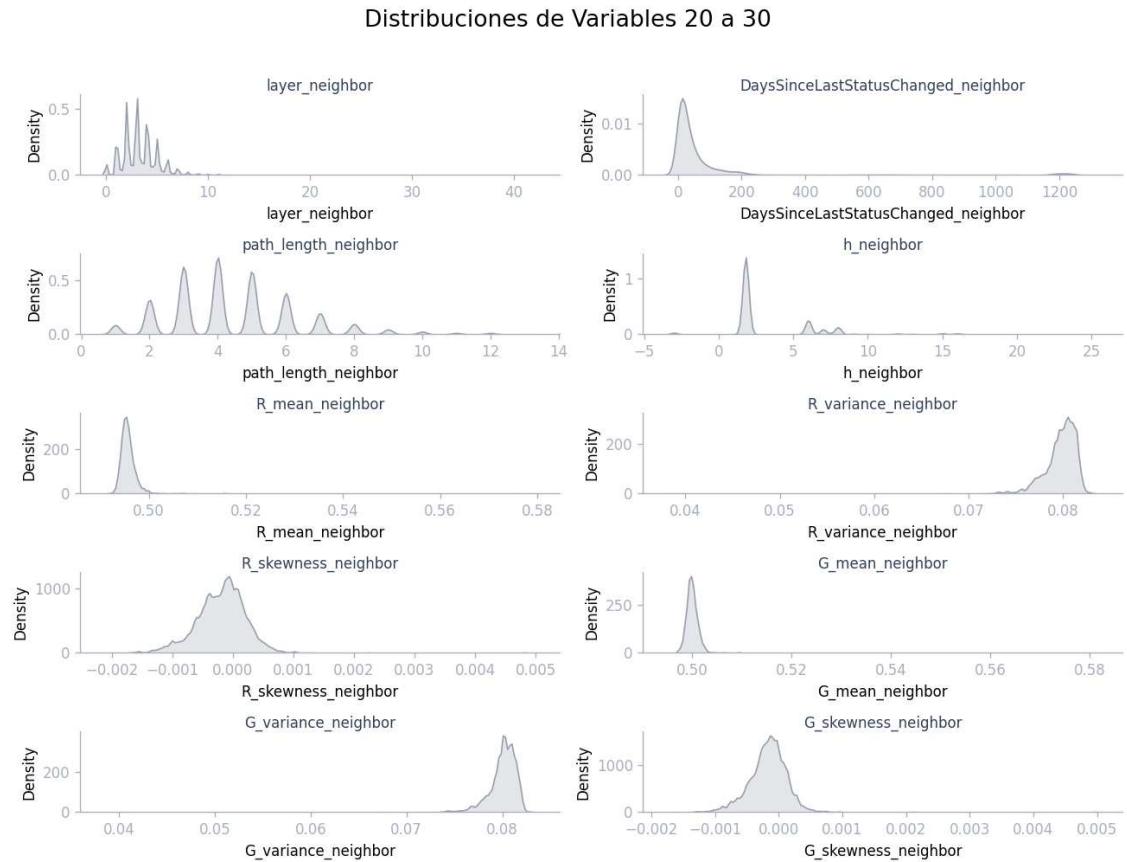
Distribución de variables numéricas no escaladas 2



Nota. El gráfico muestra la distribución de las variables numéricas no escaladas. Elaboración propia, realizado con Python.

Apéndice 4.

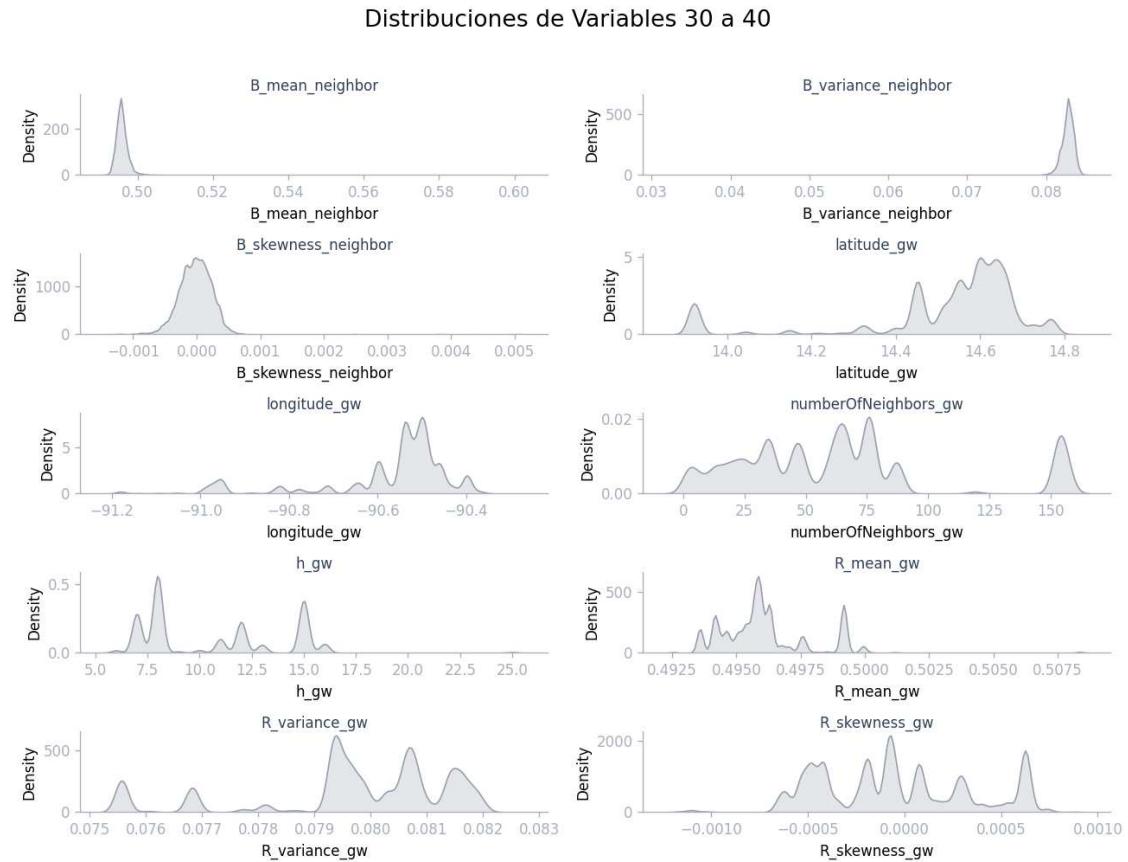
Distribución de variables numéricas no escaladas 3



Nota. El gráfico muestra la distribución de las variables numéricas no escaladas. Elaboración propia, realizado con Python.

Apéndice 5.

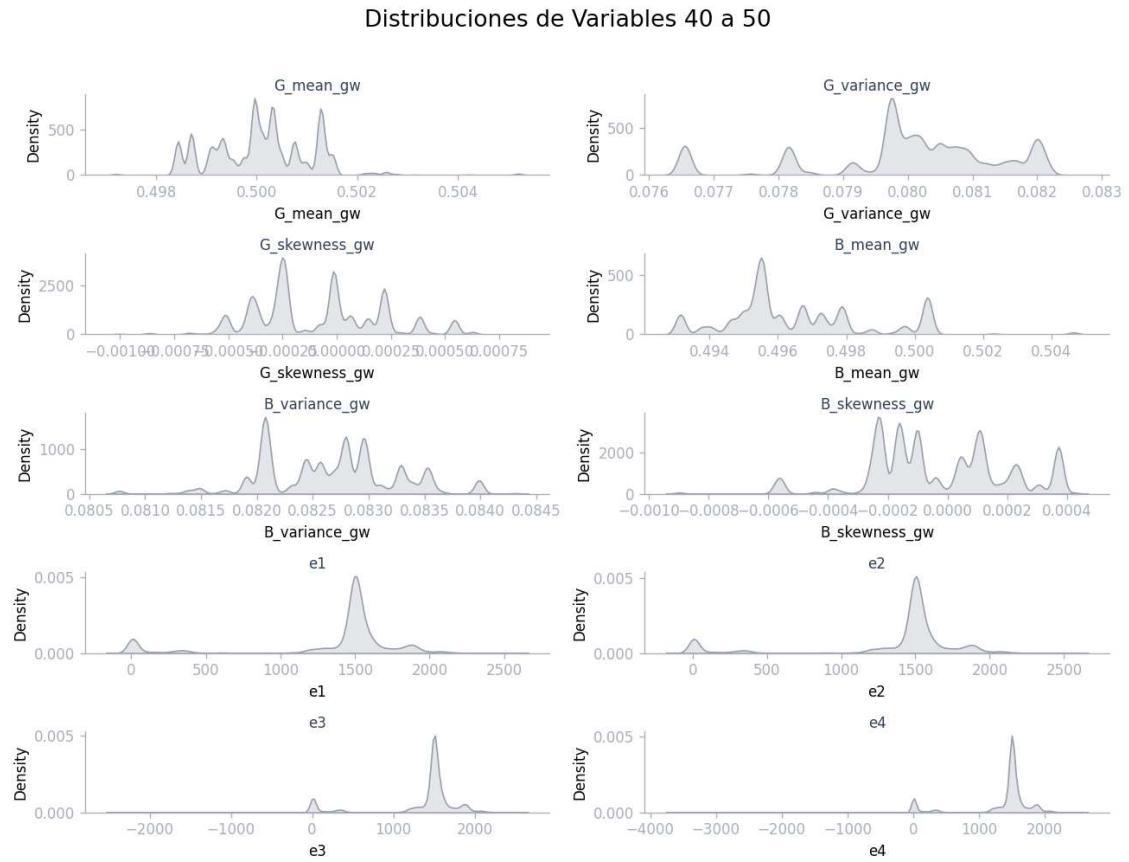
Distribución de variables numéricas no escaladas 4



Nota. El gráfico muestra la distribución de las variables numéricas no escaladas. Elaboración propia, realizado con Python.

Apéndice 6.

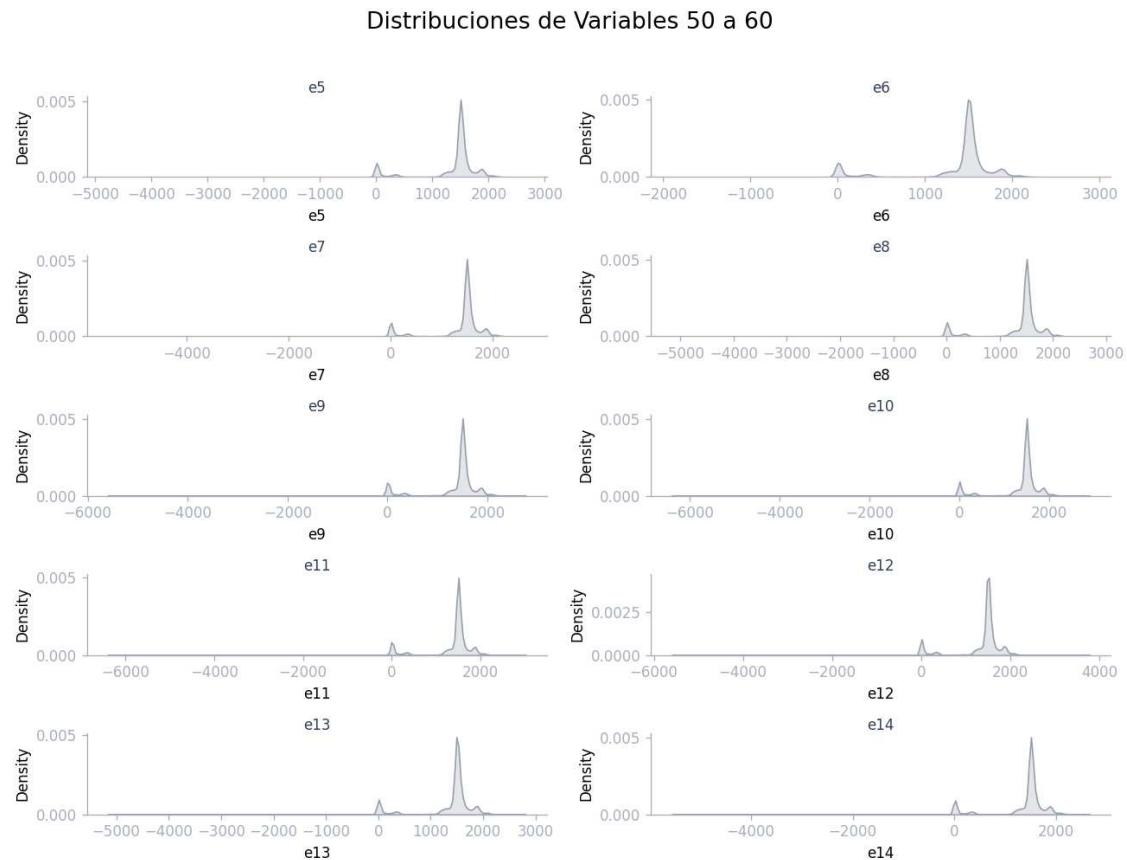
Distribución de variables numéricas no escaladas 5



Nota. El gráfico muestra la distribución de las variables numéricas no escaladas. Elaboración propia, realizado con Python.

Apéndice 7.

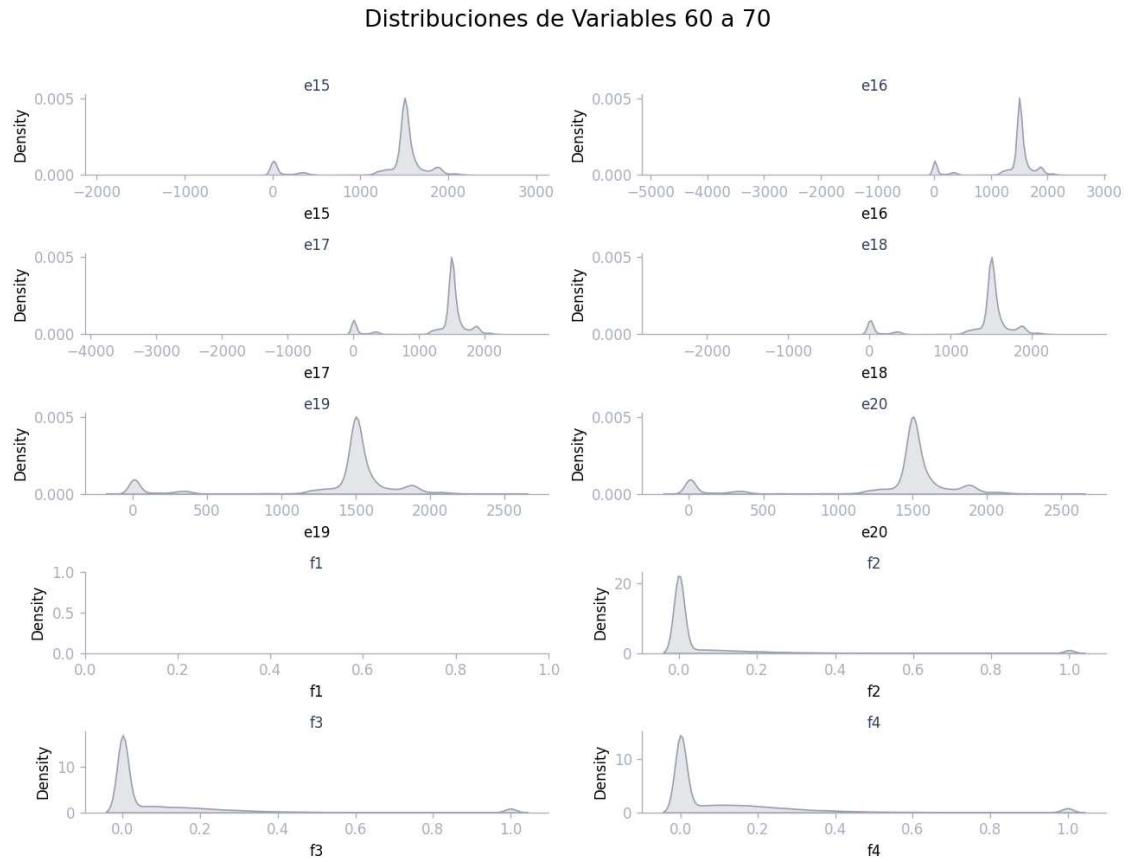
Distribución de variables numéricas no escaladas 6



Nota. El gráfico muestra la distribución de las variables numéricas no escaladas. Elaboración propia, realizado con Python.

Apéndice 8.

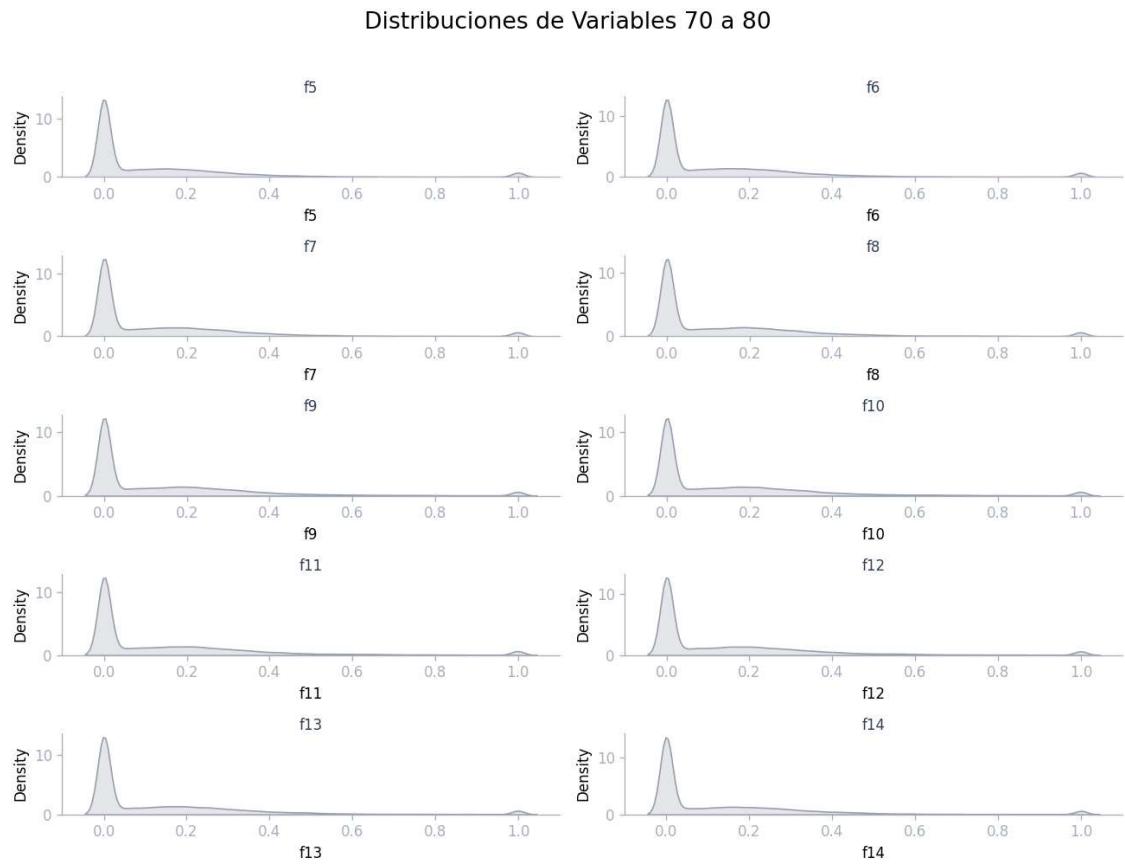
Distribución de variables numéricas no escaladas 7



Nota. El gráfico muestra la distribución de las variables numéricas no escaladas. Elaboración propia, realizado con Python.

Apéndice 9.

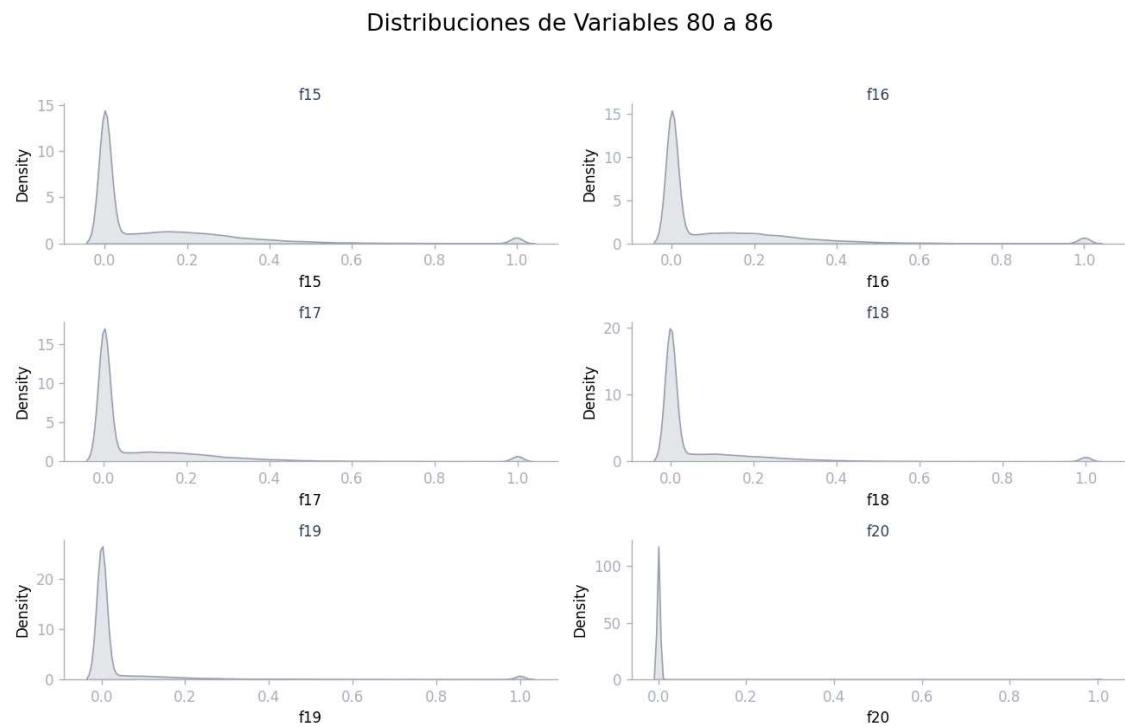
Distribución de variables numéricas no escaladas 8



Nota. El gráfico muestra la distribución de las variables numéricas no escaladas. Elaboración propia, realizado con Python.

Apéndice 10.

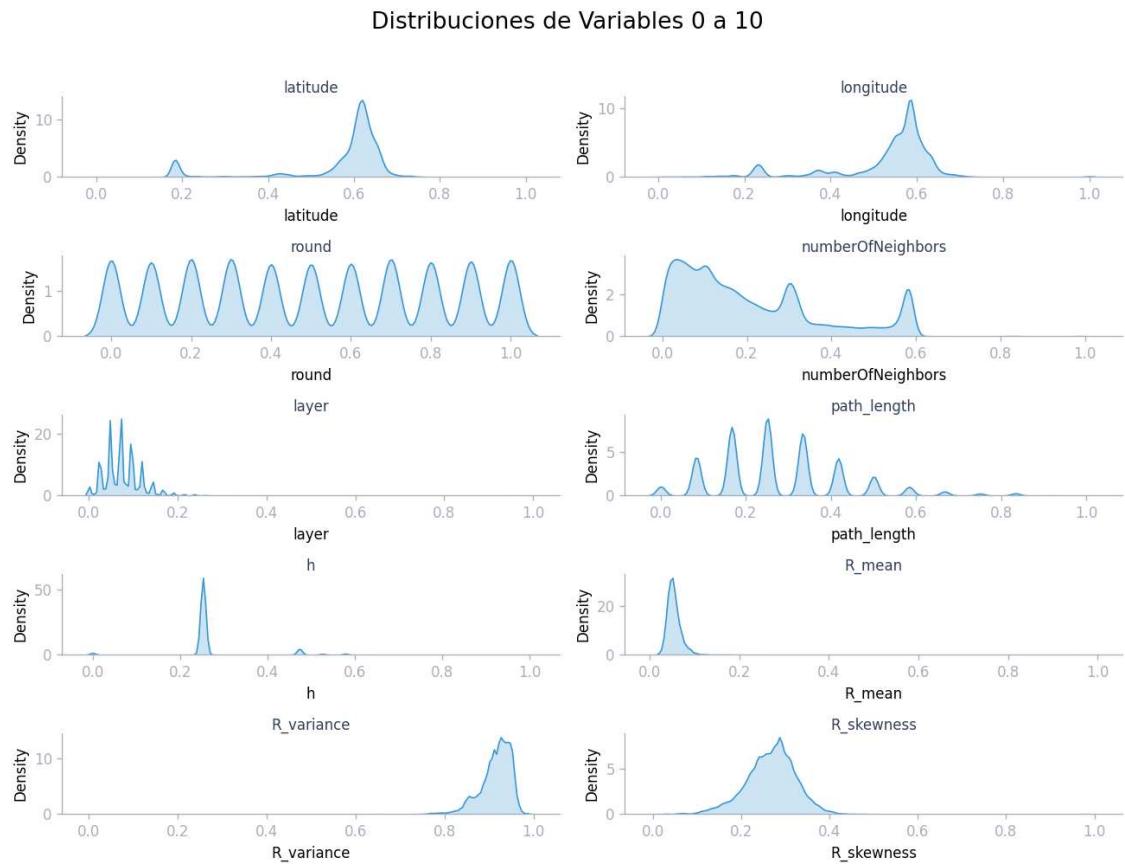
Distribución de variables numéricas no escaladas 9



Nota. El gráfico muestra la distribución de las variables numéricas no escaladas. Elaboración propia, realizado con Python.

Apéndice 11.

Distribución de variables numéricas escaladas 1

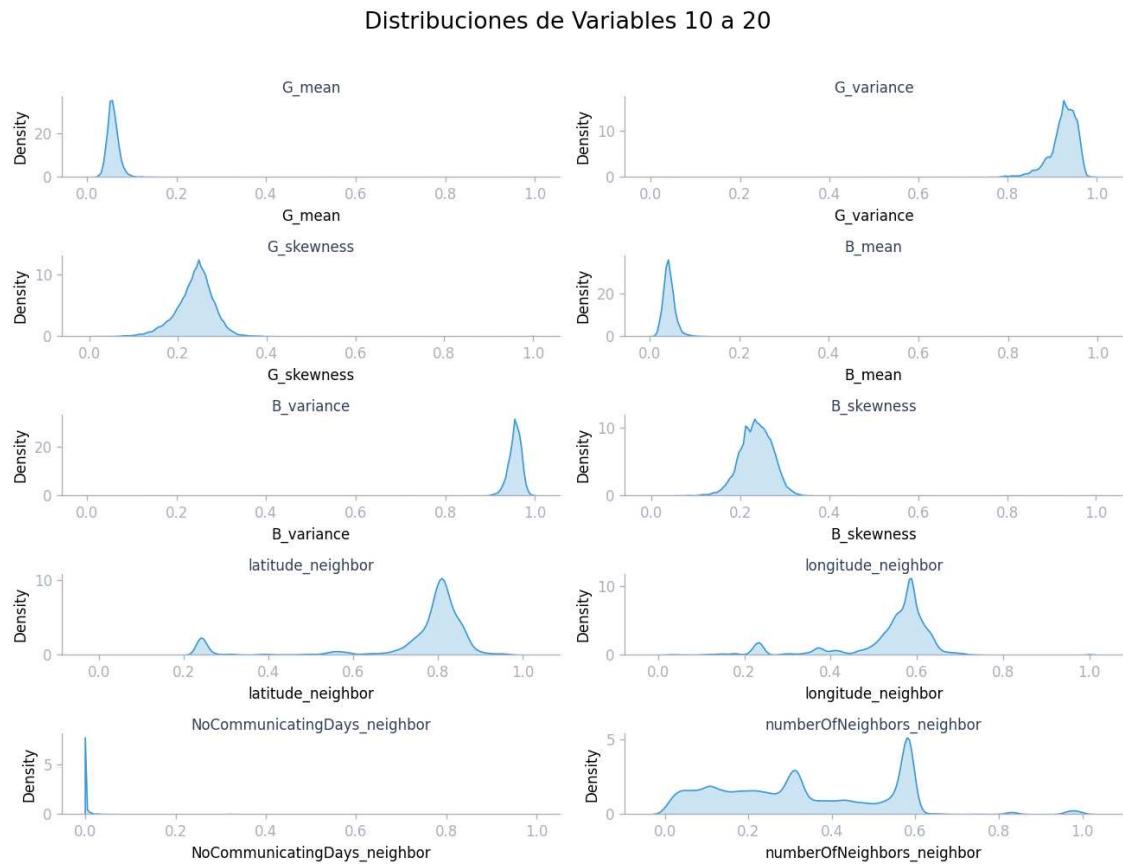


Nota. El gráfico muestra la distribución de las variables numéricas escaladas entre 0 y 1.

Elaboración propia, realizado con Python.

Apéndice 12.

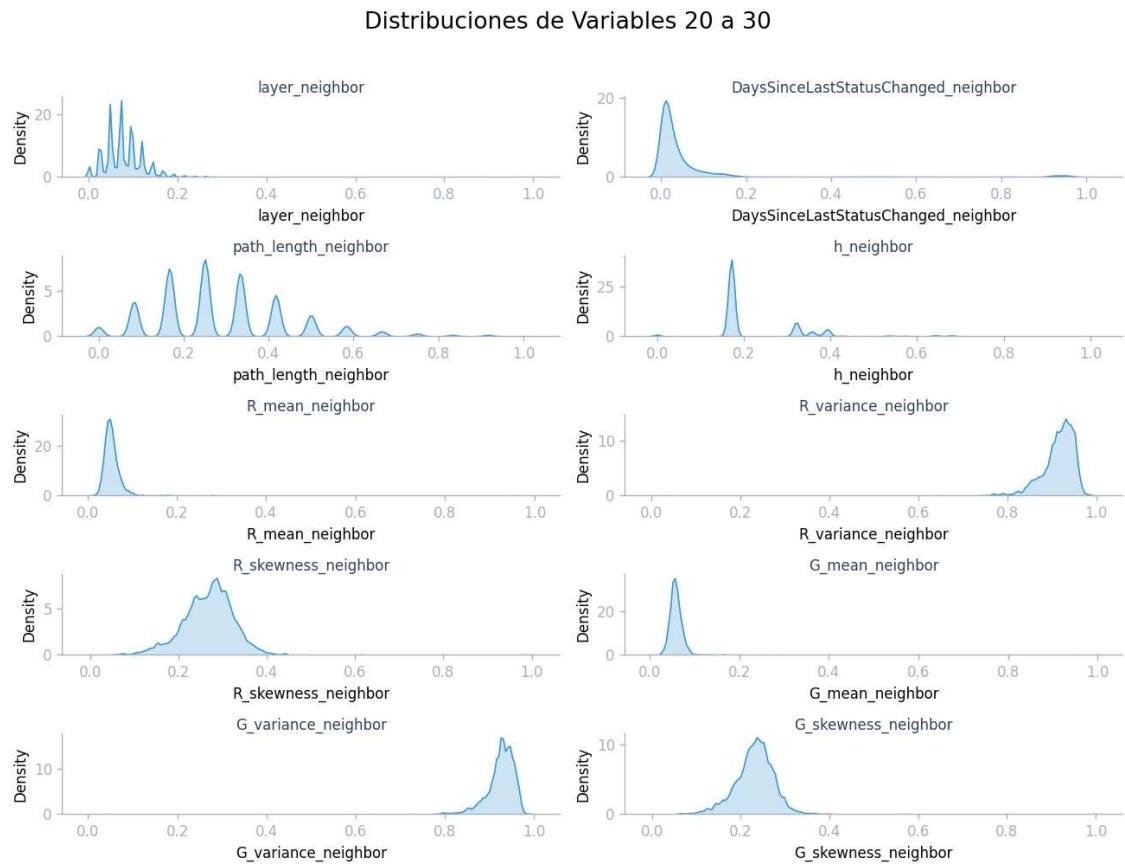
Distribución de variables numéricas escaladas 2



Nota. El gráfico muestra la distribución de las variables numéricas escaladas entre 0 y 1.
Elaboración propia, realizado con Python.

Apéndice 13.

Distribución de variables numéricas escaladas 3

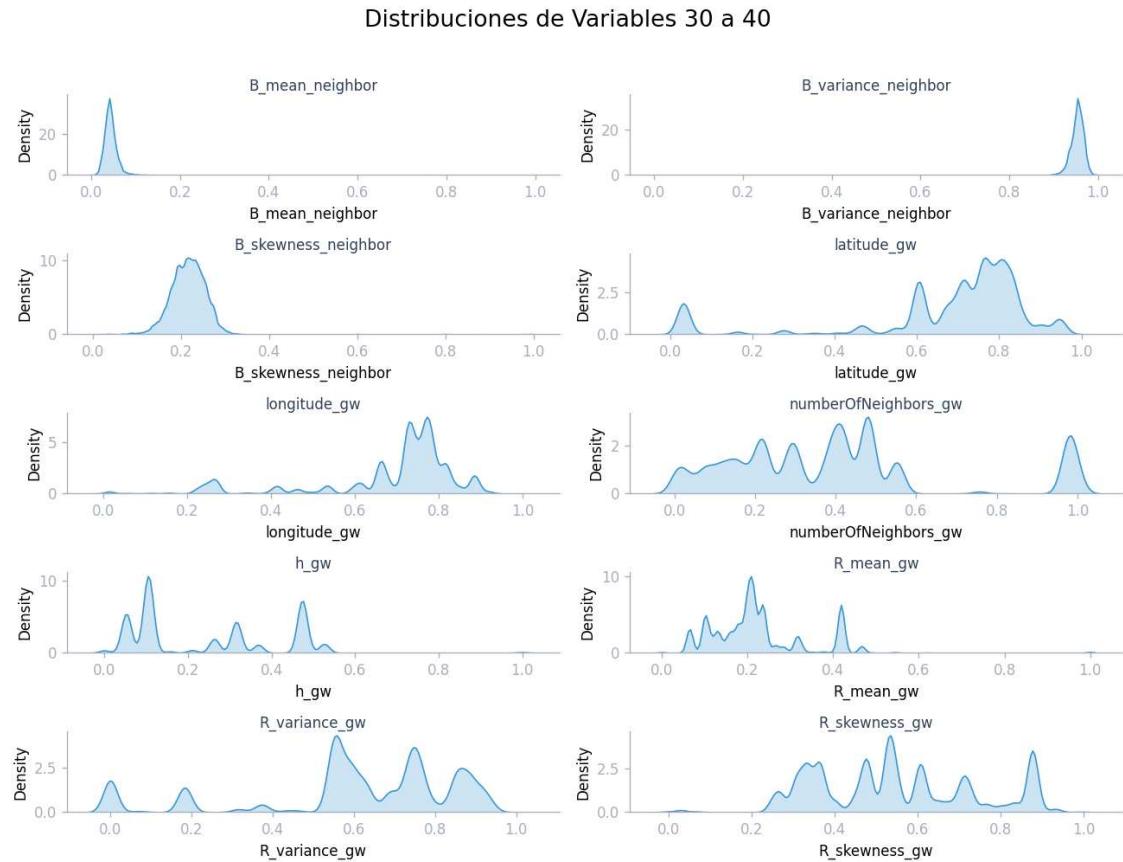


Nota. El gráfico muestra la distribución de las variables numéricas escaladas entre 0 y 1.

Elaboración propia, realizado con Python.

Apéndice 14.

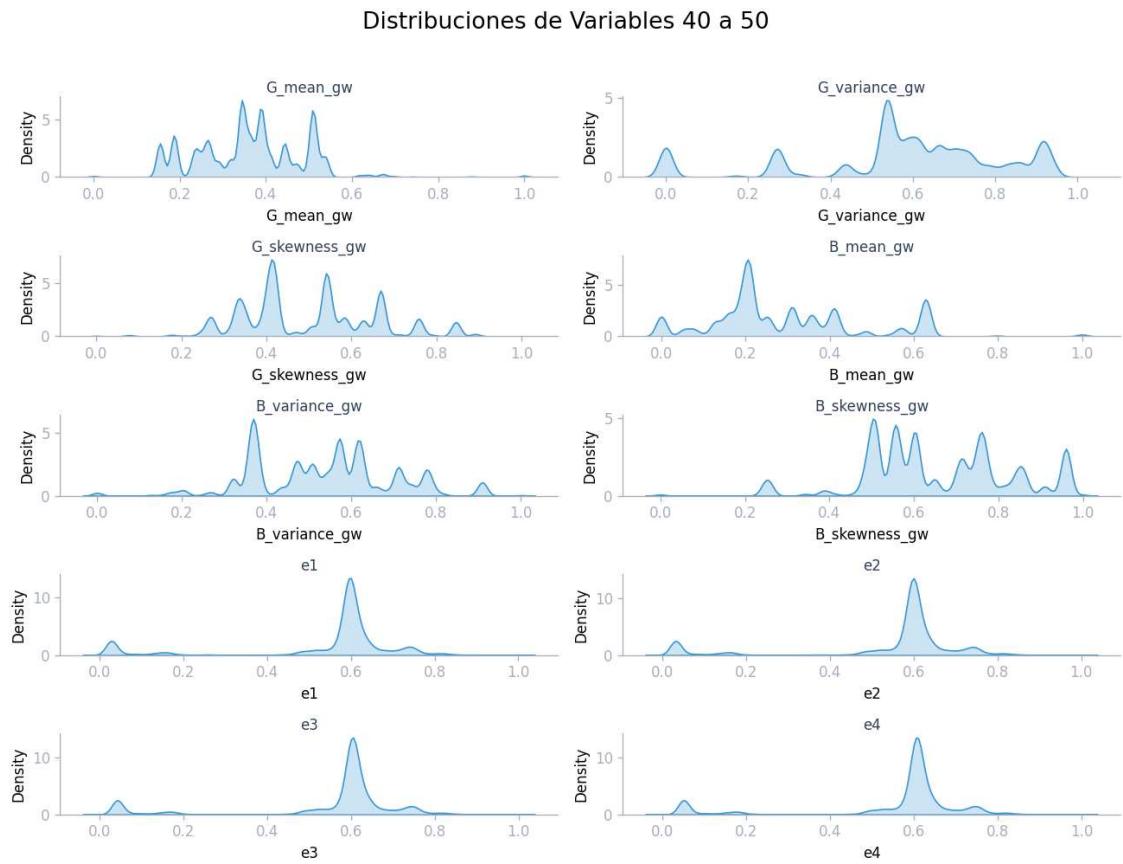
Distribución de variables numéricas escaladas 4



Nota. El gráfico muestra la distribución de las variables numéricas escaladas entre 0 y 1.
Elaboración propia, realizado con Python.

Apéndice 15.

Distribución de variables numéricas escaladas 5

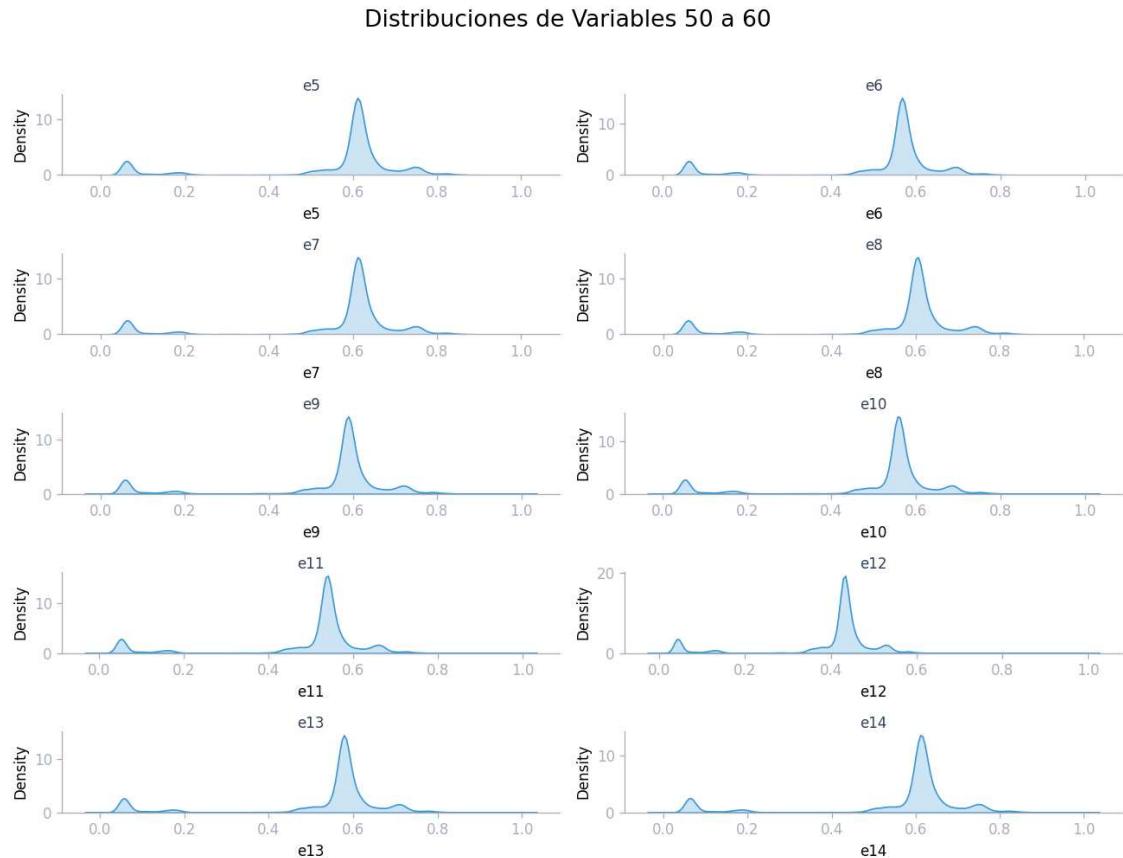


Nota. El gráfico muestra la distribución de las variables numéricicas escaladas entre 0 y 1.

Elaboración propia, realizado con Python.

Apéndice 16.

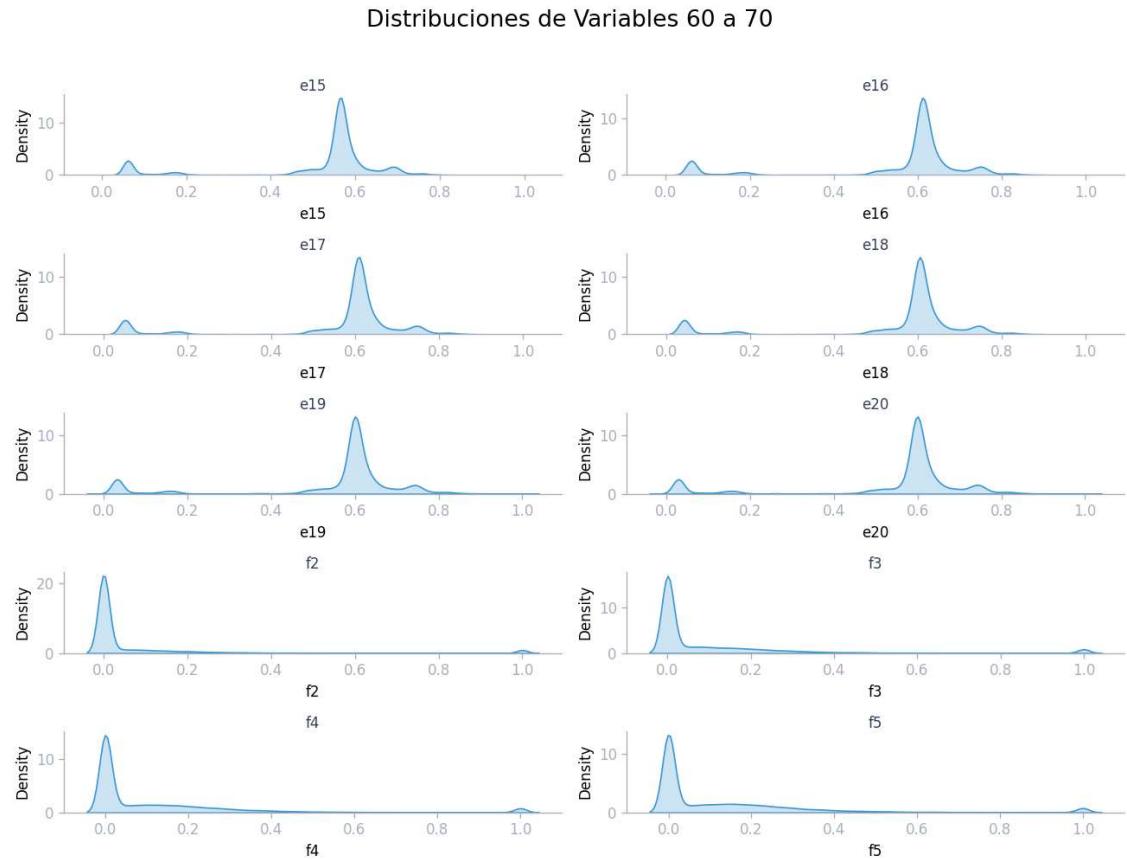
Distribución de variables numéricas escaladas 6



Nota. El gráfico muestra la distribución de las variables numéricas escaladas entre 0 y 1.
Elaboración propia, realizado con Python.

Apéndice 17.

Distribución de variables numéricas escaladas 7

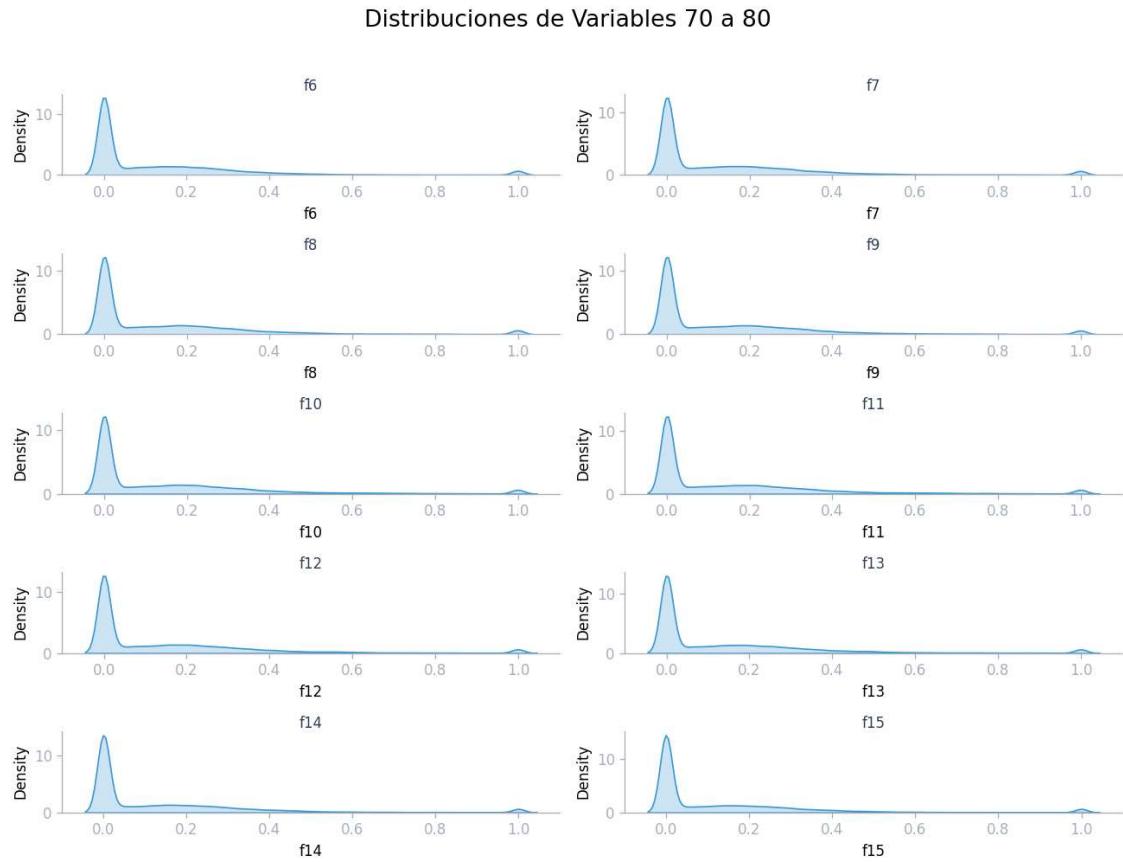


Nota. El gráfico muestra la distribución de las variables numéricas escaladas entre 0 y 1.

Elaboración propia, realizado con Python.

Apéndice 18.

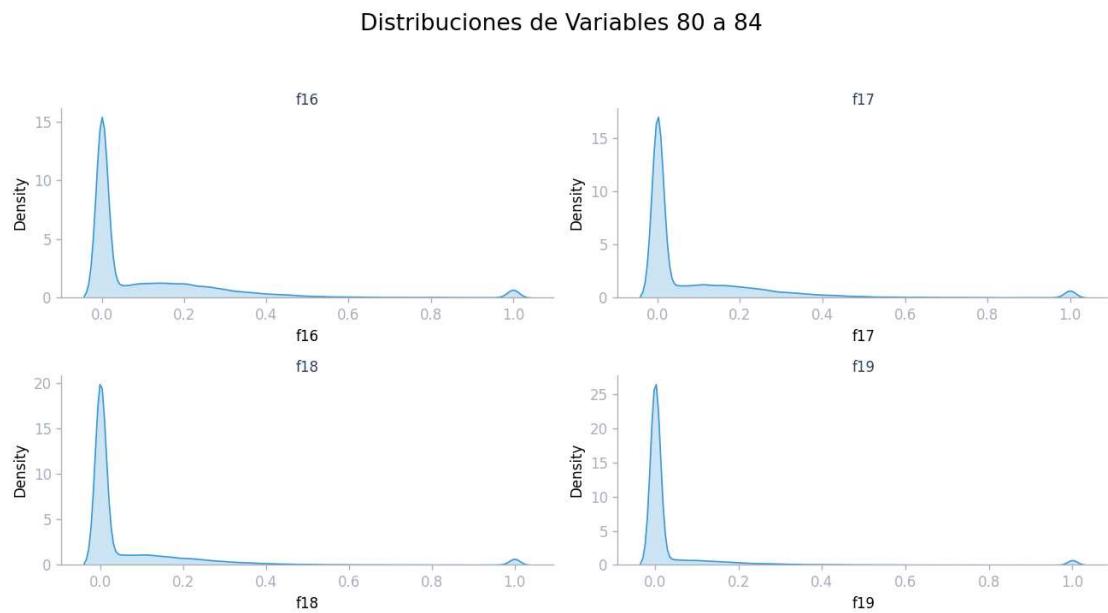
Distribución de variables numéricas escaladas 8



Nota. El gráfico muestra la distribución de las variables numéricas escaladas entre 0 y 1.
Elaboración propia, realizado con Python.

Apéndice 19.

Distribución de variables numéricas escaladas 9



Nota. El gráfico muestra la distribución de las variables numéricas escaladas entre 0 y 1.

Elaboración propia, realizado con Python.

Apéndice 20.

Resultados tabulados Machine Learning

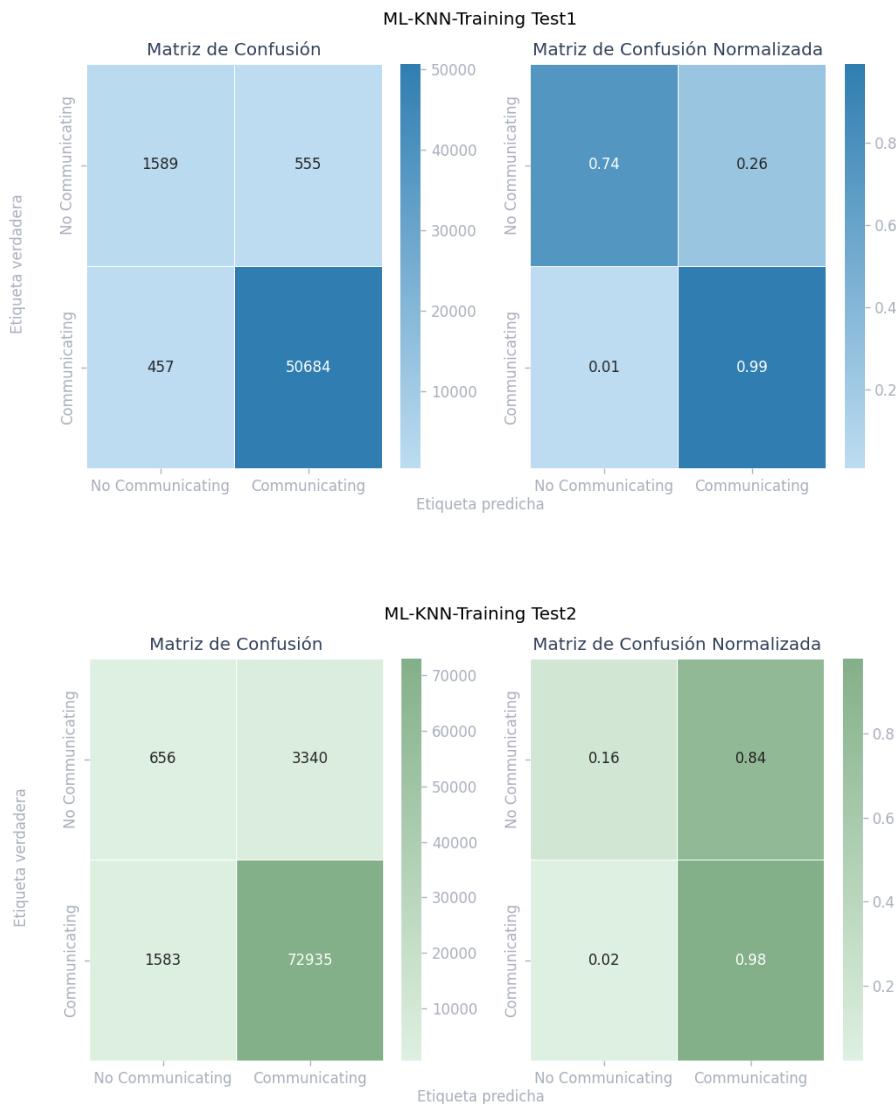
Test	Experiment	Algoritmo	Datos Entrenamiento	Accuracy	F1 Score Macro	MPCA	F1 Score Weighted	Recall	Precision	F1 Score Micro	True Positives	True Negatives	False Negatives	False Positives
Test2	ML	GBC	Training, OS	0.947756349	0.5701189	0.9424820799	0.91903779	0.9475653492	0.9475653492	0.9475653492	73738	599	720	3397
Test1	ML	GBC	Training, OS	0.993738812	0.967738311	0.985368814	0.98429494	0.998998827	0.998998827	0.998998827	50861	2093	280	51
Test2	ML	GBC	Training, US	0.94688238	0.57548015	0.55246122	0.952126208	0.96155093	0.95424372	0.94688238	73889	453	629	3843
Test1	ML	GBC	Training, US	0.99104048	0.946967911	0.986653346	0.991426512	0.901125046	0.98940496	0.99104048	50694	2114	447	30
Test2	ML	KNN	Training	0.937397181	0.588818808	0.57148049	0.9288528478	0.97815881	0.95621108	0.937297807	72935	656	1583	3340
Test1	ML	KNN	Training	0.98100779	0.874293905	0.86610099	0.980794774	0.99106392	0.989116841	0.981007788	50684	1589	487	555
Test2	ML	KNN	Training, US	0.82340729	0.54136162	0.61610223	0.8644421248	0.846508447	0.86257638	0.82340729	63110	1539	11408	2457
Test1	ML	KNN	Training, US	0.8578924	0.50505333	0.52348656	0.85640677	0.95486107	0.95486107	0.95486107	7359	1918	43752	226
Test2	ML	RF	Training	0.95076037	0.532310156	0.52348656	0.929664885	0.98918141	0.95138126	0.950760374	74457	191	61	38056
Test1	ML	RF	Training, OS	0.99566279	0.97105116	0.960652226	0.99566279	0.996699	0.996699	0.99566279	51074	1978	67	166
Test2	ML	RF	Training, OS	0.95228912	0.540955124	0.52845161	0.930308546	0.98080442	0.95186659	0.95208912	74376	235	142	3761
Test1	ML	RF	Training, OS	0.99746645	0.983602119	0.98371017	0.99746673	0.98817034	0.998658987	0.997466454	51073	2077	98	67
Test2	ML	RF	Training, US	0.94873526	0.560083614	0.62722032	0.98174656	0.98174656	0.98174656	0.948735257	73413	1076	1105	2920
Test1	ML	RF	Training, US	0.98705076	0.524823286	0.58165348	0.987791898	0.98724269	0.98987142	0.987050765	50503	2092	638	52
Test2	ML	SVM	Training, US	0.86739944	0.577412934	0.53161933	0.891563909	0.891411054	0.96334525	0.867399445	66528	1475	7890	2321
Test1	ML	SVM	Training, US	0.90222389	0.588337593	0.50951492	0.92574558	0.90181881	0.99617587	0.90222389	5033	1967	5033	177
Test2	ML	GBC	Training, OS	0.86171893	0.8030444802	0.70820187	0.880989752	0.87911646	0.97255007	0.861718929	66510	2147	9008	1849
Test1	ML	GBC	Training, OS	0.87574364	0.8401237415	0.8250442	0.907891401	0.9777302	0.98186848	0.875743643	44888	1776	6253	368
Test2	ML	GBC	Training, US	0.91479553	0.695787148	0.6246076	0.91965353	0.96196014	0.94179755	0.91479755	70418	1205	3900	2791
Test1	ML	GBC	Training, US	0.89529962	0.791931036	0.84871085	0.859537501	0.91512617	0.99762188	0.95129962	48662	2028	249	116
Test2	ML	GBC	Oversampled	0.90979264	0.601871412	0.590951492	0.90979264	0.93049258	0.9897397626	0.90979264	65652	1080	4956	2916
Test1	ML	KNN	Oversampled	0.94705827	0.761465822	0.89958054	0.90951492	0.99617587	0.99834313	0.947058272	48646	1818	2495	326
Test2	ML	KNN	Undersampled	0.74679675	0.5030450047	0.62812092	0.815985668	0.780245385	0.965657006	0.74679675	56652	1982	17886	2014
Test1	ML	KNN	Undersampled	0.768333734	0.583333734	0.583333734	0.768324908	0.76958275	0.7653337337	0.7653337337	38809	1972	12332	172
Test2	ML	RF	Training, OS	0.8625086	0.683827455	0.65994979	0.889723067	0.885624931	0.96688305	0.862508597	65983	1736	8535	2260
Test1	ML	RF	Training, OS	0.88409496	0.651823724	0.8614167	0.913193619	0.86067066	0.99523549	0.884094961	45315	1794	5826	350
Test2	ML	RF	Training, US	0.83921339	0.590977047	0.72103448	0.877167389	0.88239337	0.97483659	0.83921339	63534	2356	10934	1840
Test1	ML	RF	Training, US	0.85211598	0.612267558	0.8334479	0.882681131	0.88365349	0.99101083	0.85211598	43657	1748	7484	396
Test2	ML	GBC	Training, OS	0.91940291	0.803026065	0.60633363	0.921343816	0.96465525	0.96191083	0.919402909	71139	1047	3379	2949
Test1	ML	GBC	Training, OS	0.59599367	0.87172244	0.96498959	0.965792148	0.96034493	0.99798551	0.959913672	49113	2036	2028	108
Test2	ML	GBC	Training, US	0.91683012	0.8343617177	0.655995466	0.923046826	0.94896162	0.965683399	0.916830119	704393	1491	4025	2505
Test1	ML	GBC	Training, US	0.94448719	0.77262093	0.83912977	0.9974487192	0.9974487192	0.9974487192	48326	2001	2815	143	
Test2	ML	KNN	Oversampled	0.87511781	0.561641591	0.60522322	0.884959445	0.90520064	0.96146621	0.875117813	67491	1218	2077	2778
Test1	ML	KNN	Oversampled	0.91799625	0.713418725	0.9017928	0.937151679	0.97195333	0.99496247	0.741731	1691	4010	253	
Test2	ML	KNN	Undersampled	0.75029939	0.513797999	0.64097874	0.811937999	0.76288182	0.96730491	0.75029931	56834	17684	1921	1921
Test1	ML	KNN	Undersampled	0.770334813	0.7029934813	0.648713664	0.8181534145	0.835952361	0.76641051	0.962526303	39195	1853	11946	291
Test2	ML	SVM	Training, US	0.86701735	0.616830295	0.860701448	0.89536756	0.96184279	0.867017347	66721	1352	7787	2644	
Test1	ML	SVM	Training, US	0.895335517	0.677743569	0.904505599	0.9212477	0.88495445	0.99501576	0.895355166	45748	1961	5333	183

Nota. La gráfica muestra los resultados tabulados de *Machine Learning*. Elaboración propia.

Apéndice 21.

Matrices de confusión machine learning

Figuras E1. ML Algoritmo KNN - Training



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 22.

ML Algoritmo KNN Training Undersampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 23.

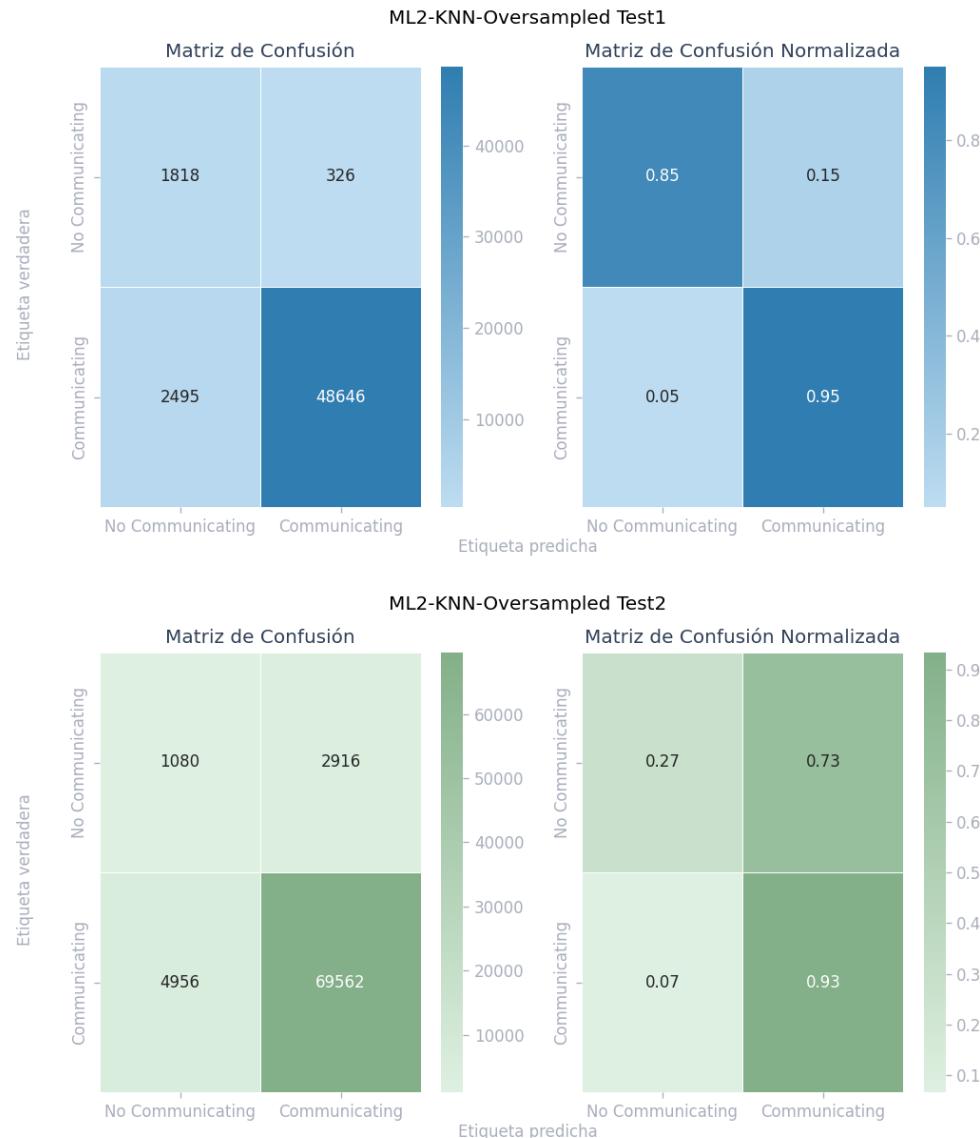
ML2 Algoritmo KNN Training Undersampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 24.

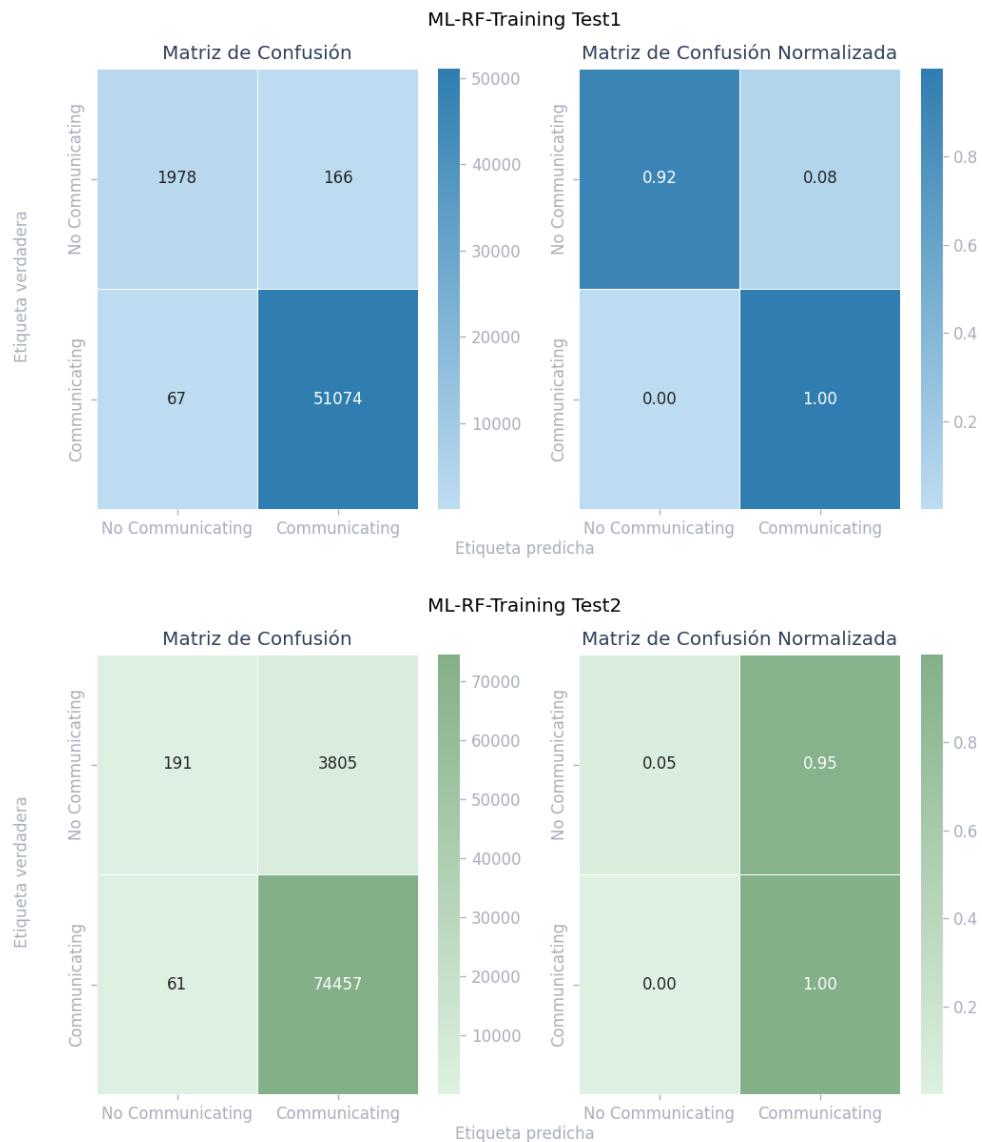
ML2 Algoritmo KNN Training Oversampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 25.

ML Algoritmo RF Training



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 26.

ML Algoritmo RF Training Undersampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 27.

ML Algoritmo RF Training Oversampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 28.

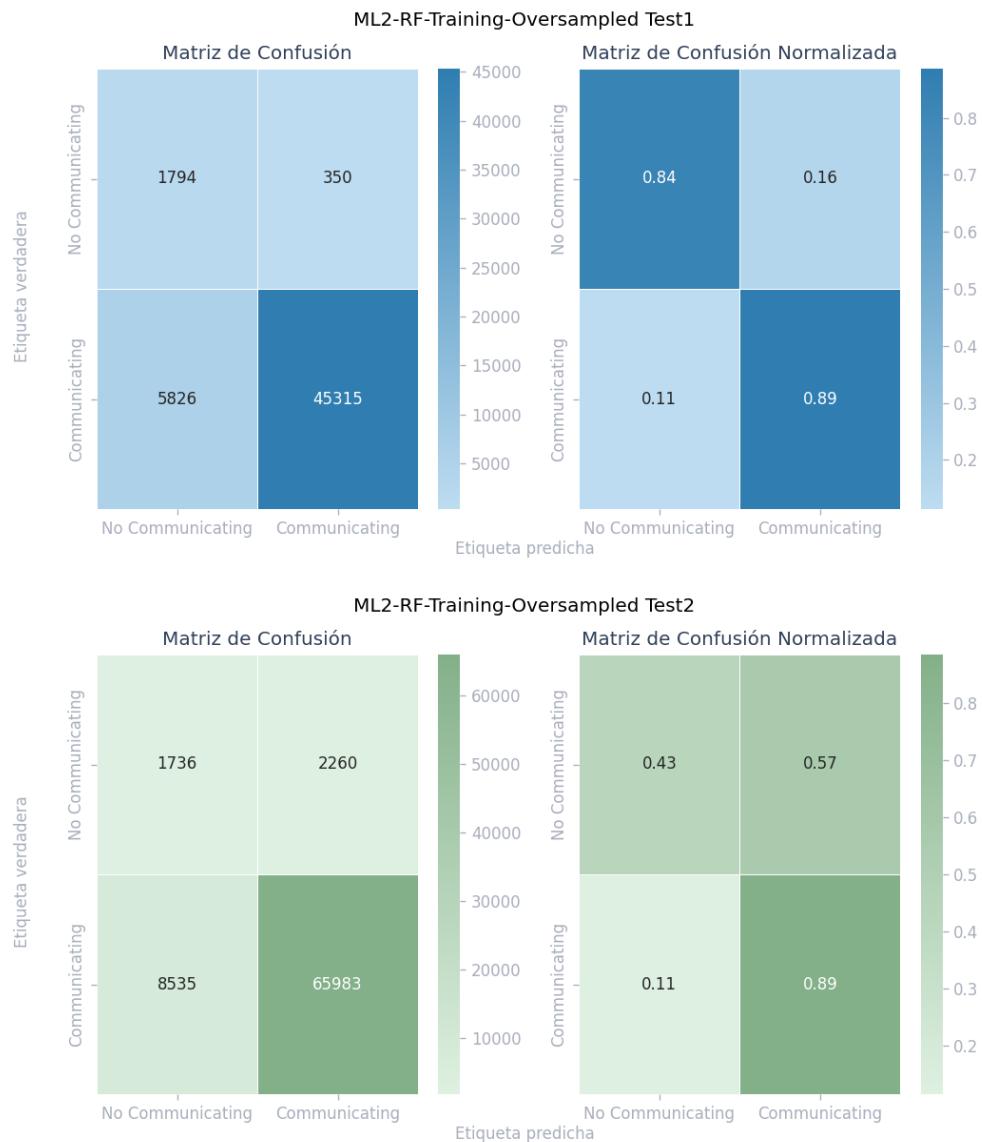
ML2 Algoritmo RF Training Undersampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 29.

ML2 Algoritmo RF Training Oversampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 30.

ML Algoritmo GBC Training Undersampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 31.

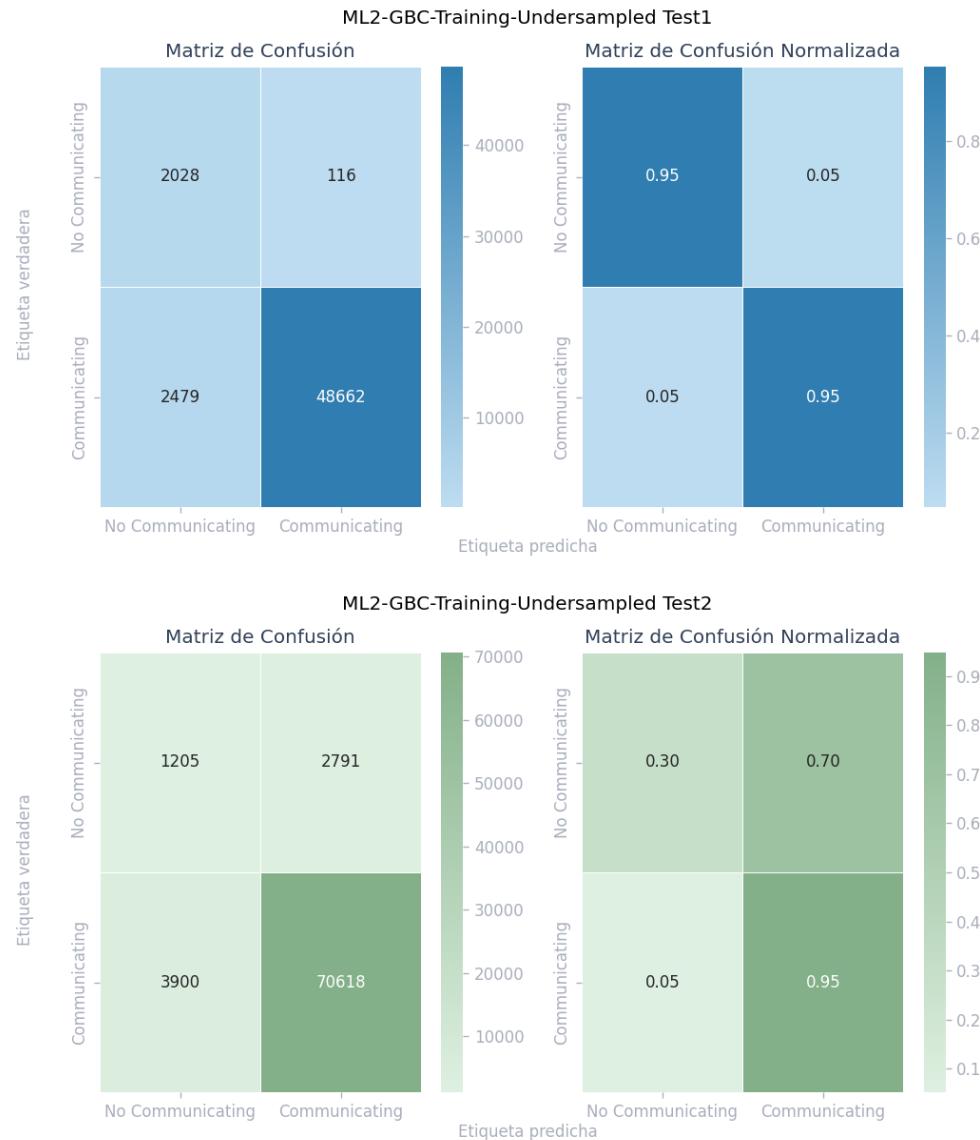
ML Algoritmo GBC Training Oversampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 32.

ML2 Algoritmo GBC Training Undersampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 33.

ML2 Algoritmo GBC Training Oversampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 34.

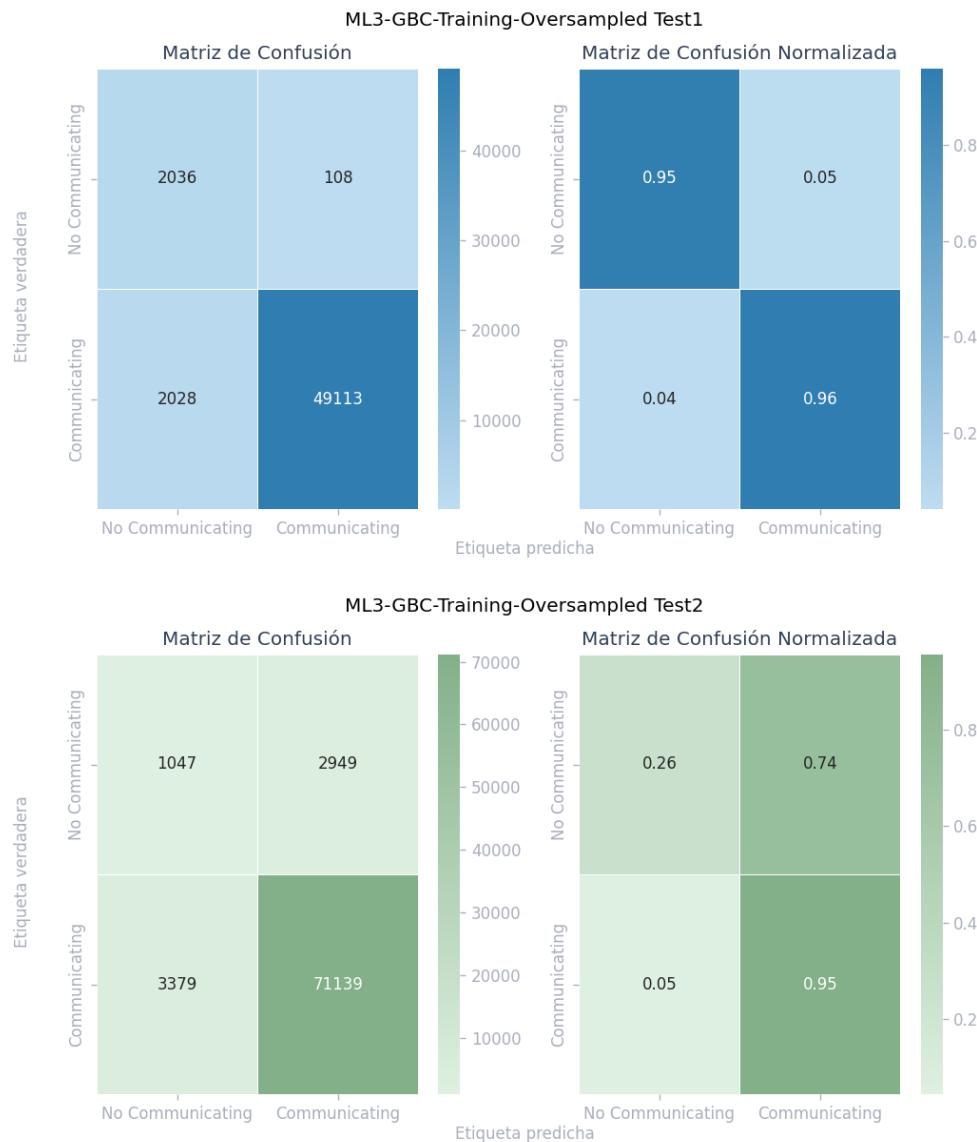
ML3 Algoritmo GBC Training Undersampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 35.

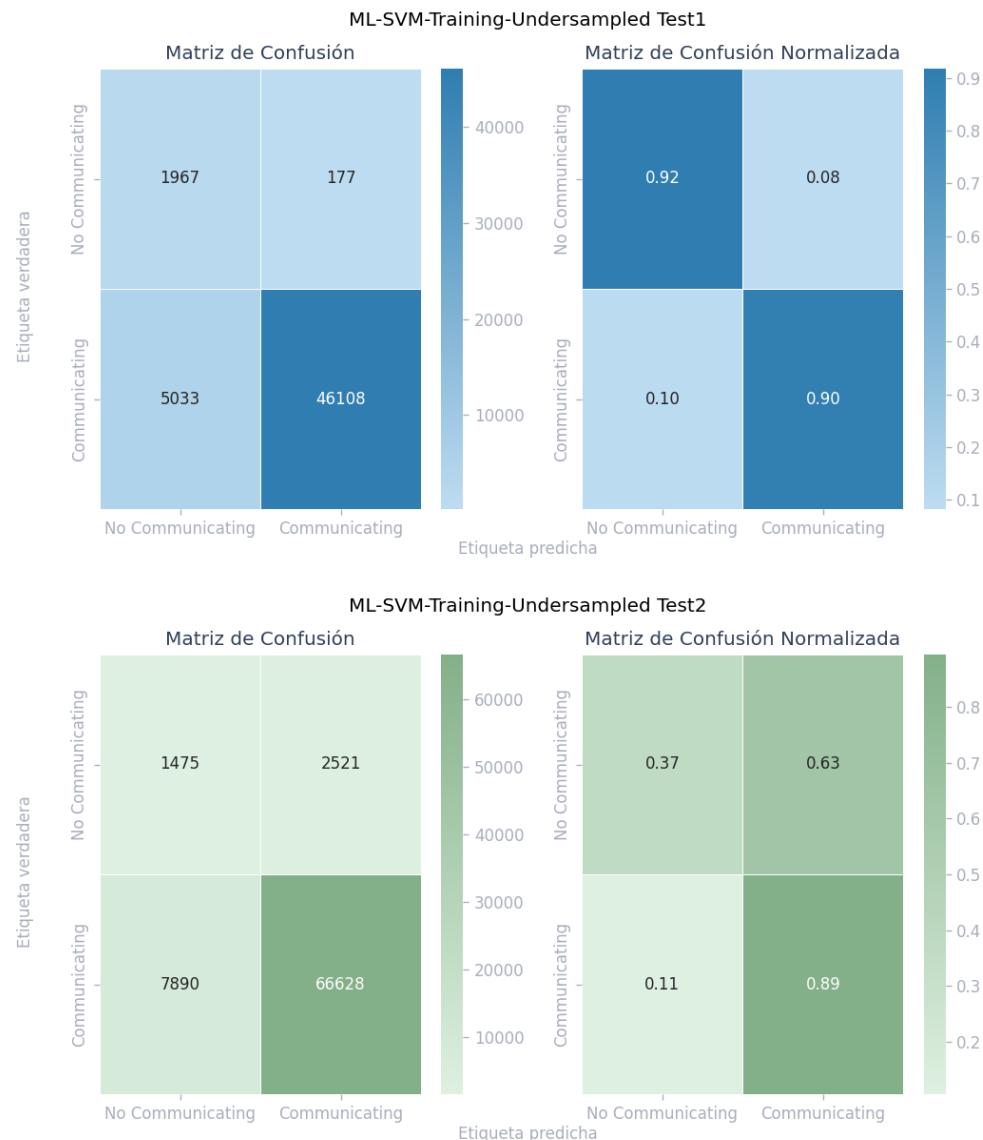
ML3 Algoritmo GBC Training Oversampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 36.

ML Algoritmo SVM Training Undersampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 37.

ML3 Algoritmo SVM Training Undersampled



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Machine Learning*. Elaboración propia, realizado con Python.

Apéndice 38.

Tabla de resultados Deep Learning parte 1

Nota. La gráfica muestra los resultados tabulados de *Deep Learning*. Elaboración propia.

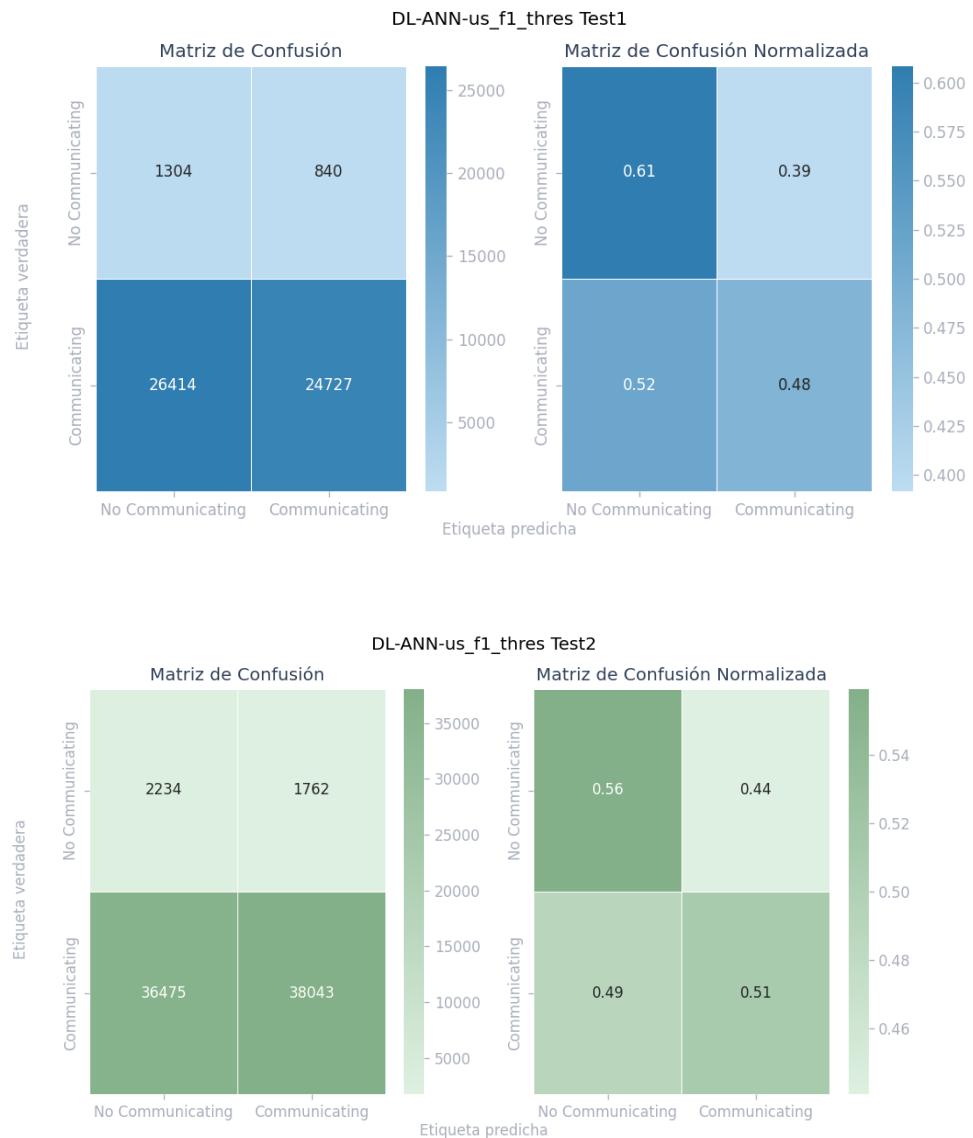
Apéndice 39.

Tabla de resultados Deep Learning parte 2

Nota. La gráfica muestra los resultados tabulados de *Deep Learning*. Elaboración propia.

Apéndice 40.

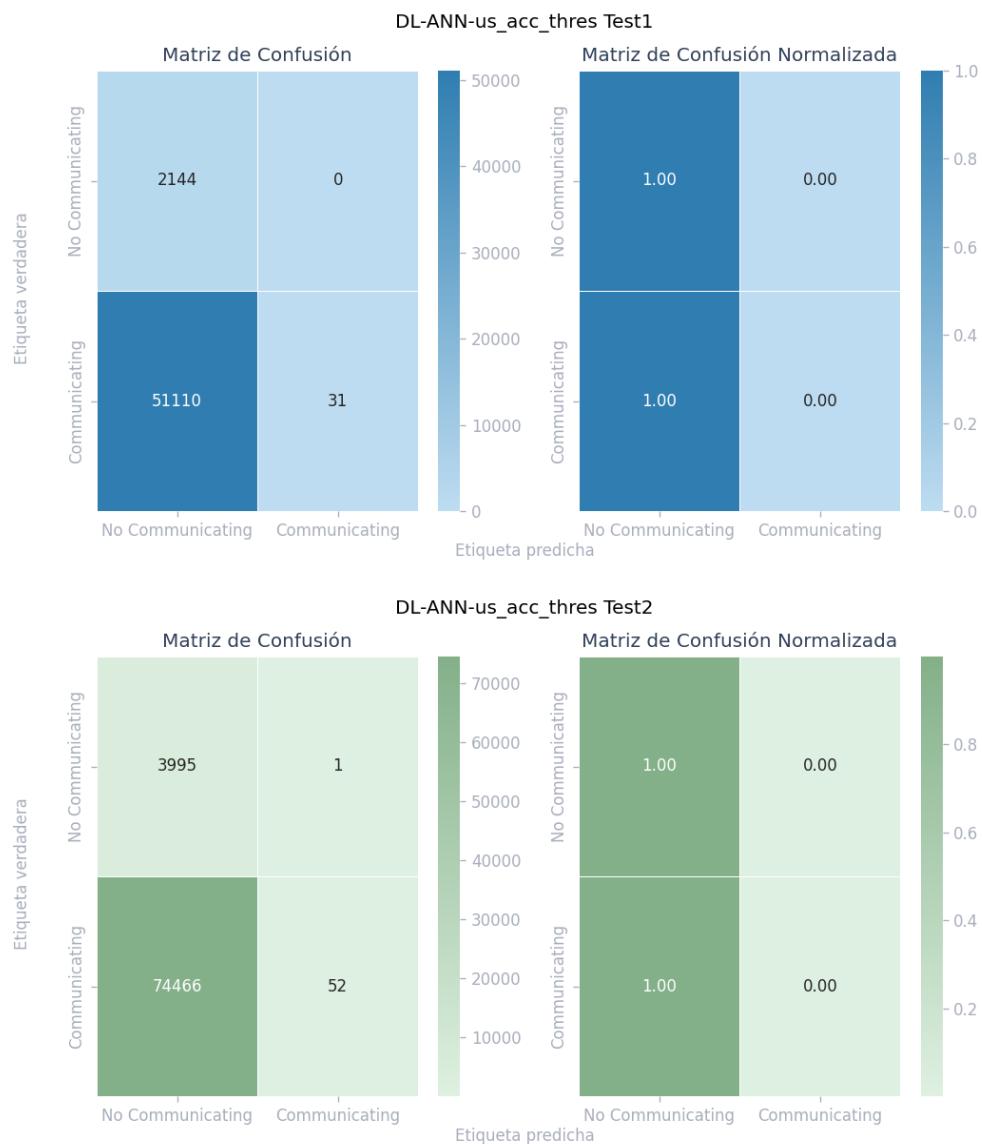
DL ANN Undersampled Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 41.

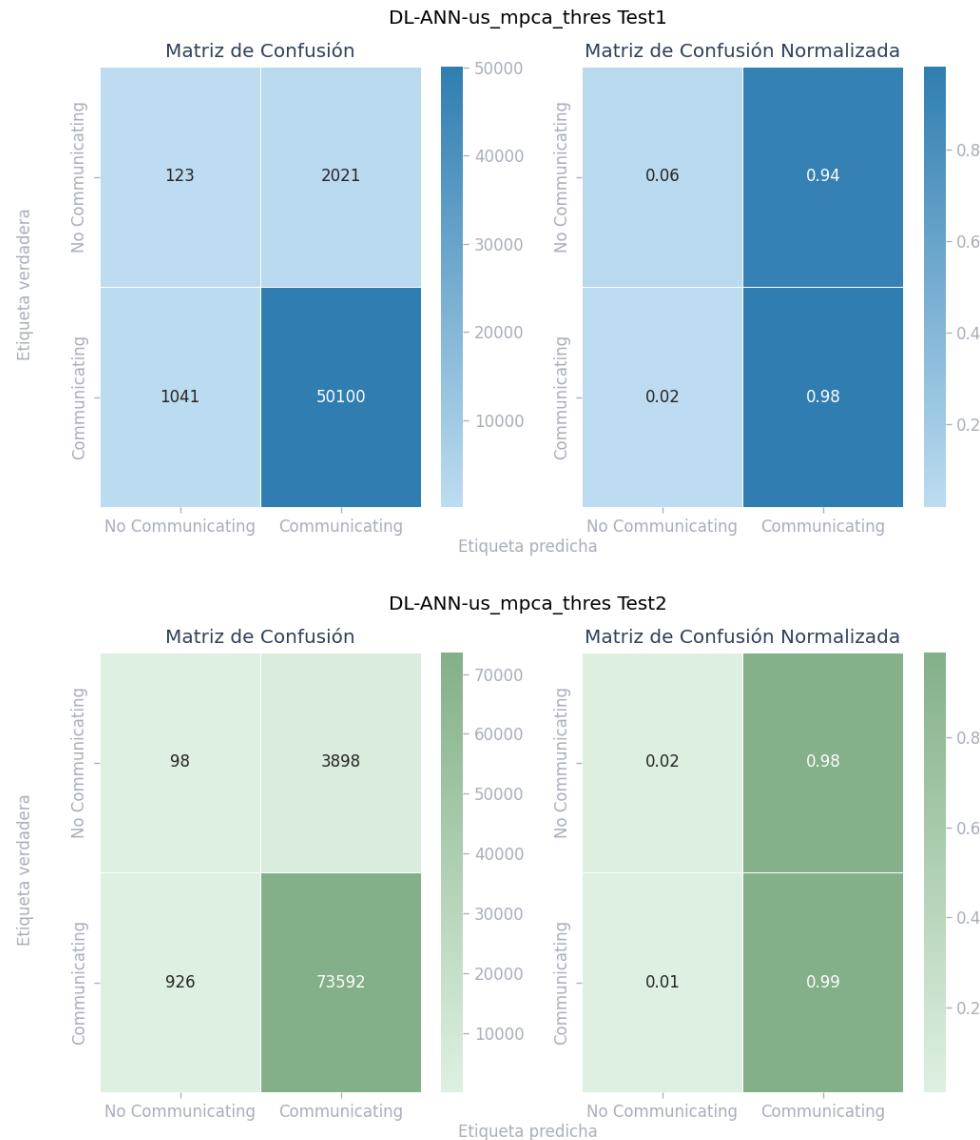
DL ANN Undersampled Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 42.

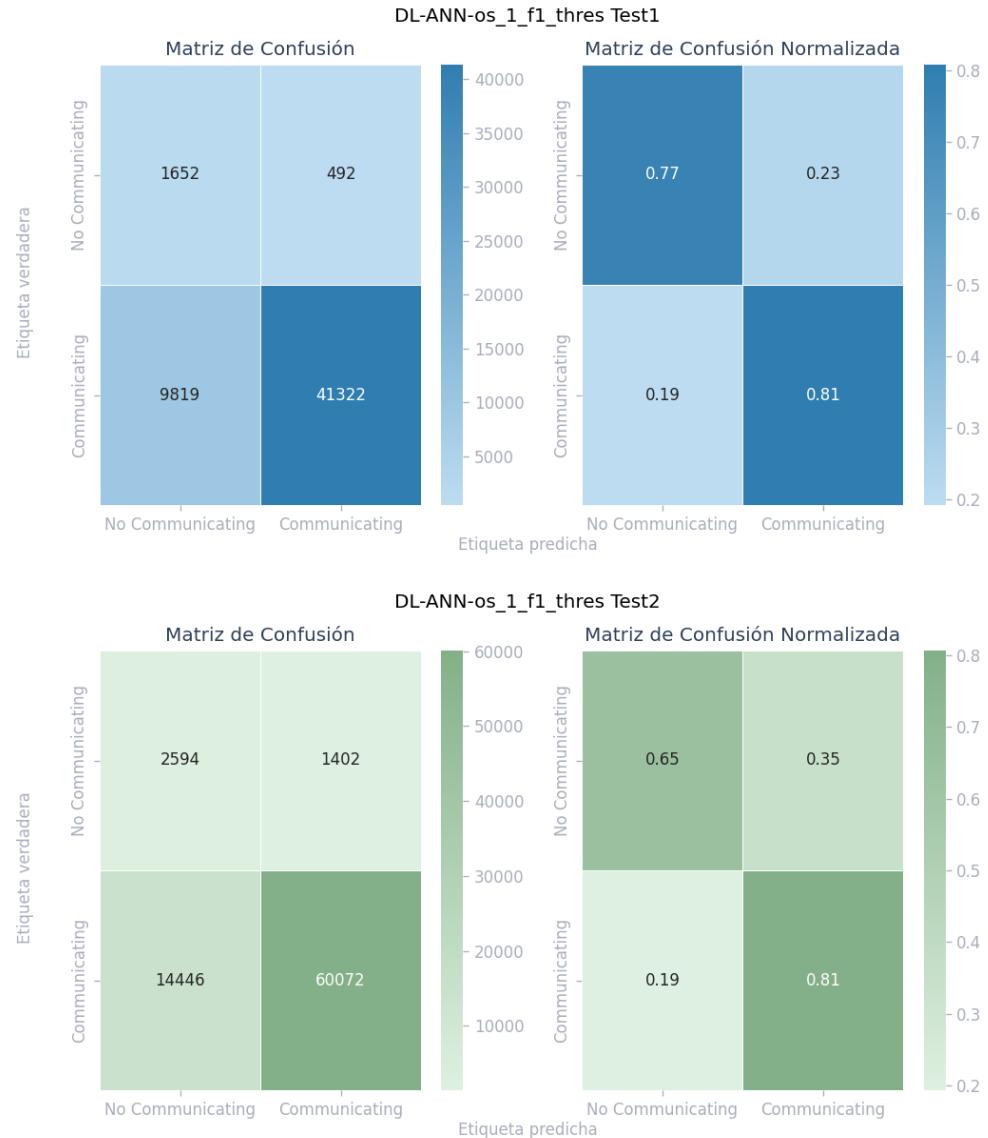
DL ANN Undersampled Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 43.

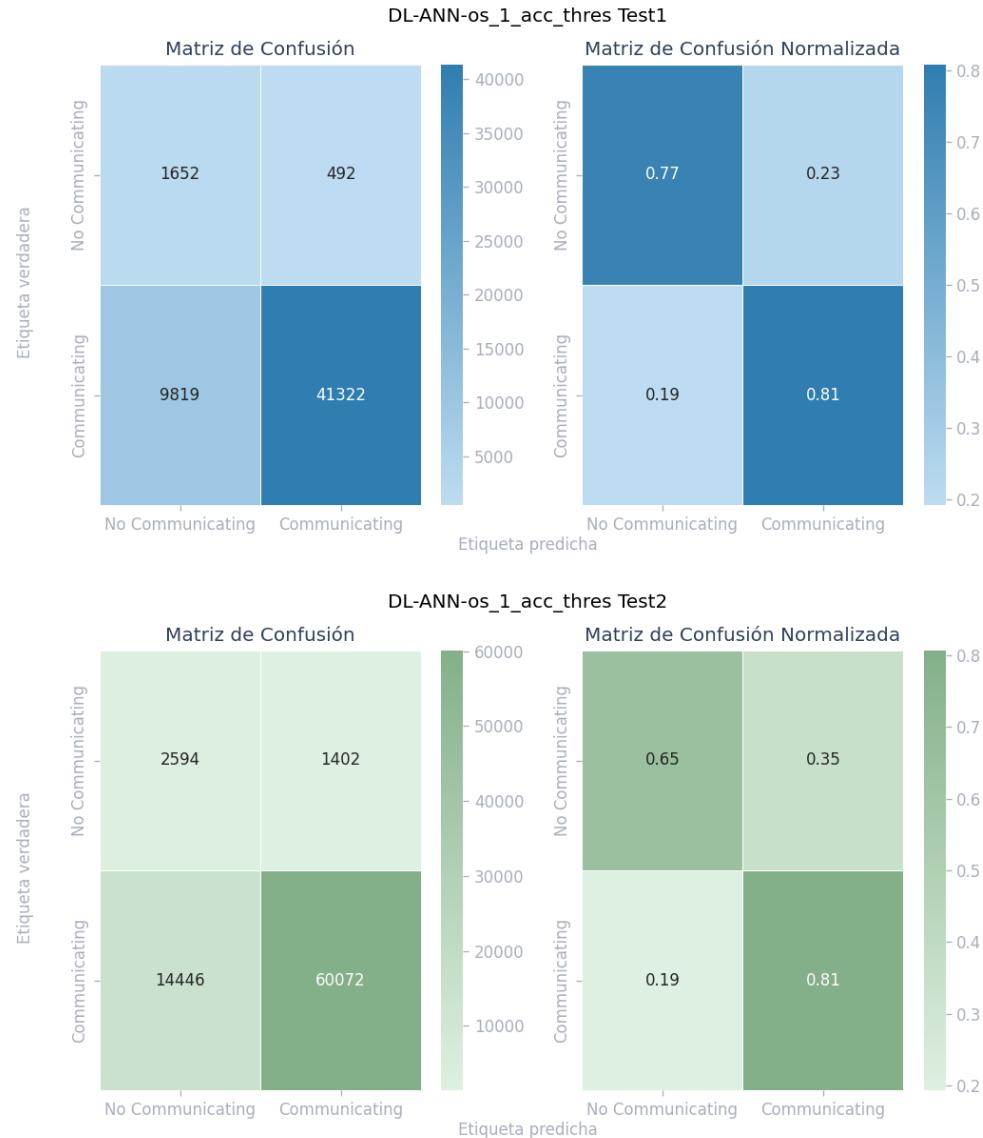
DL ANN Oversampled_1 Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 44.

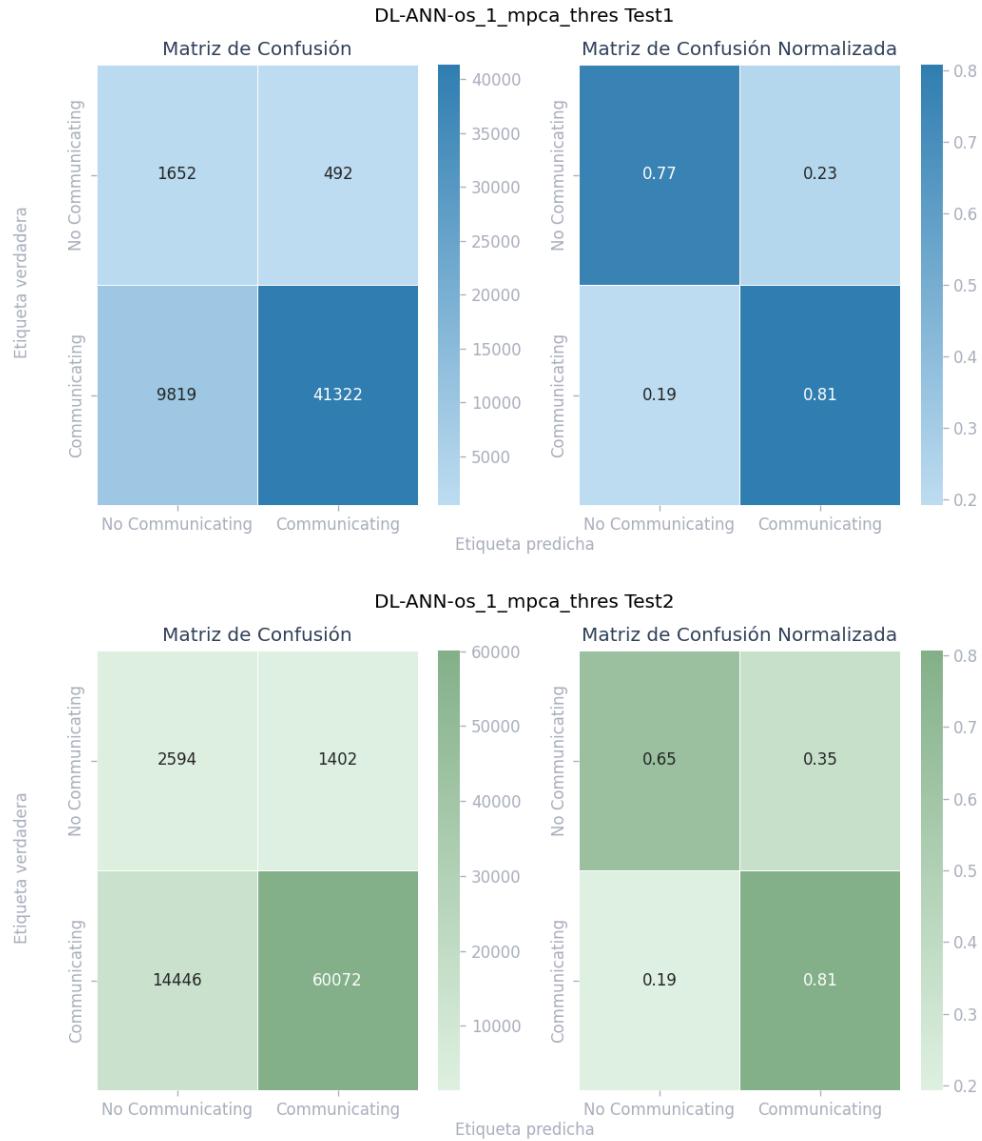
DL ANN Oversampled_1 Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 45.

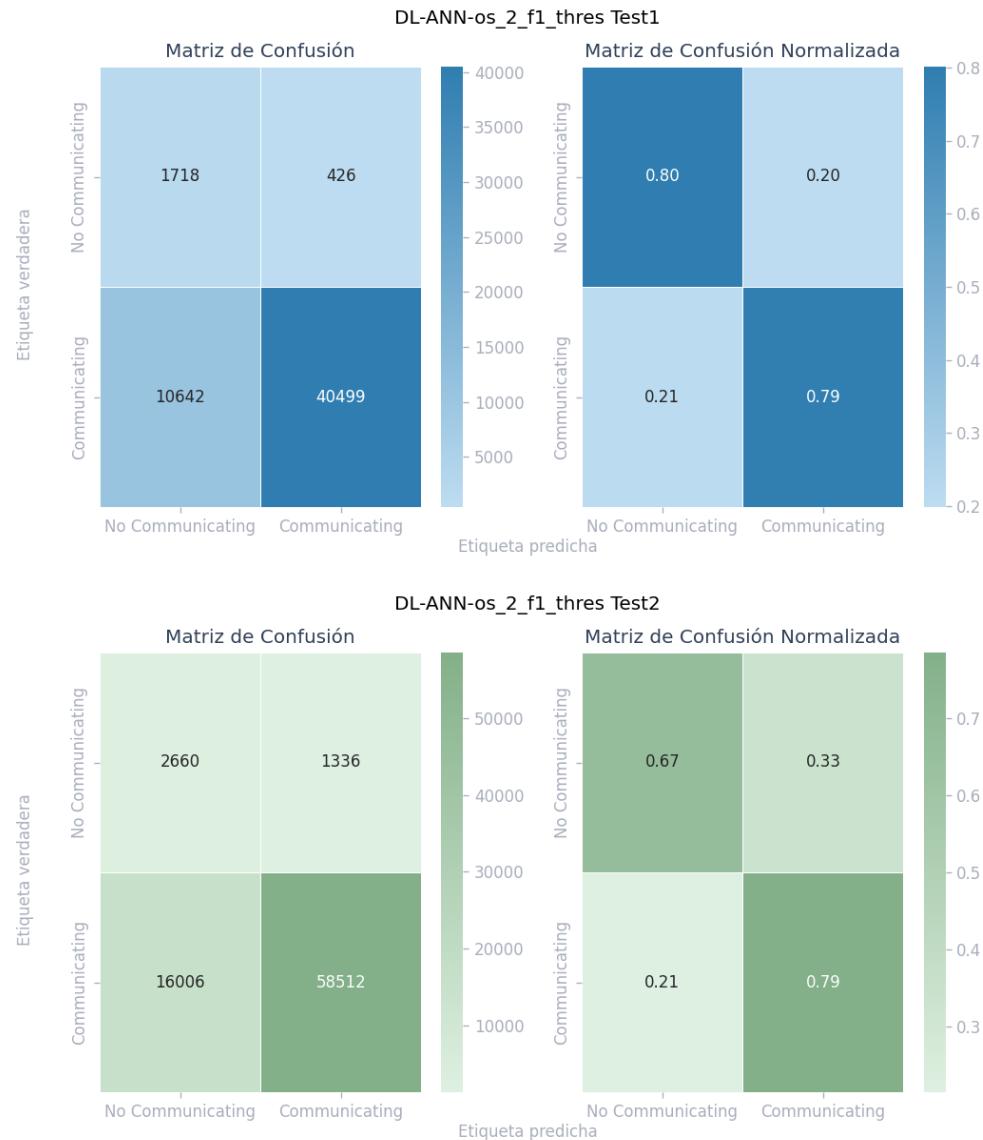
DL ANN Oversampled_1 Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 46.

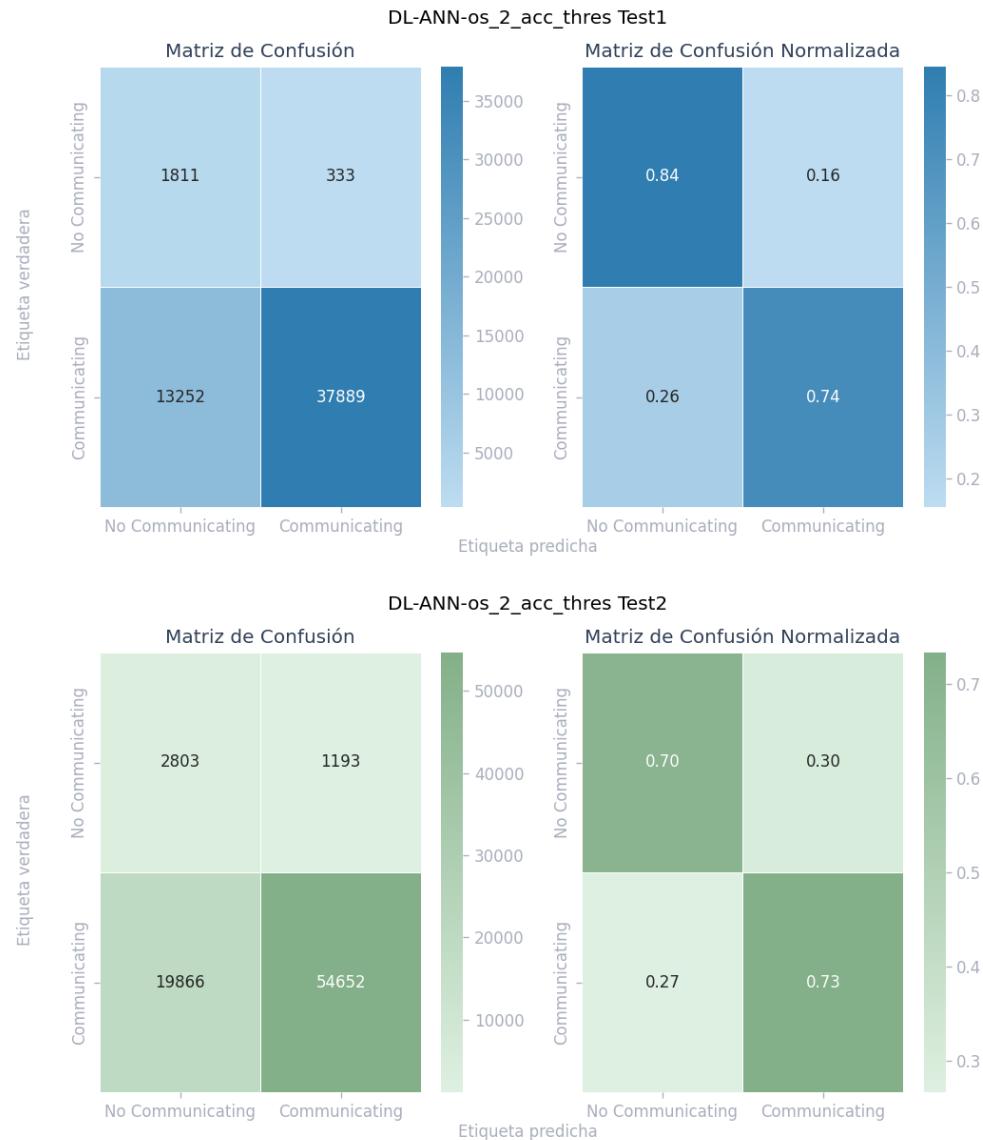
DL ANN Oversampled_2 Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 47.

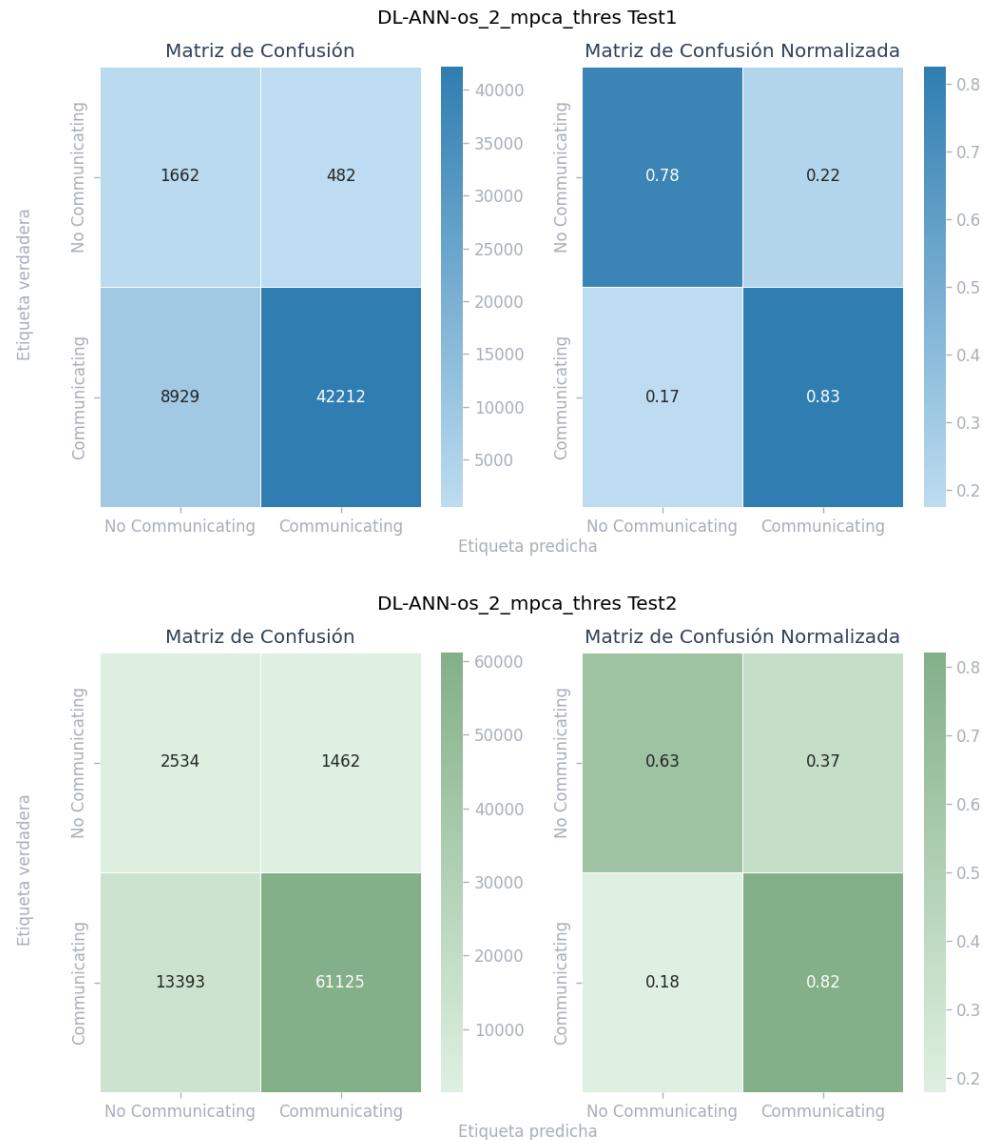
DL ANN Oversampled_2 Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 48.

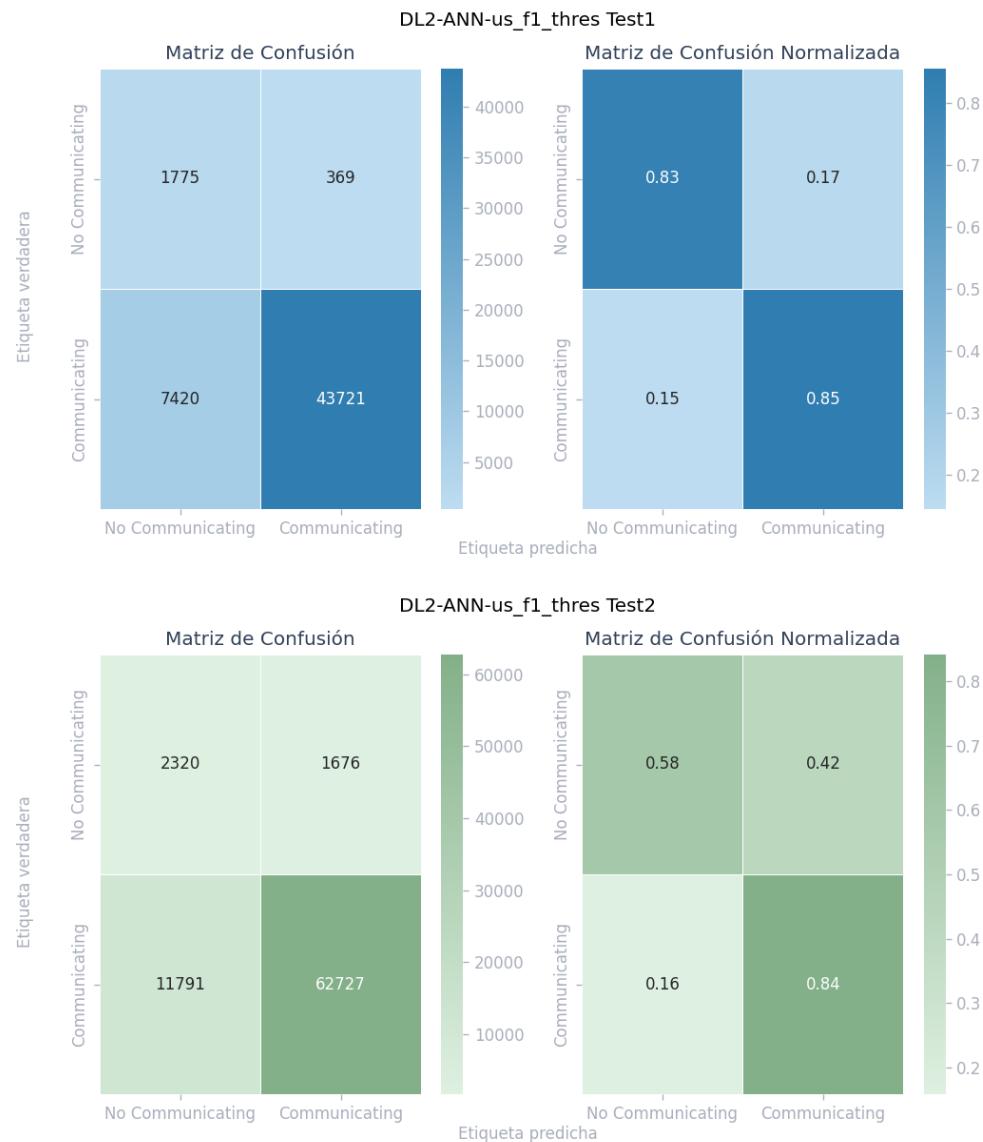
DL ANN Oversampled_2 Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 49.

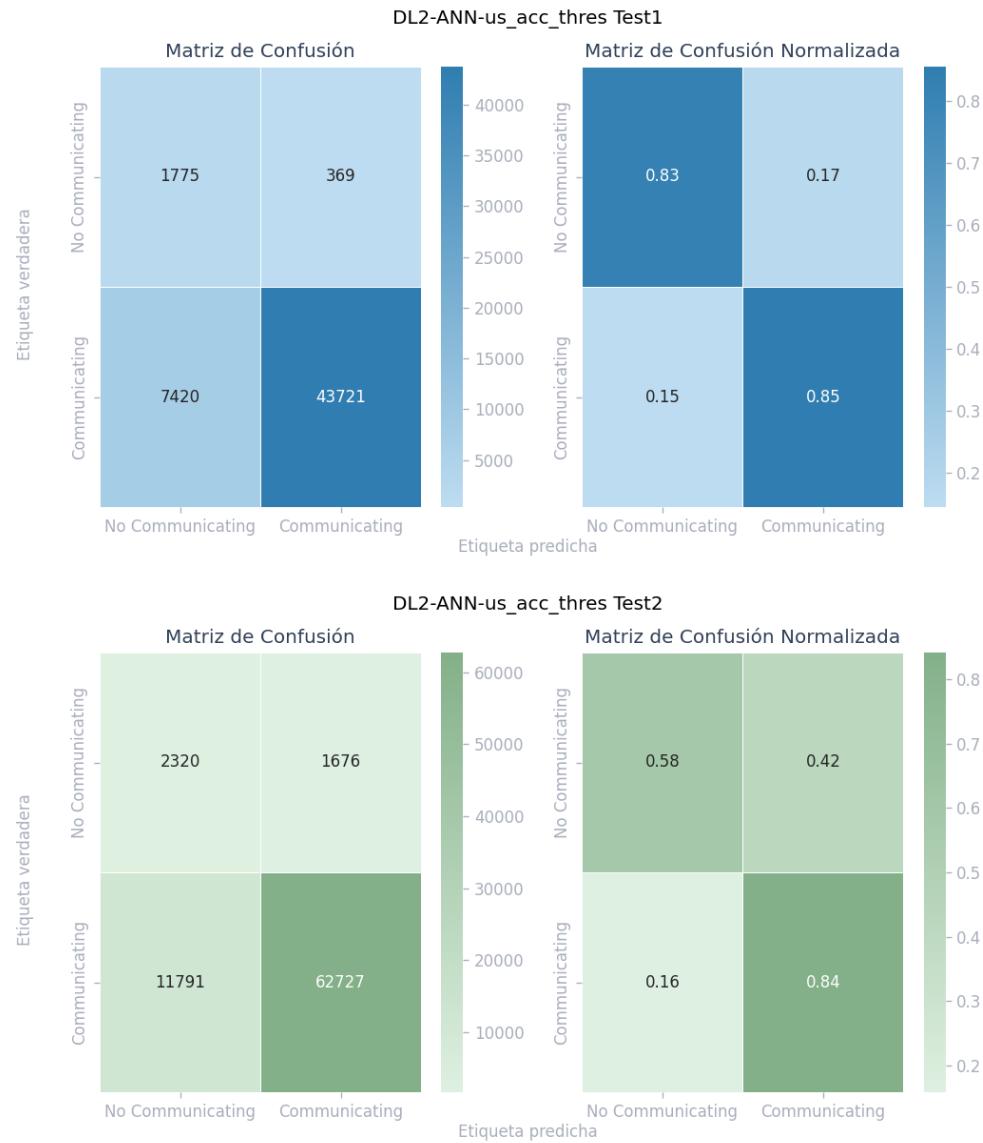
DL2 ANN Undersampled Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 50.

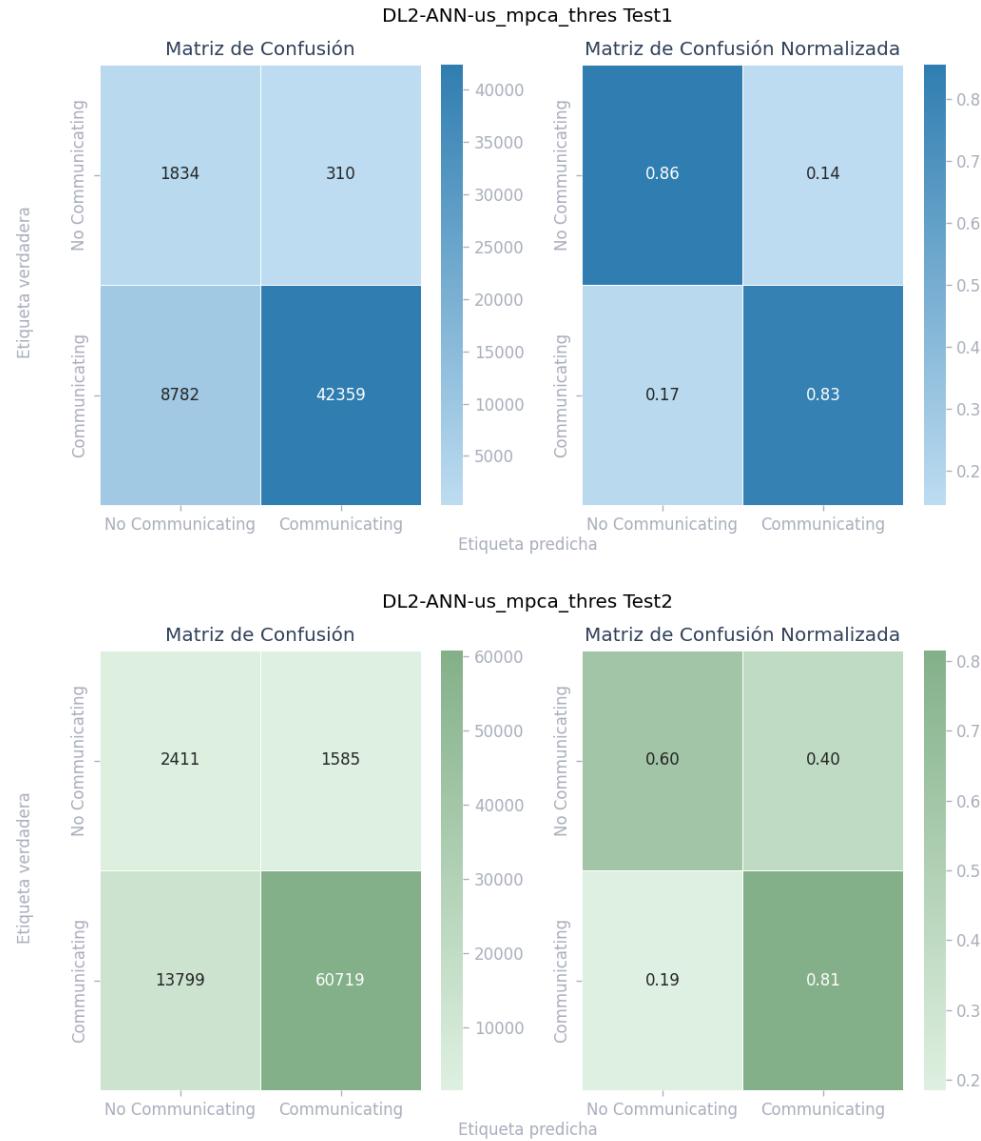
DL2 ANN Undersampled Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 51.

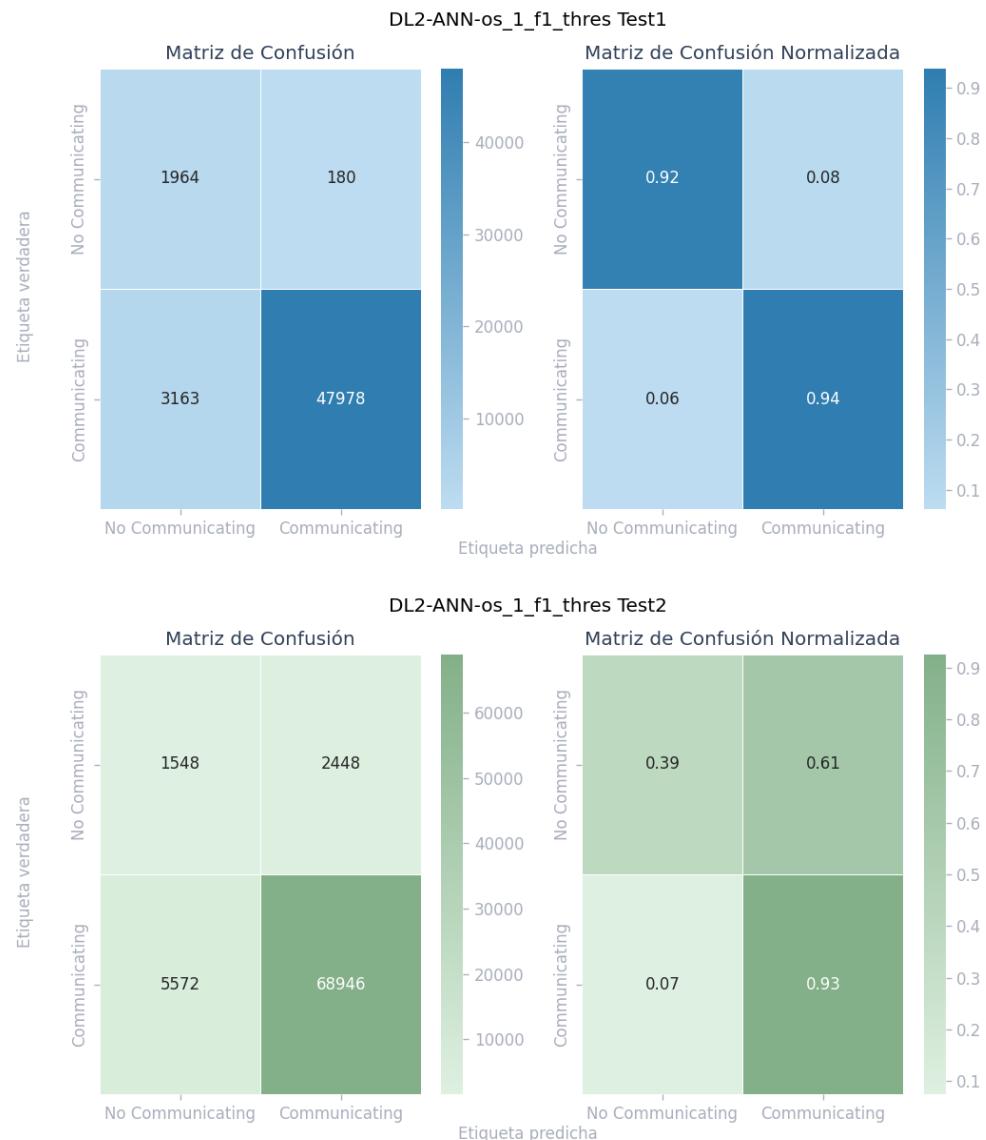
DL2 ANN Undersampled Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 52.

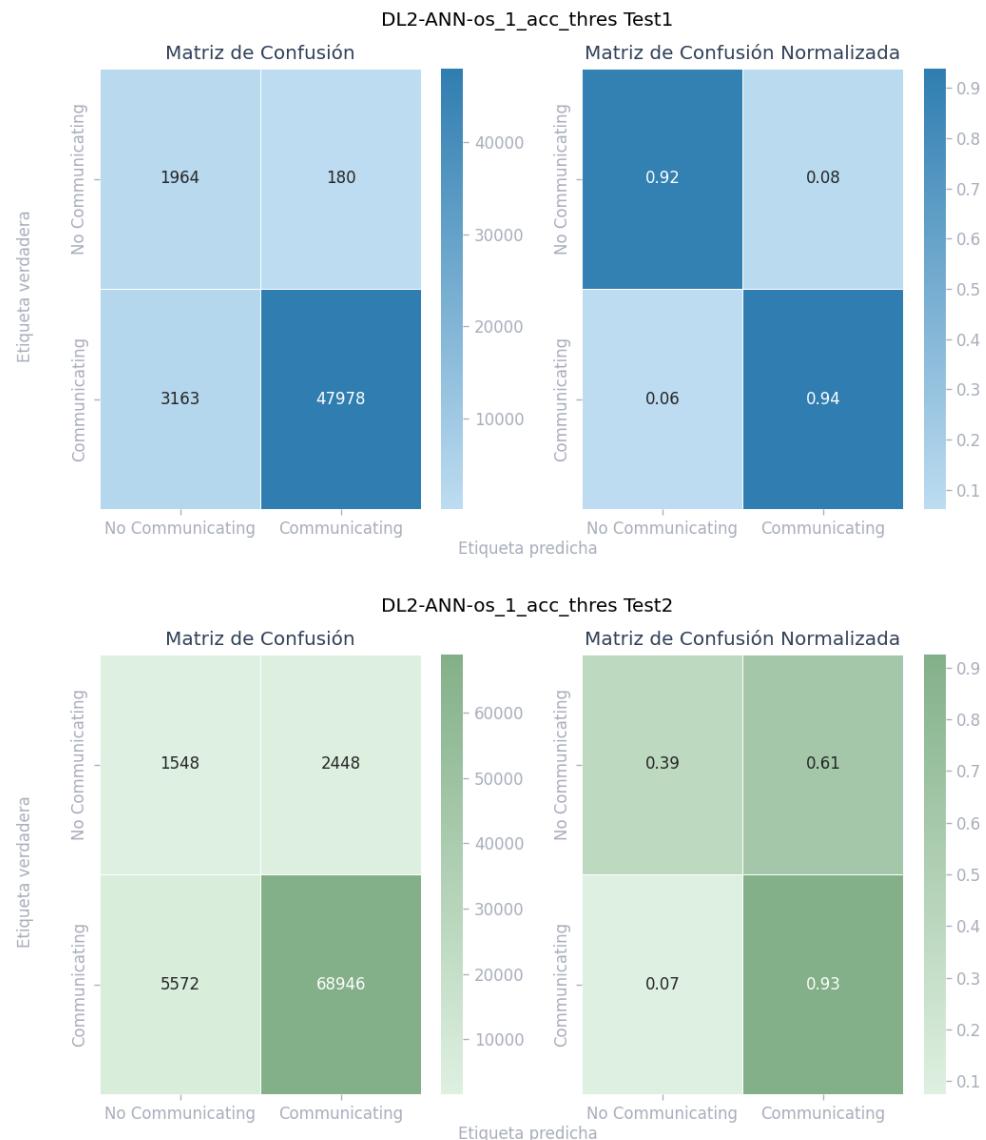
DL2 ANN Oversampled_1 Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 53.

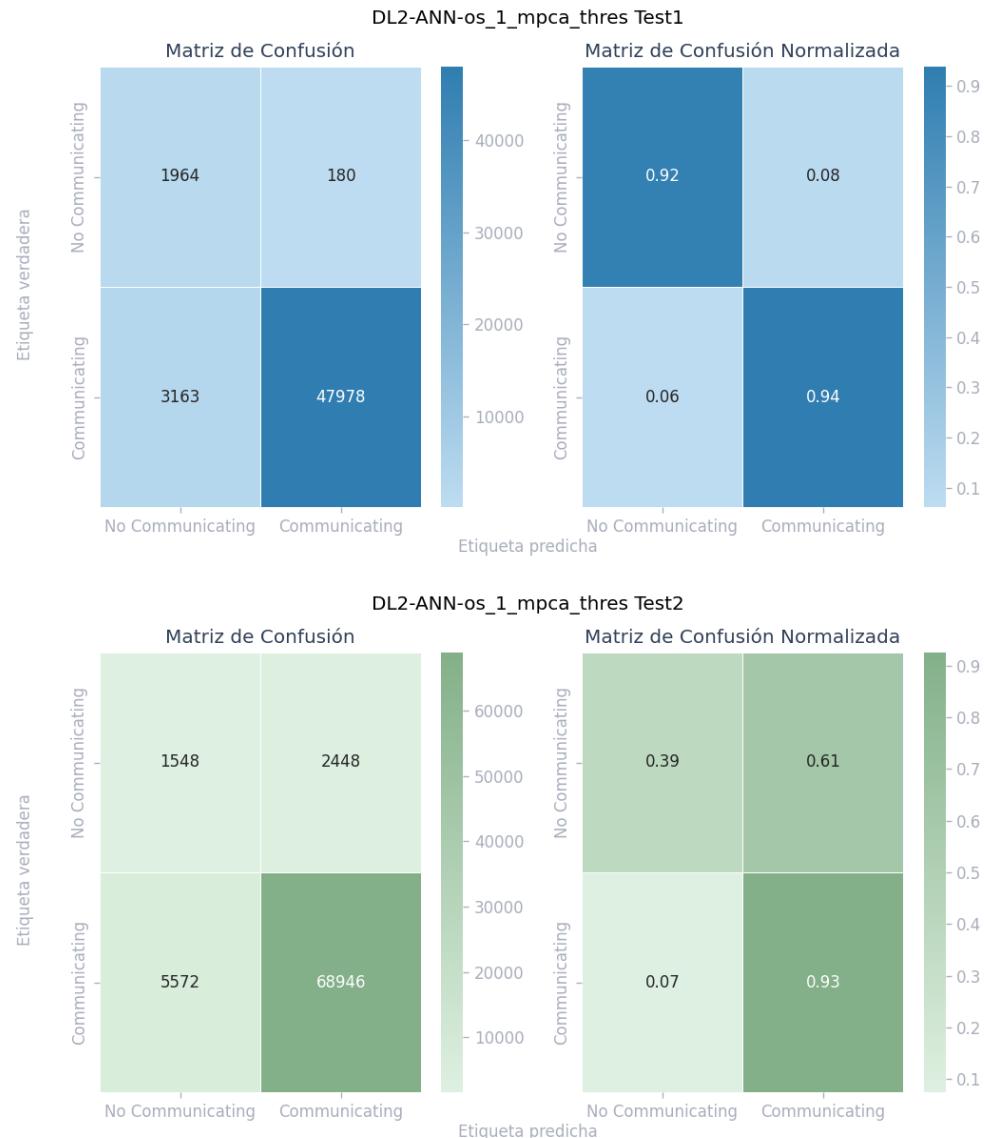
DL2 ANN Oversampled_1 Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 54.

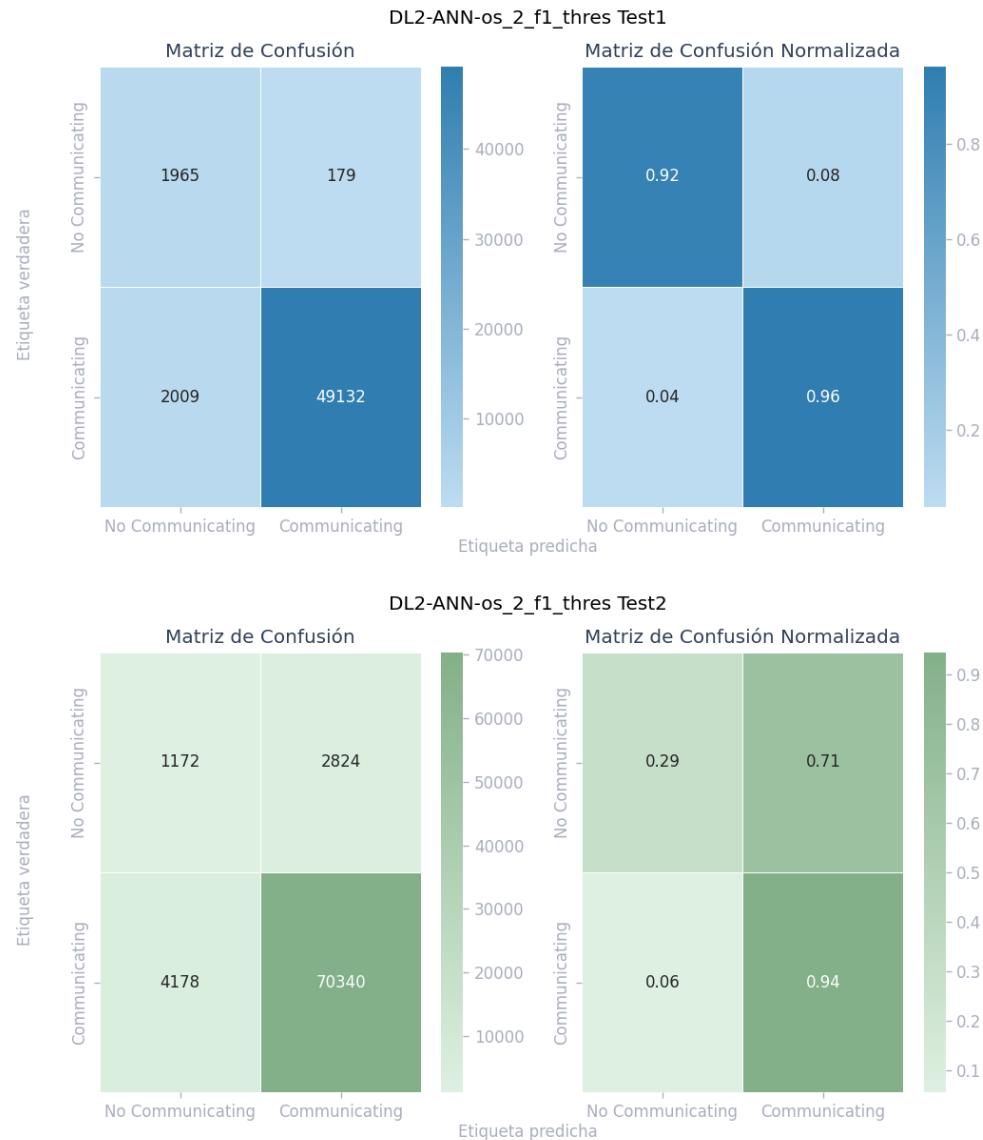
DL2 ANN Oversampled_1 Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 55.

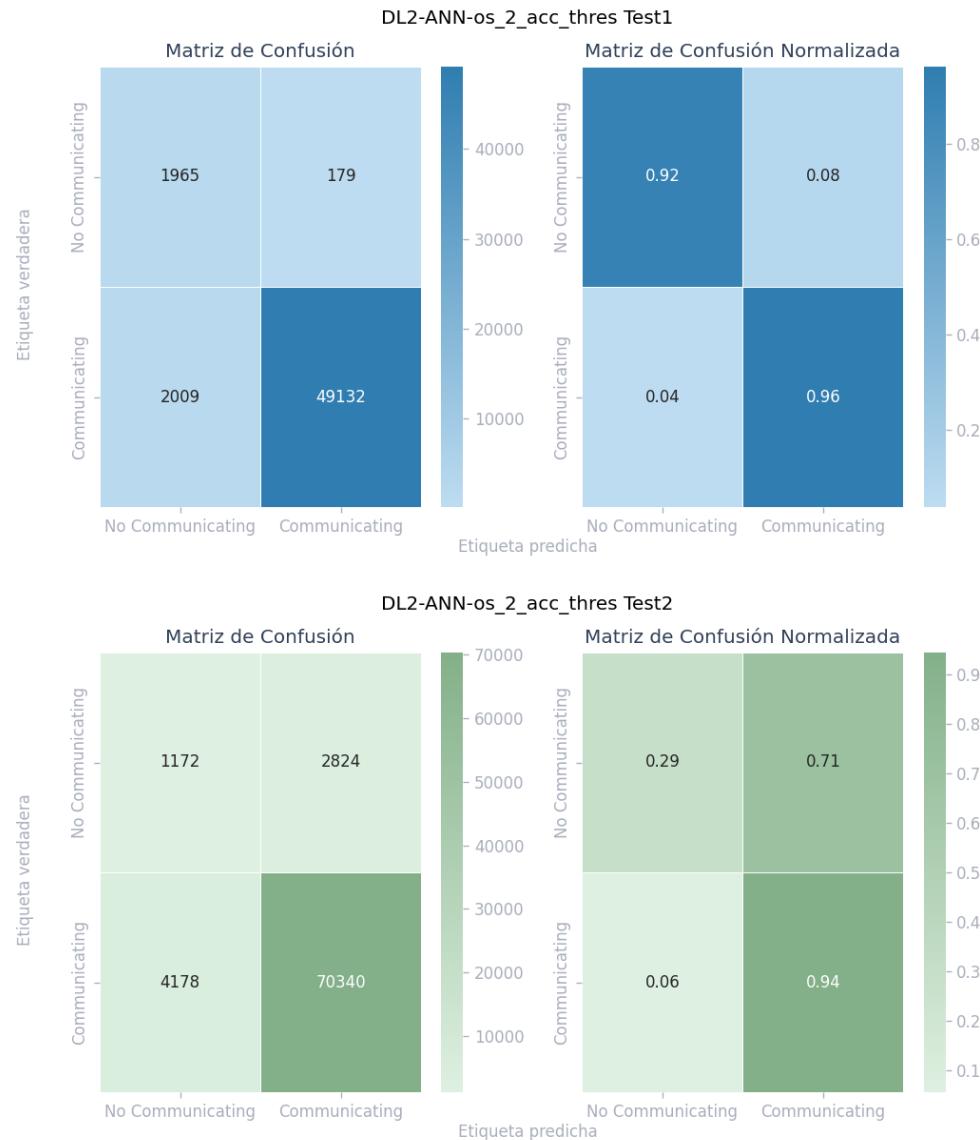
DL2 ANN Oversampled_2 Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 56.

DL2 ANN Oversampled_2 Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 57.

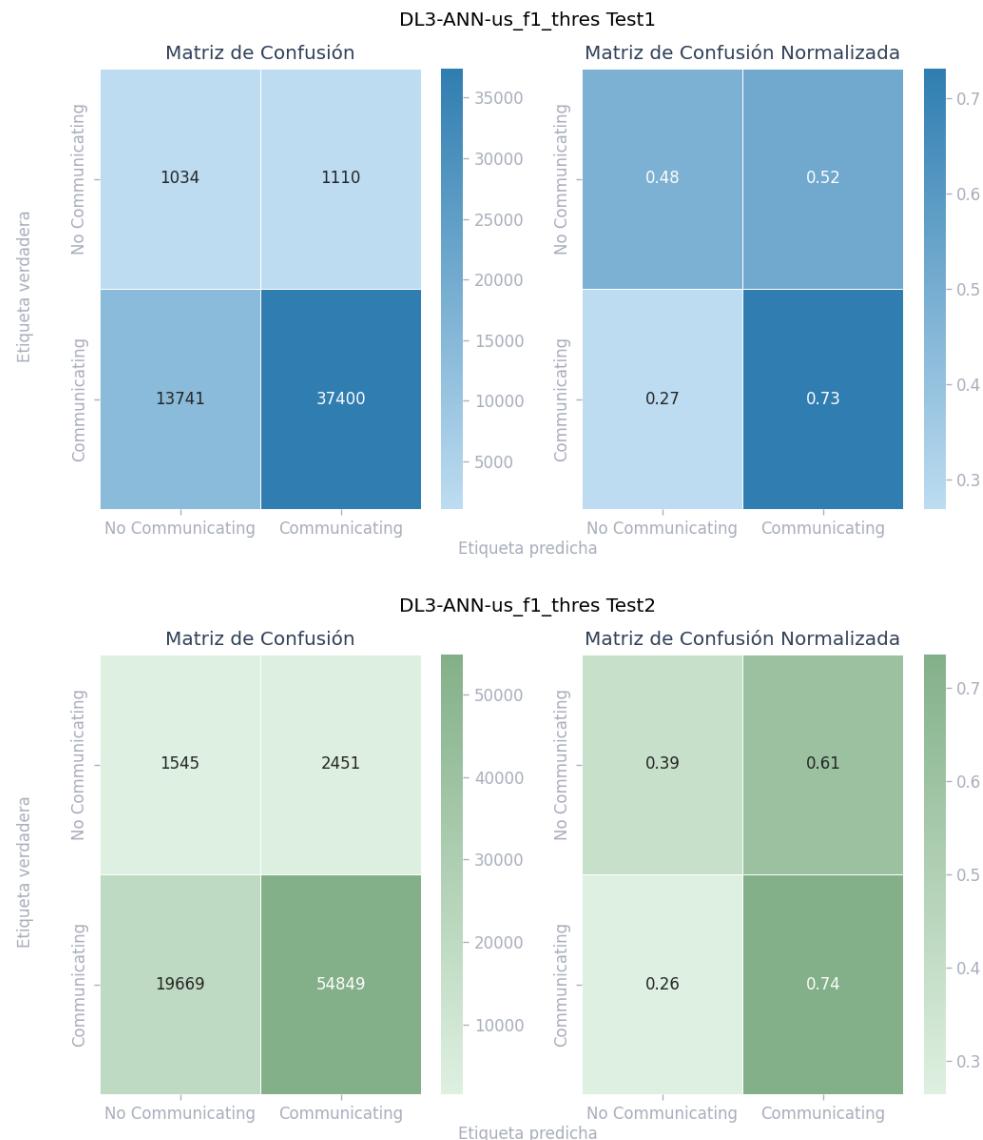
DL2 ANN Oversampled_2 Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 58.

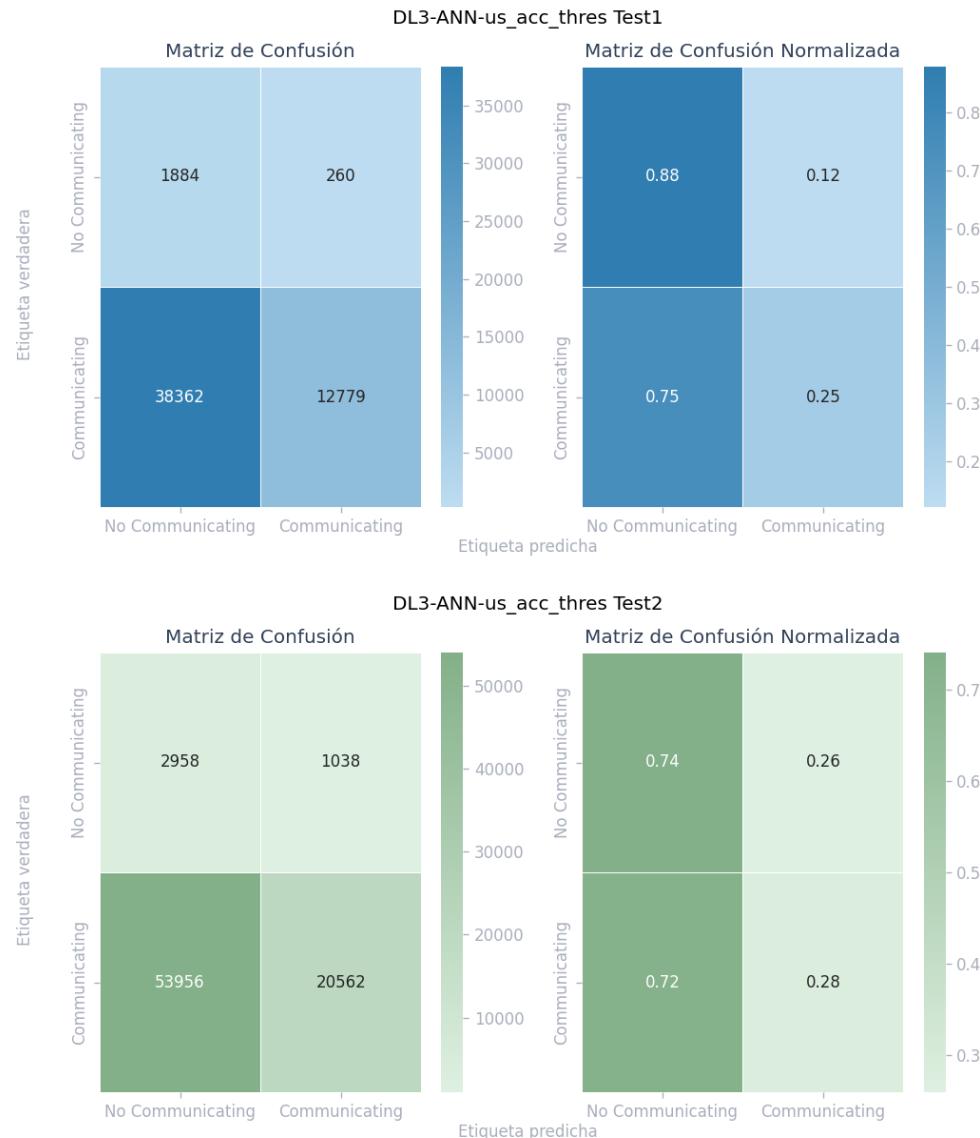
DL3 ANN Undersampled Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 59.

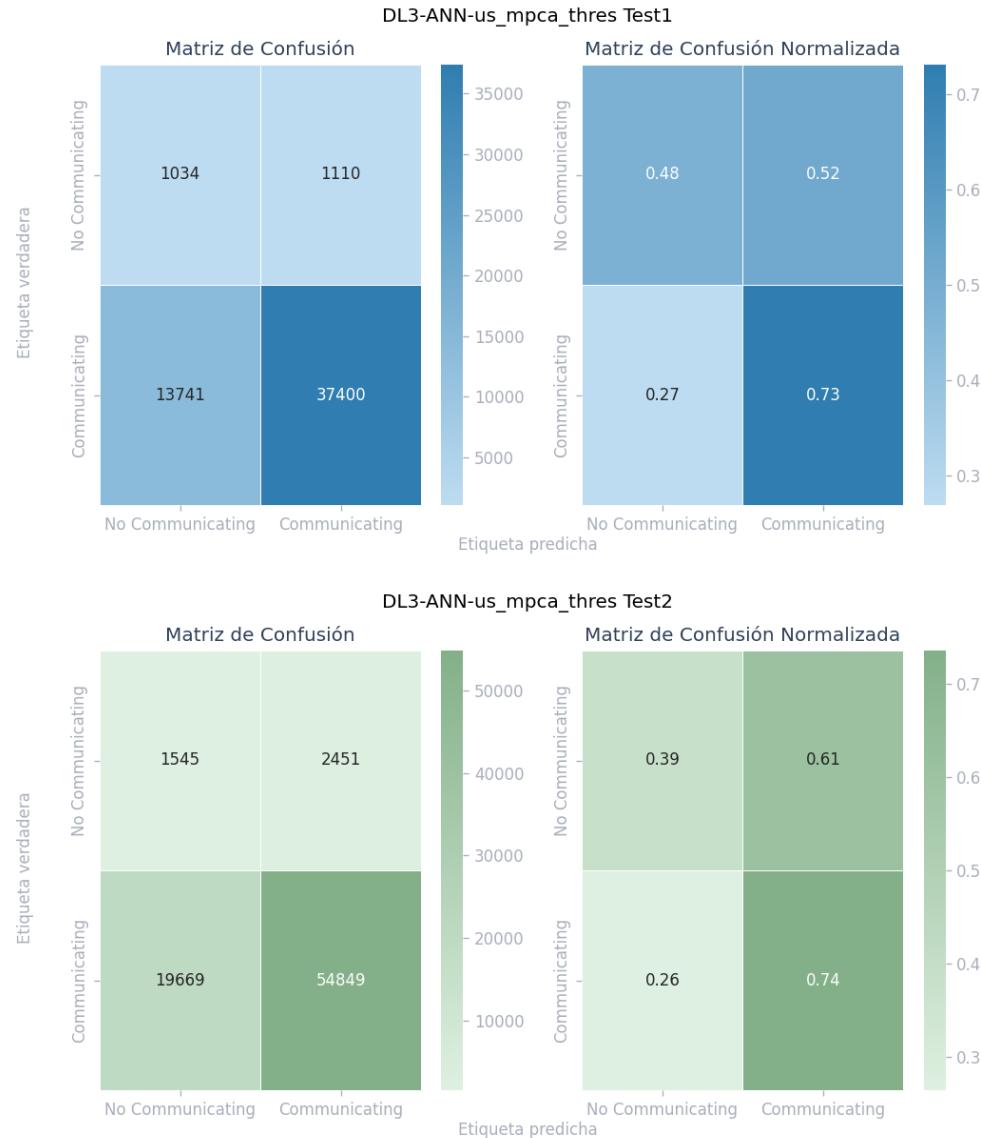
DL3 ANN Undersampled Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 60.

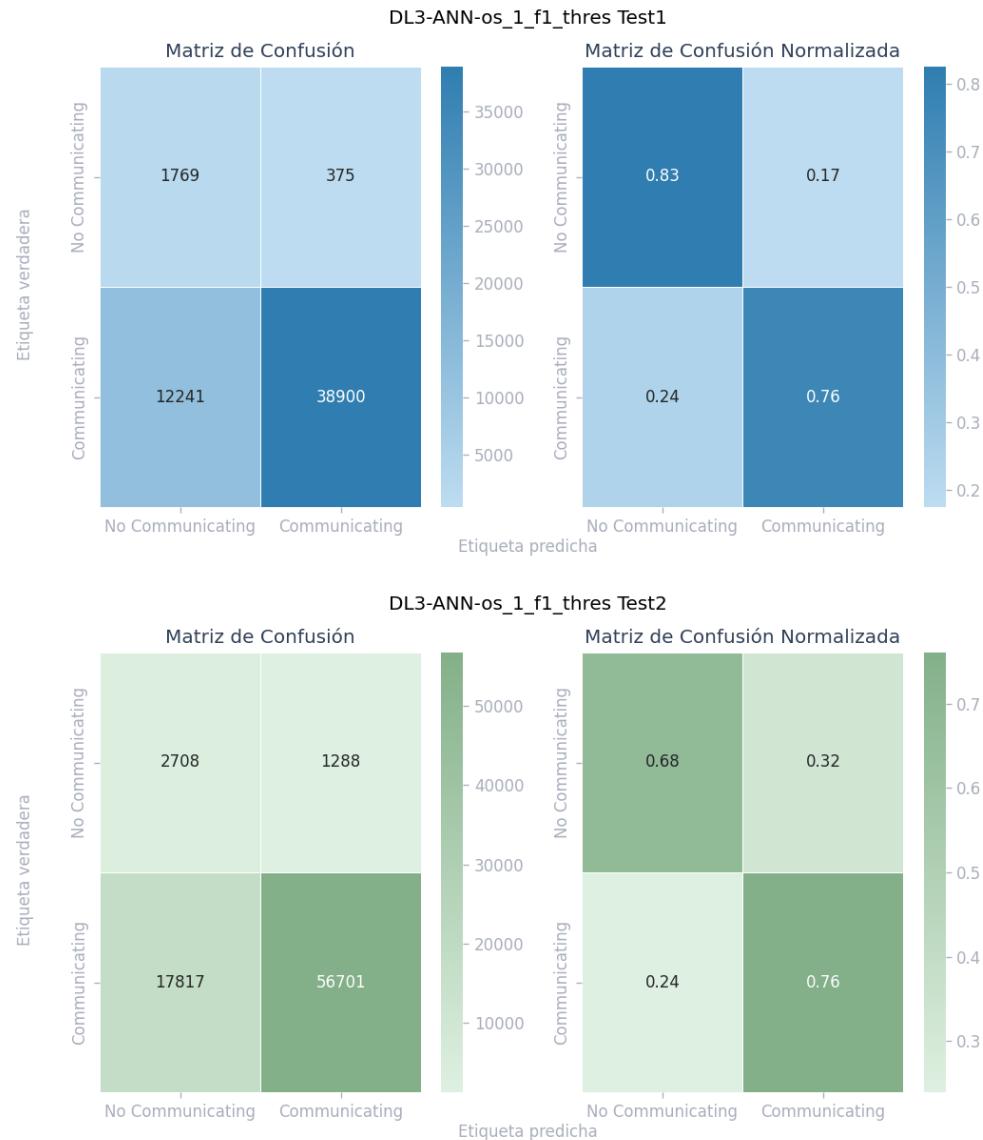
DL3 ANN Undersampled Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 61.

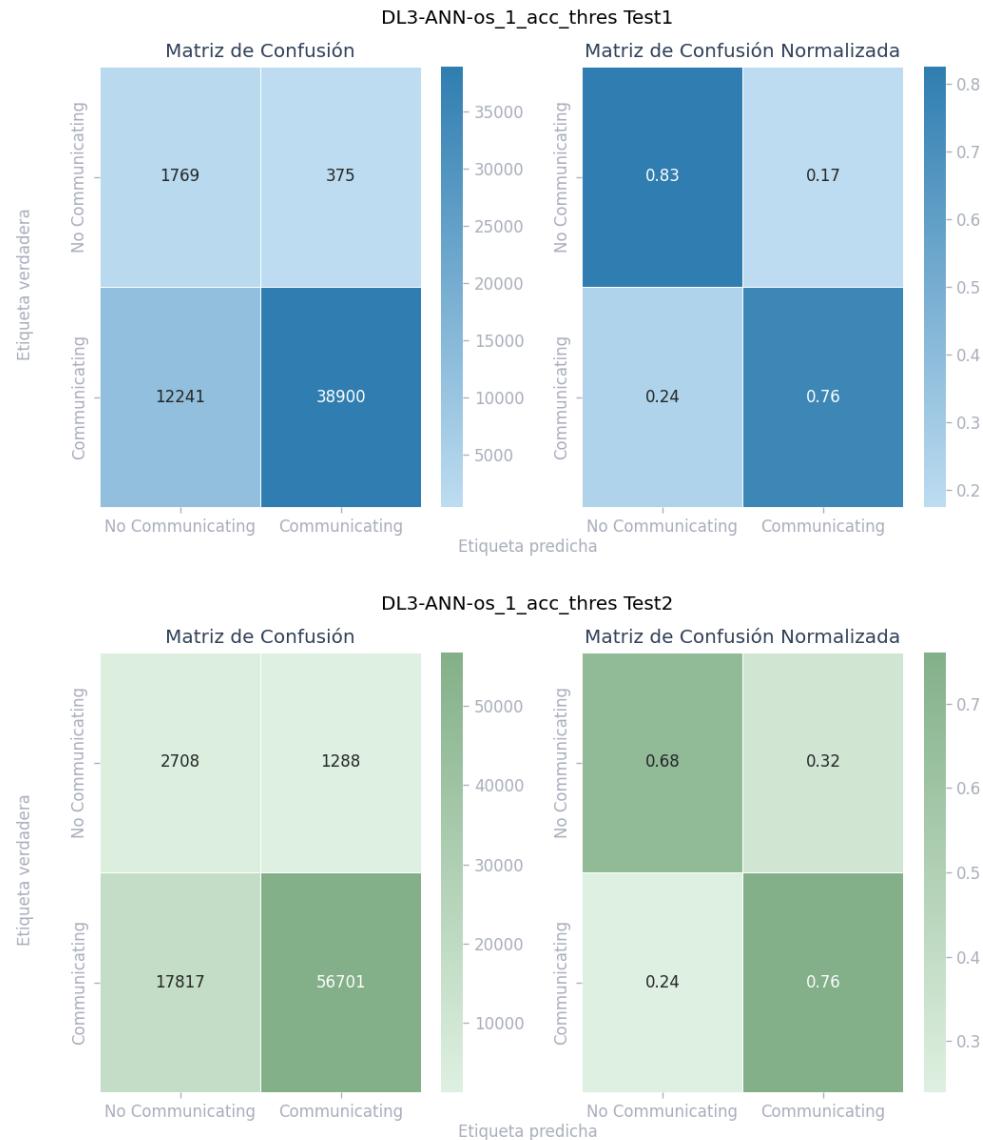
DL3 ANN Oversampled_1 Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 62.

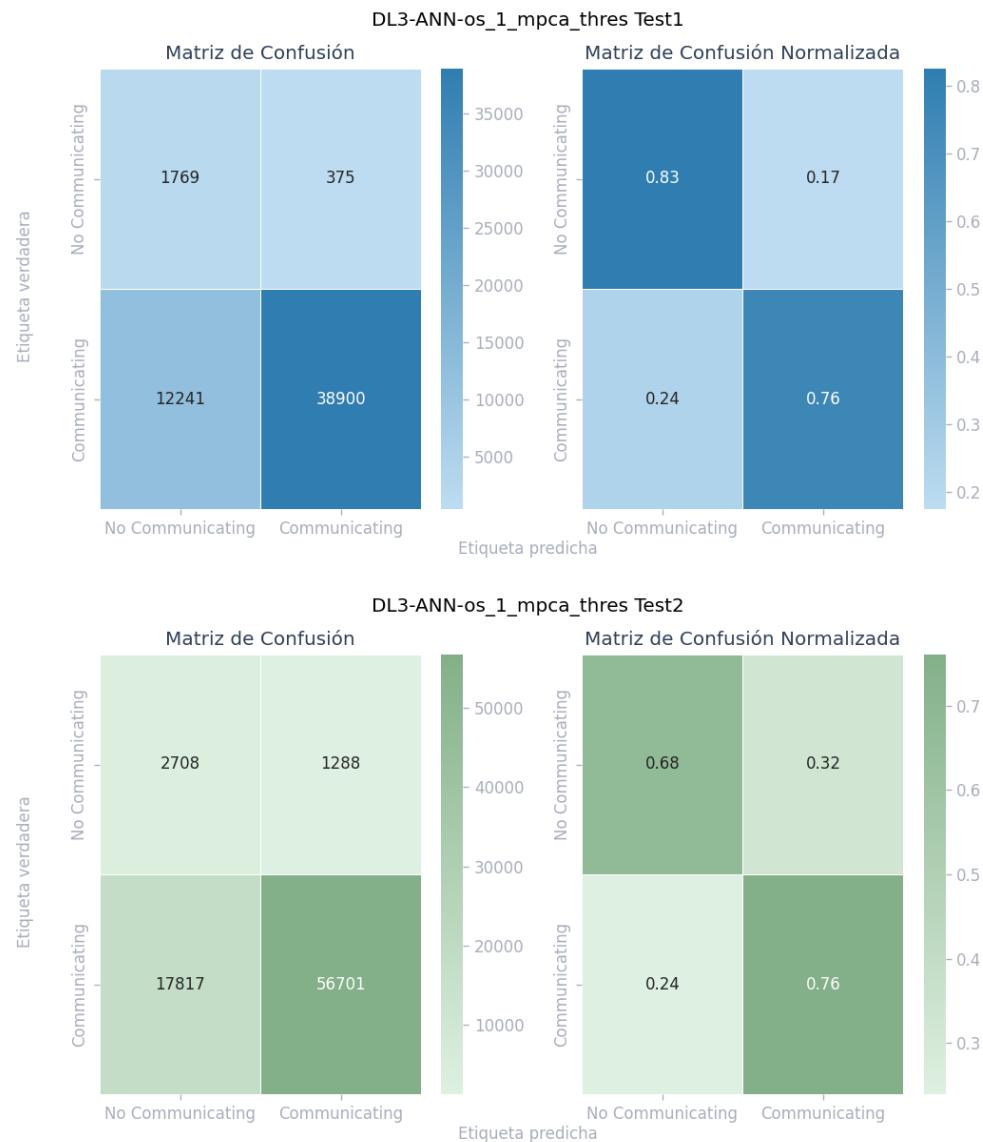
DL3 ANN Oversampled_1 Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 63.

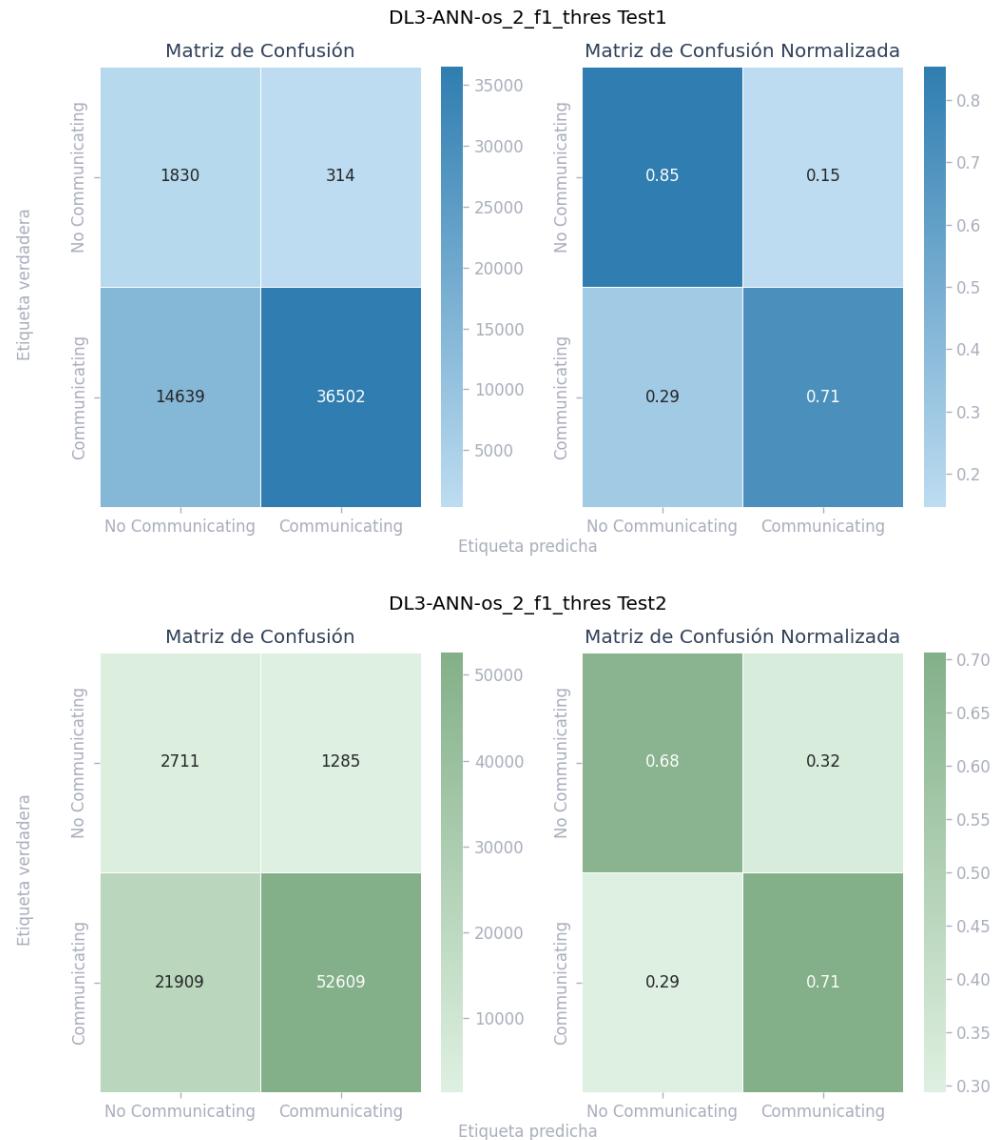
DL3 ANN Oversampled_1 Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 64.

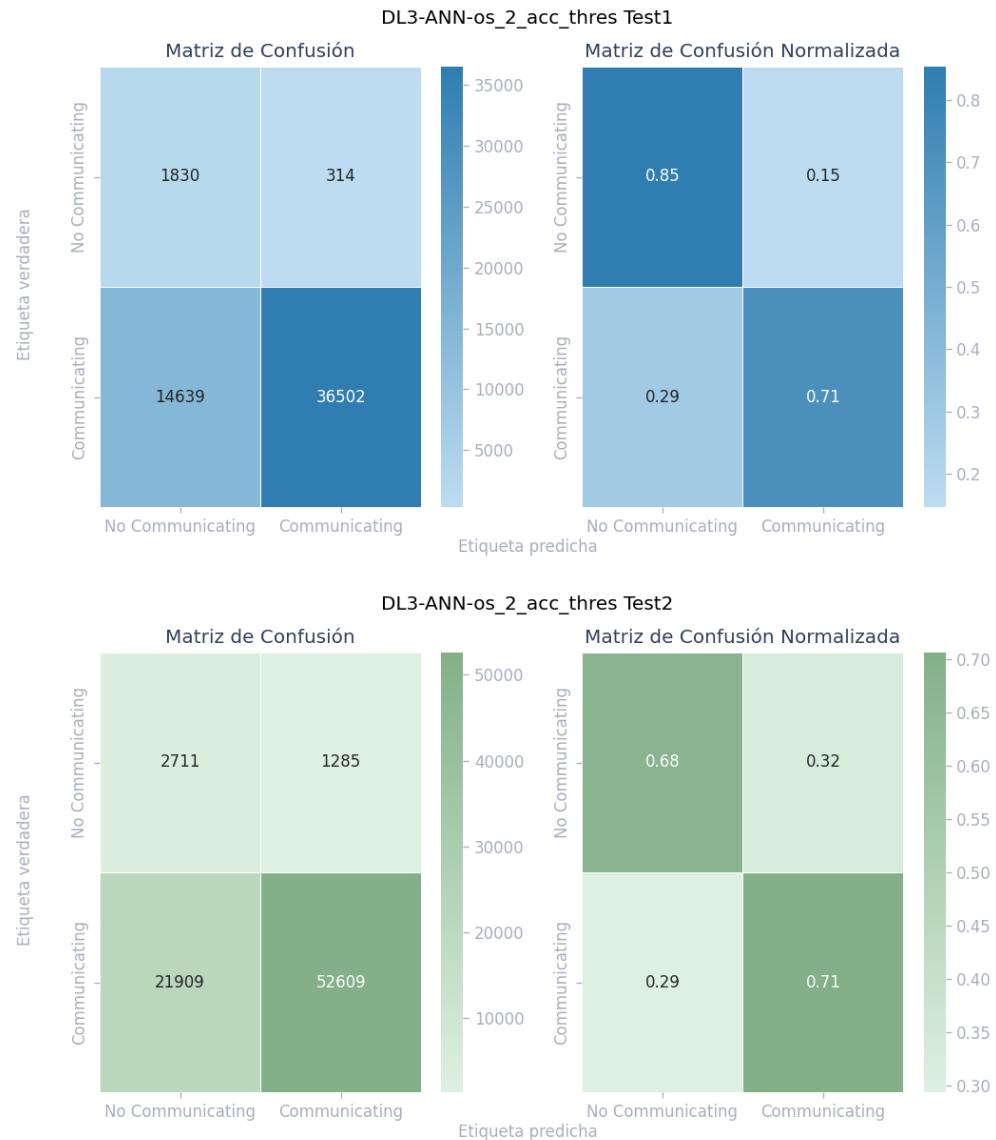
DL3 ANN Oversampled_2 Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 65.

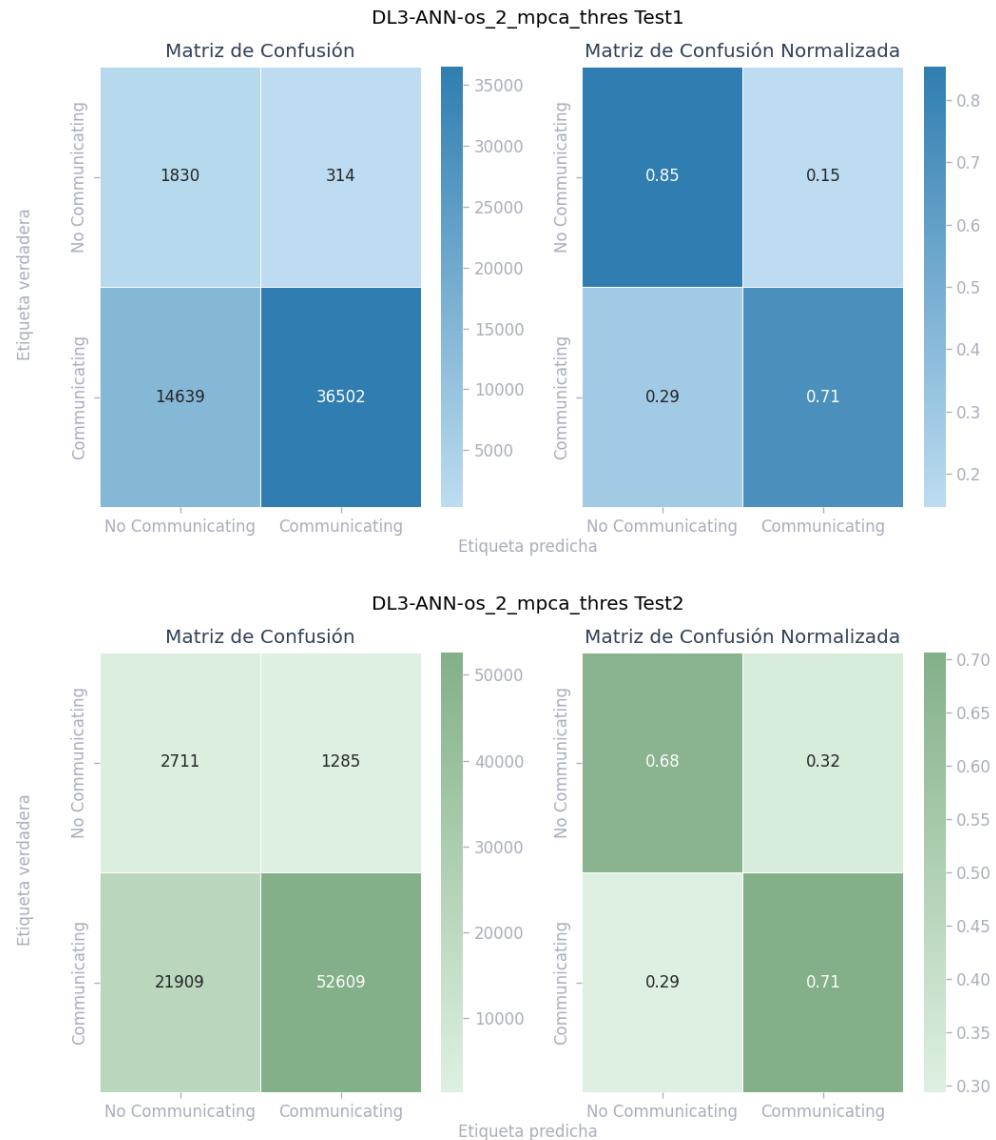
DL3 ANN Oversampled_2 Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 66.

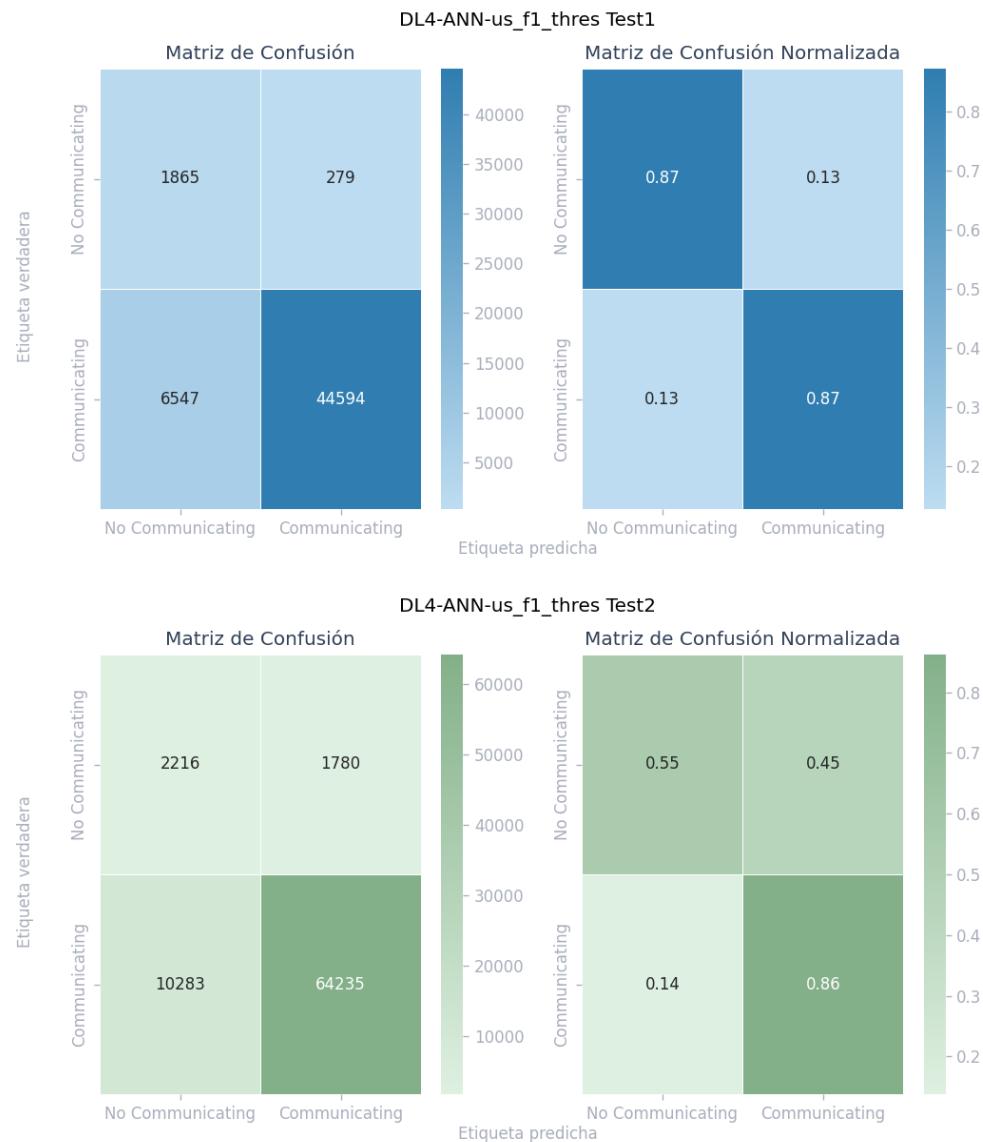
DL3 ANN Oversampled_2 Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 67.

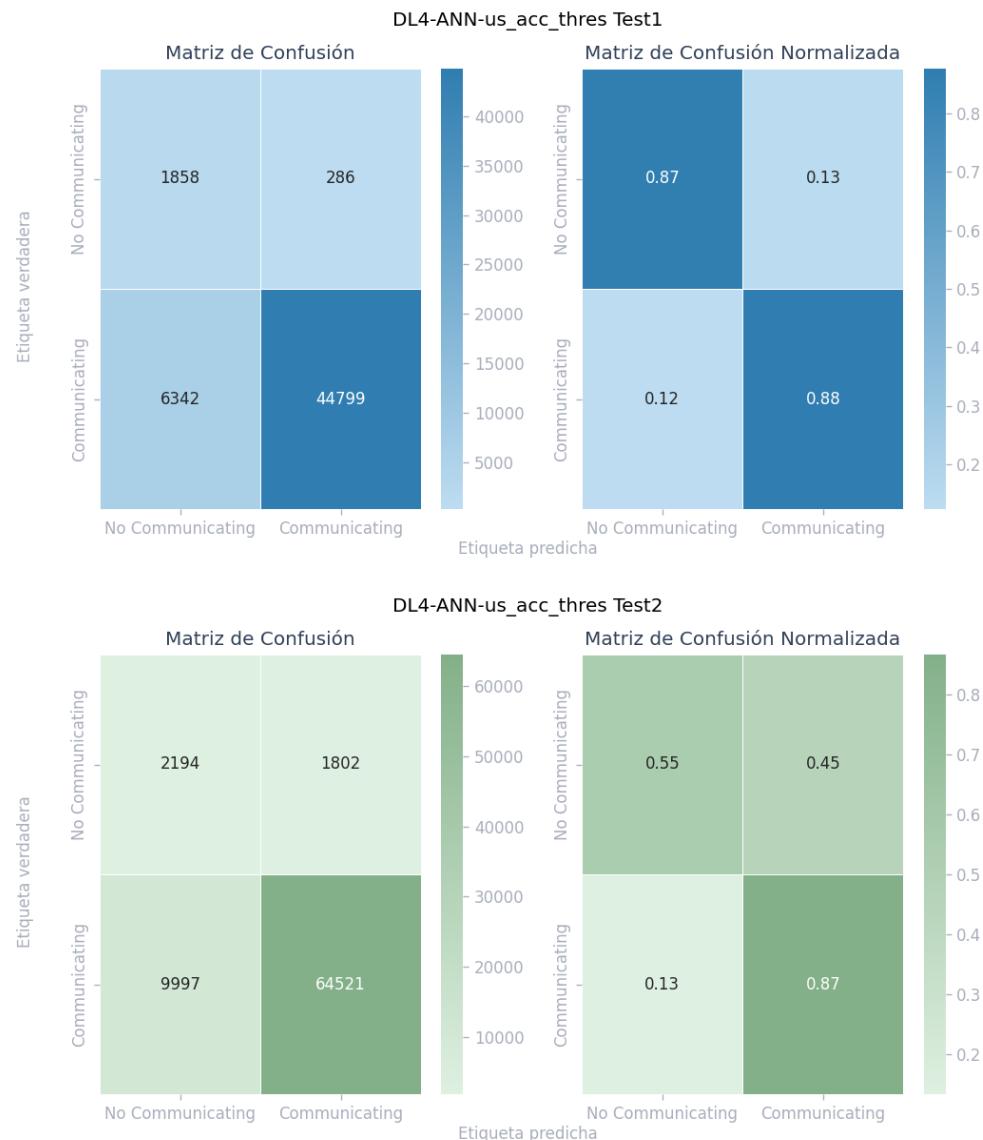
DL4 ANN Undersampled Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 68.

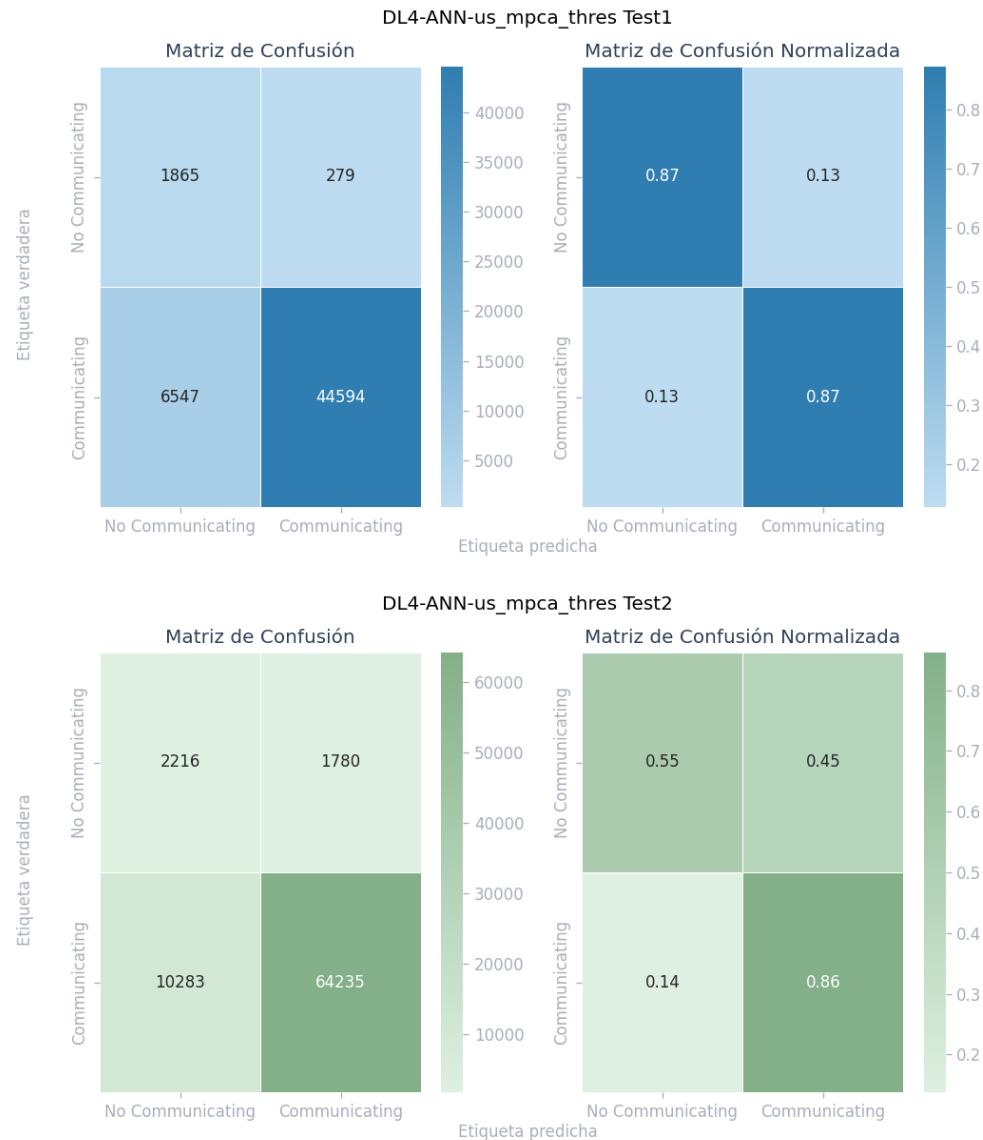
DL4 ANN Undersampled Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 69.

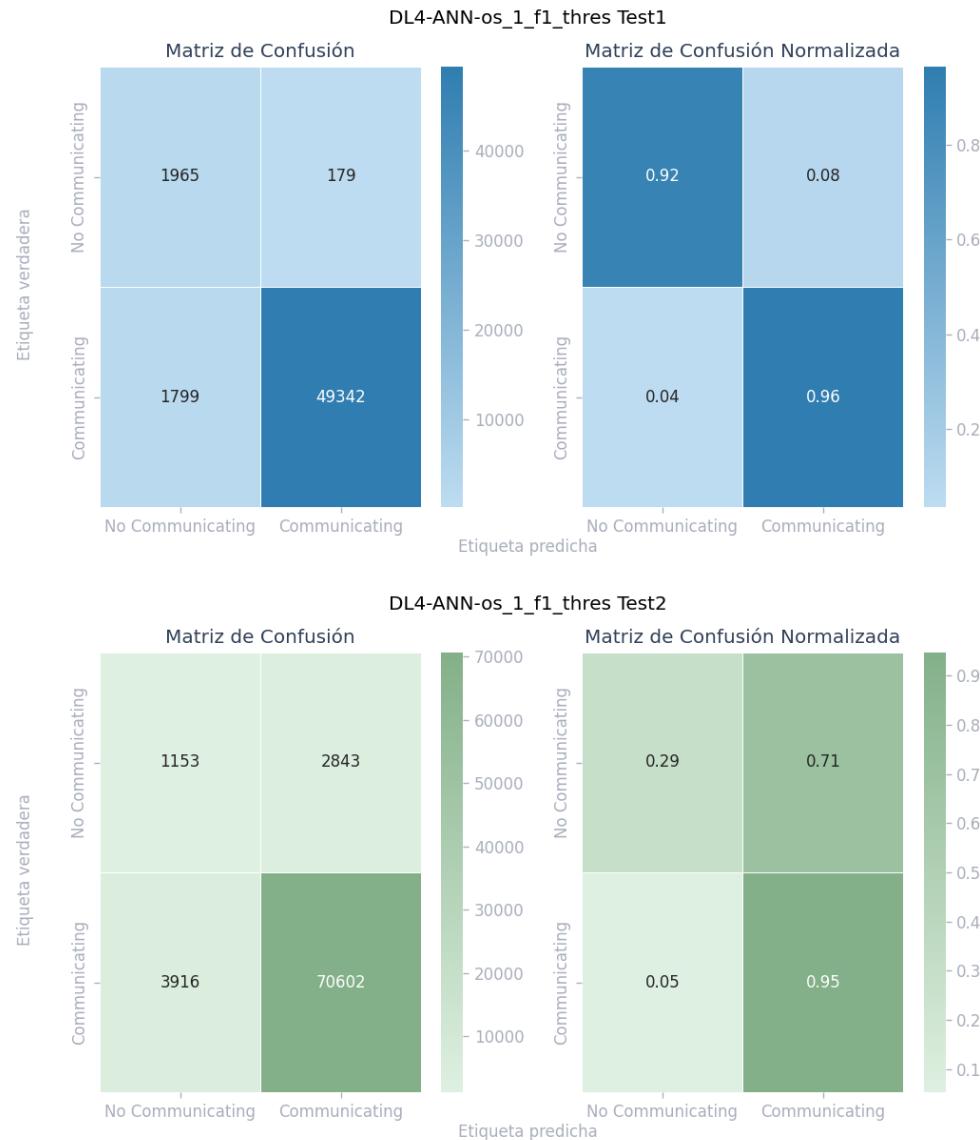
DL4 ANN Undersampled Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 70.

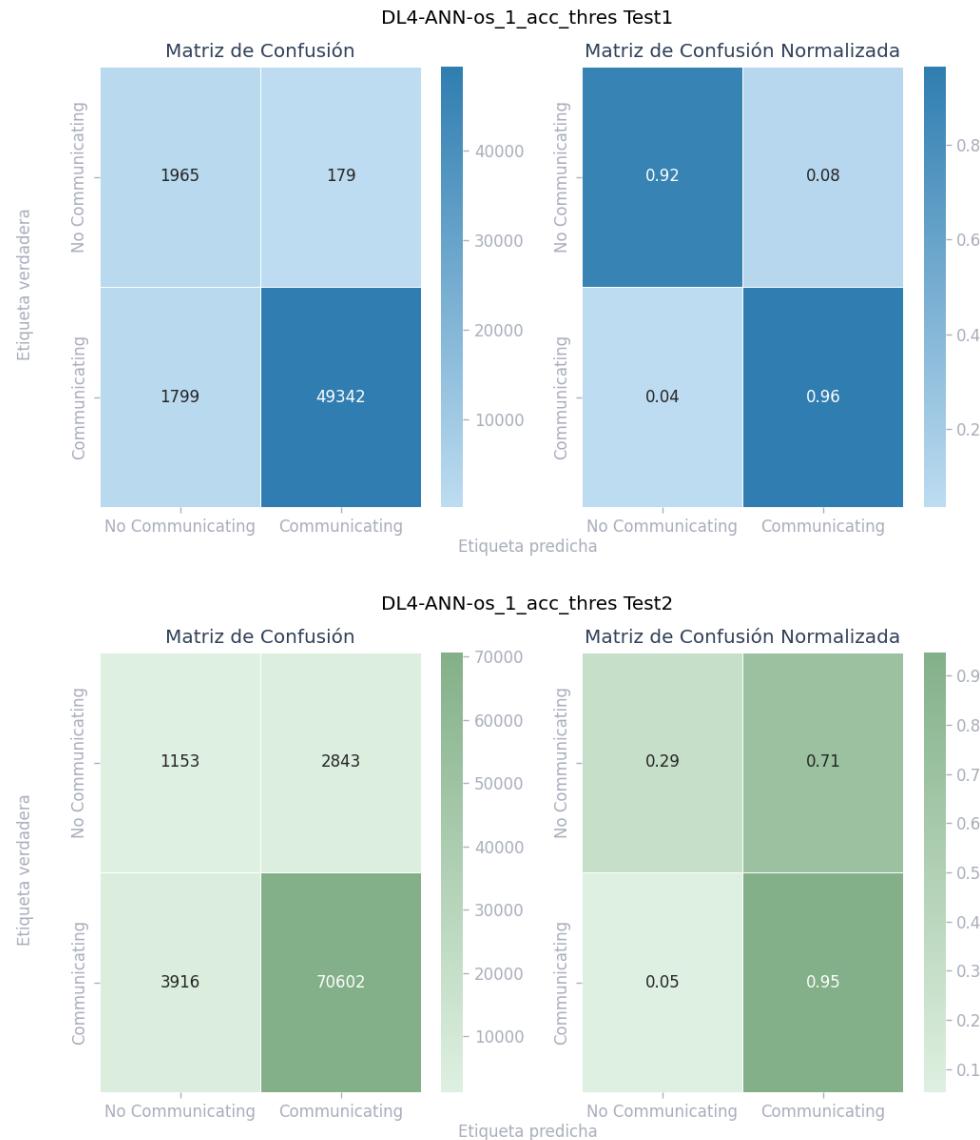
DL4 ANN Oversampled_1 Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 71.

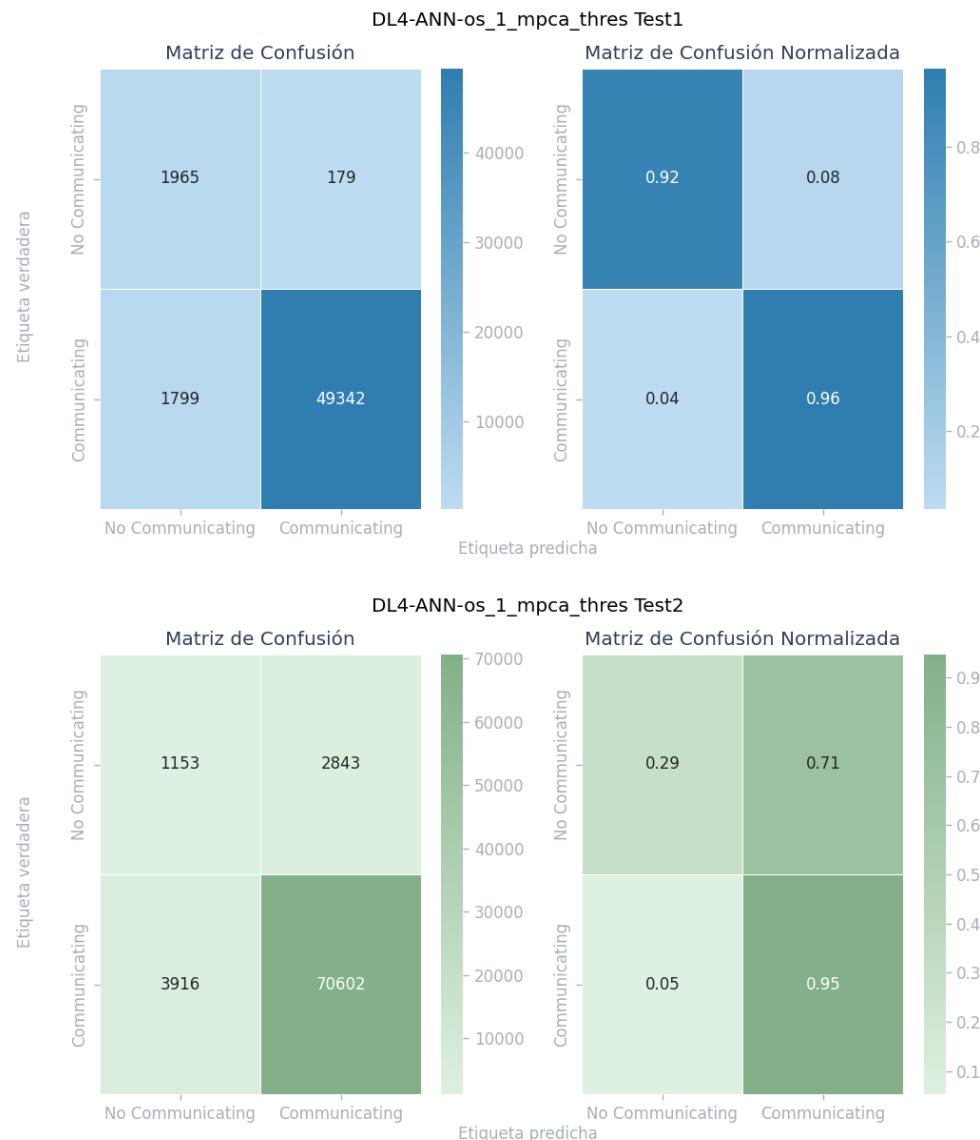
DL4 ANN Oversampled_1 Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 72.

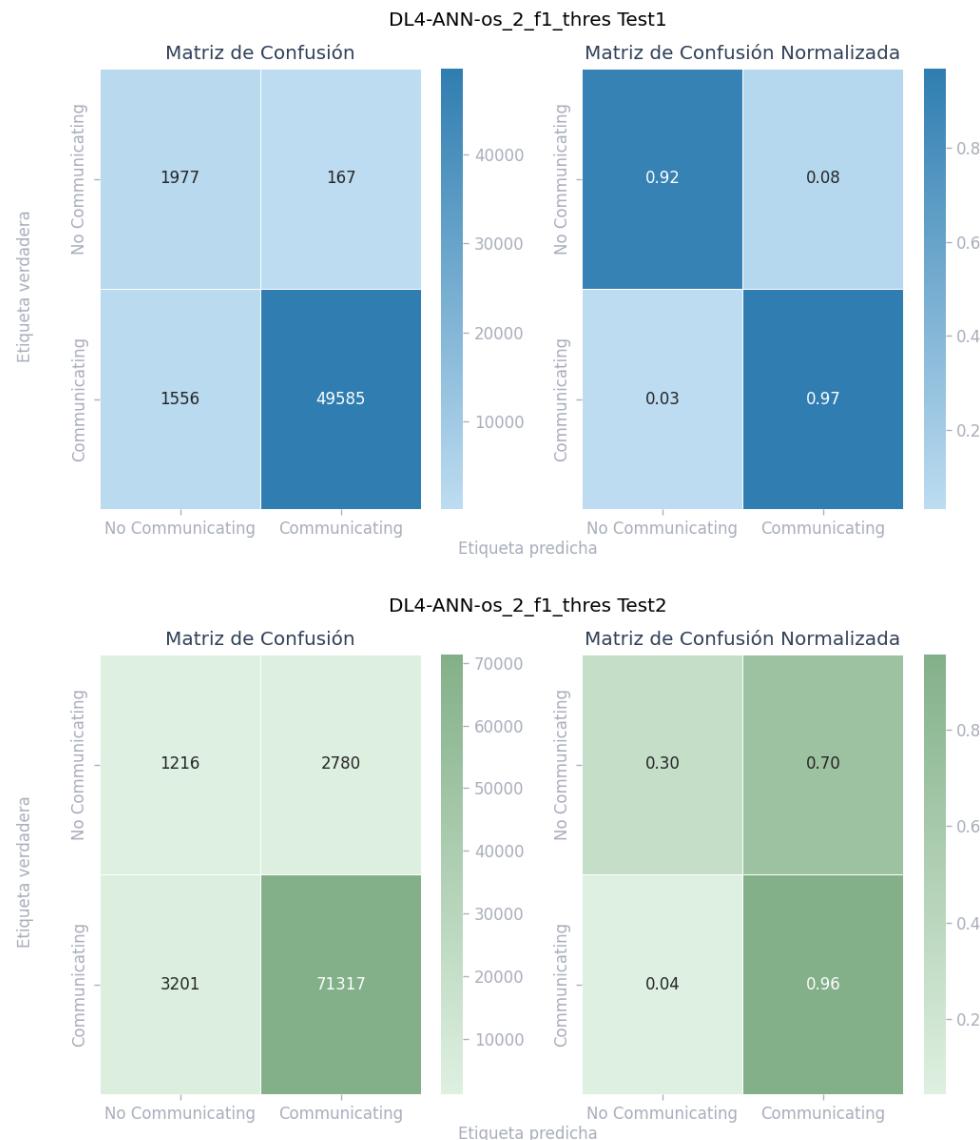
DL4 ANN Oversampled_1 Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 73.

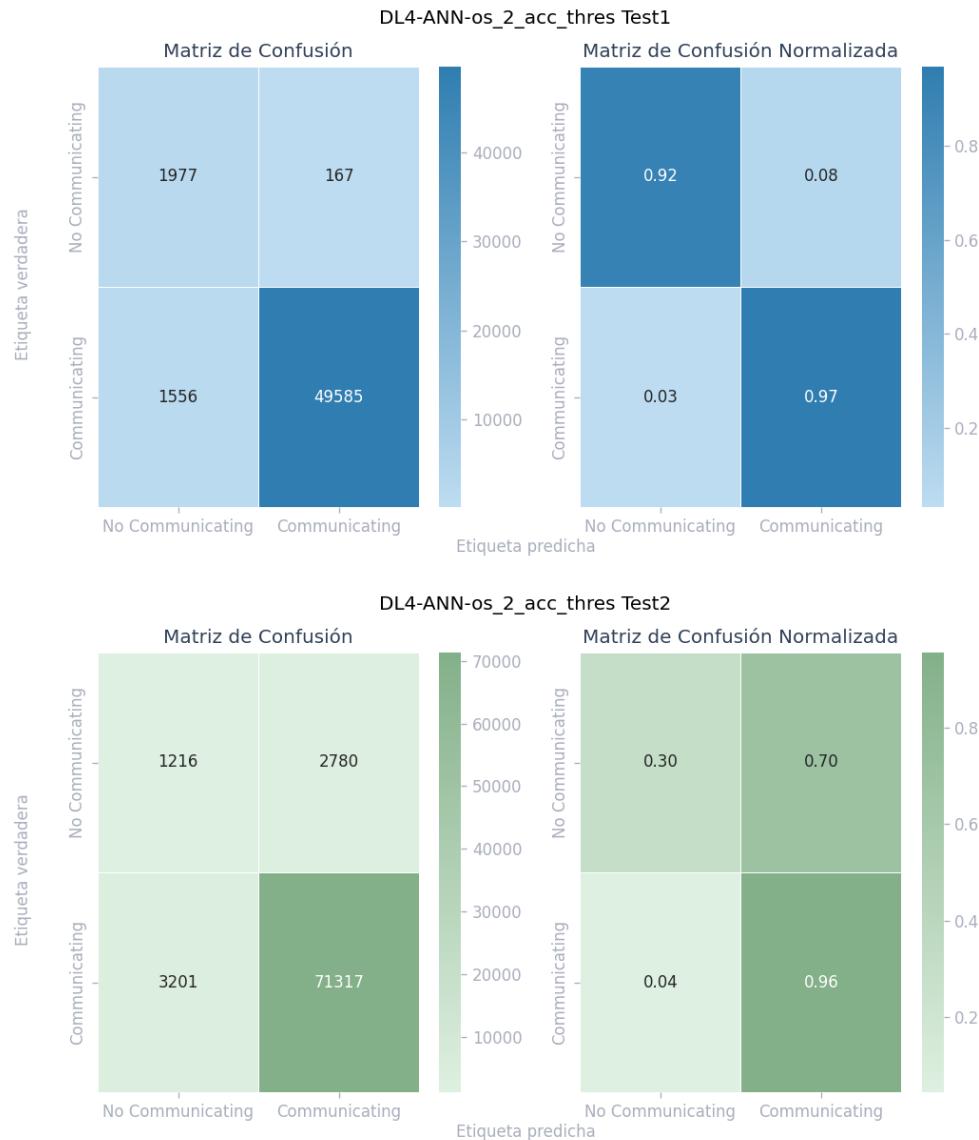
DL4 ANN Oversampled_2 Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 74.

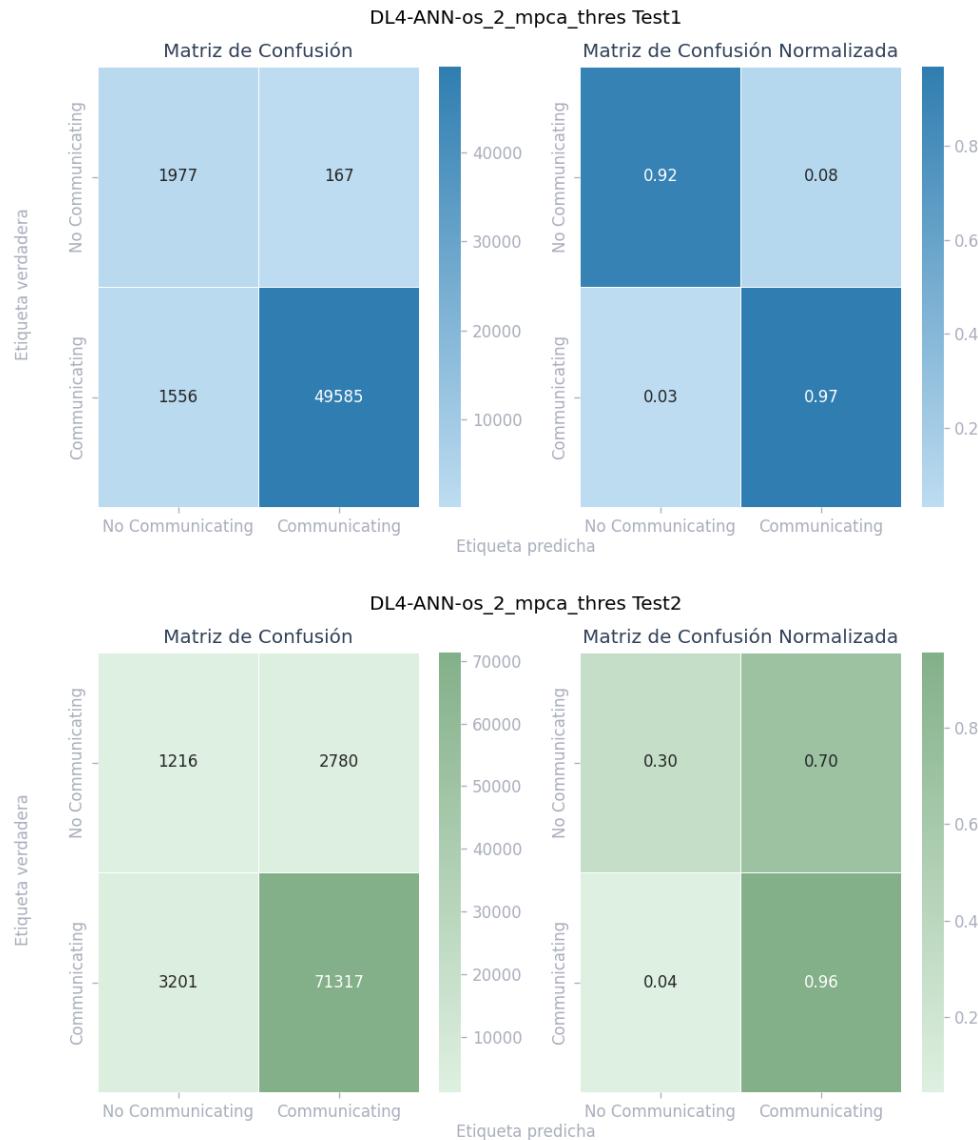
DL4 ANN Oversampled_2 Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 75.

DL4 ANN Oversampled_2 Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 76.

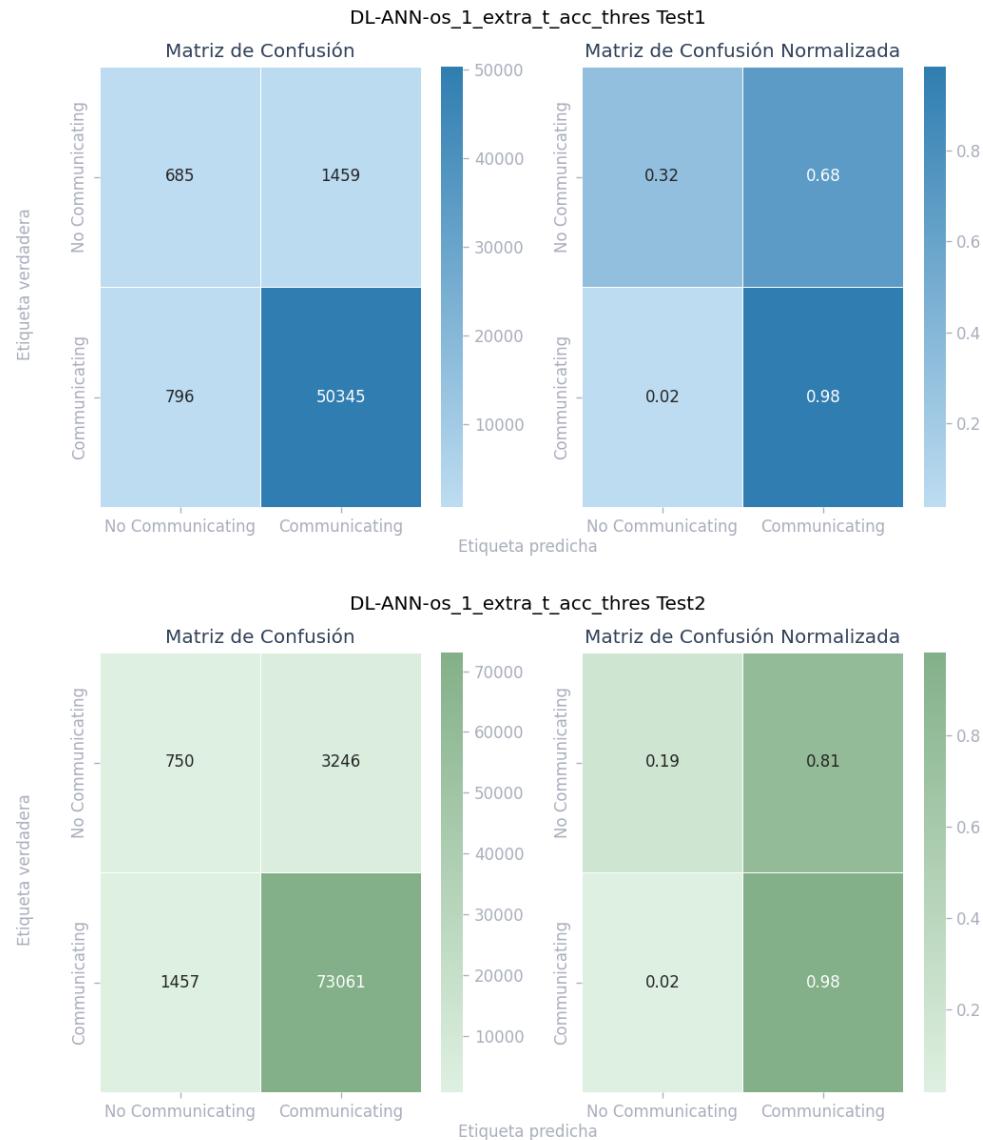
DL ANN Oversampled_1 Extra_T Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 77.

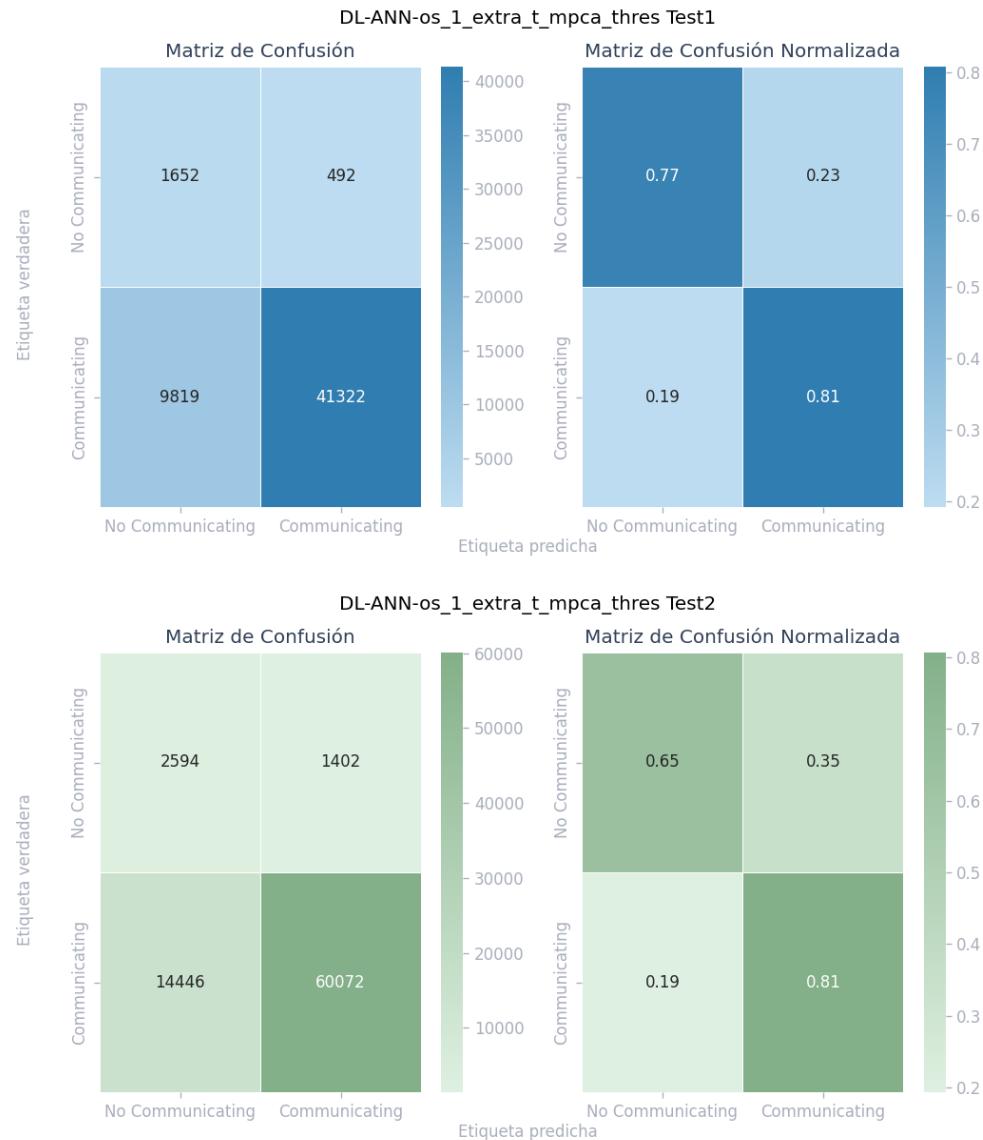
DL ANN Oversampled_1 Extra_T Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 78.

DL ANN Oversampled_1 Extra_T Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 79.

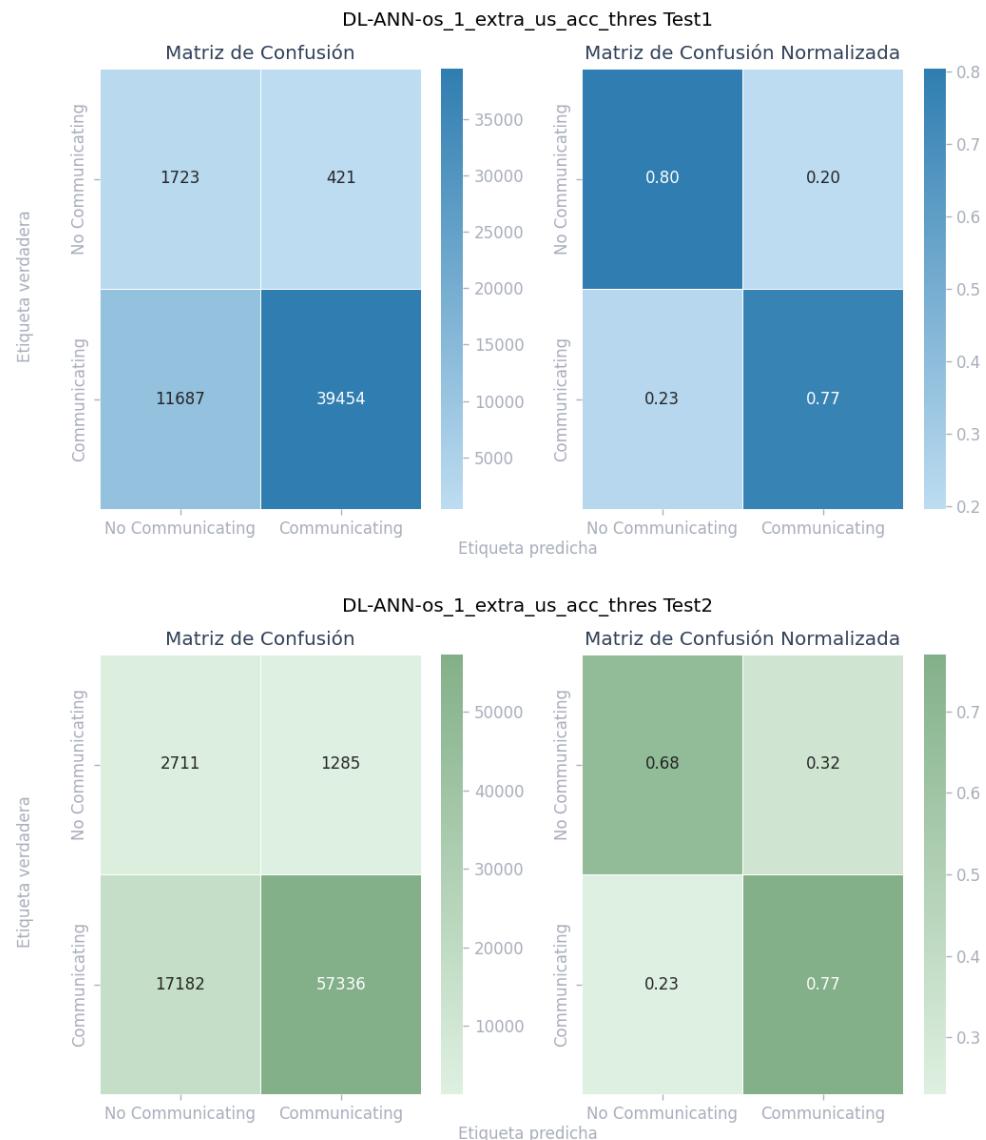
DL ANN Oversampled_1 Extra_US Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de Deep Learning. Elaboración propia, realizado con Python.

Apéndice 80.

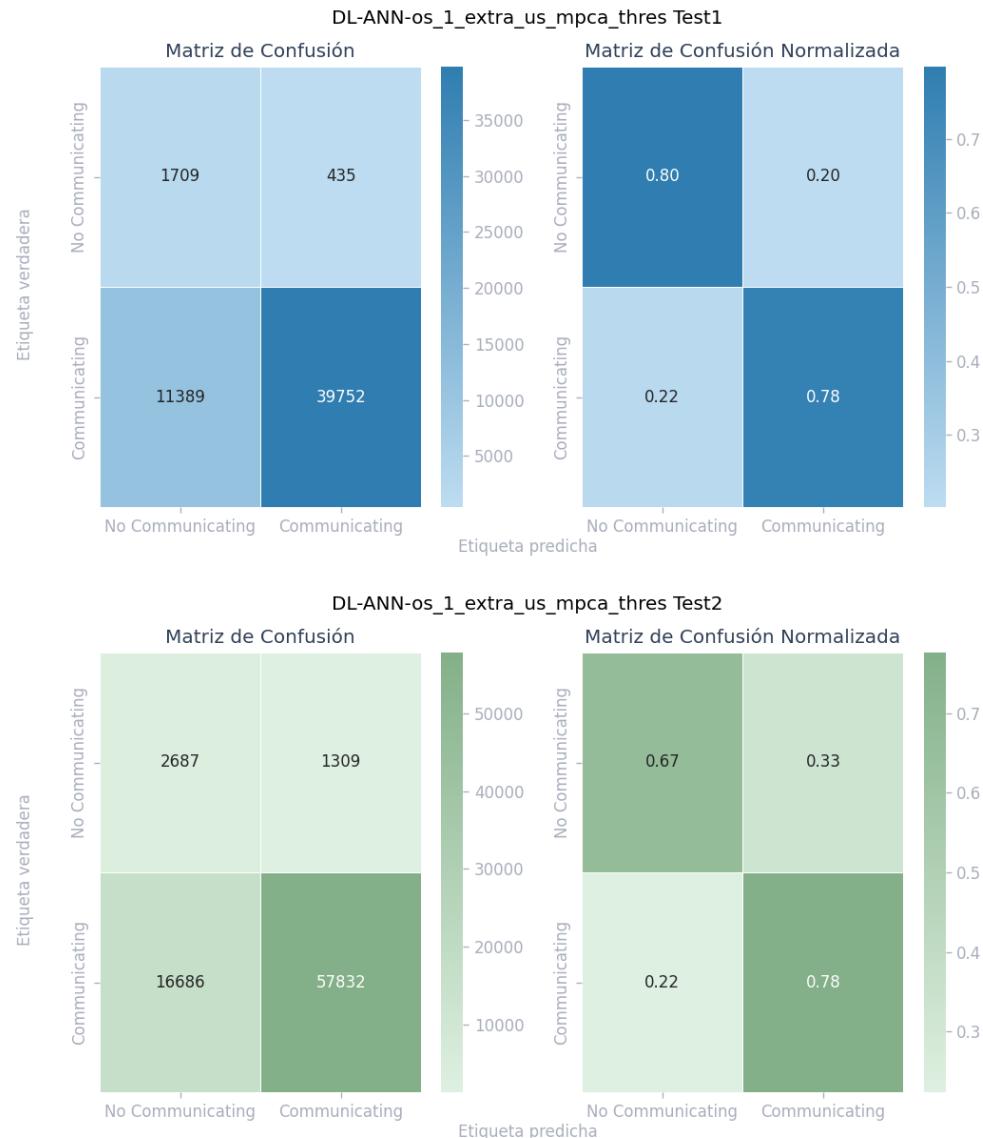
DL ANN Oversampled_1 Extra_US Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 81.

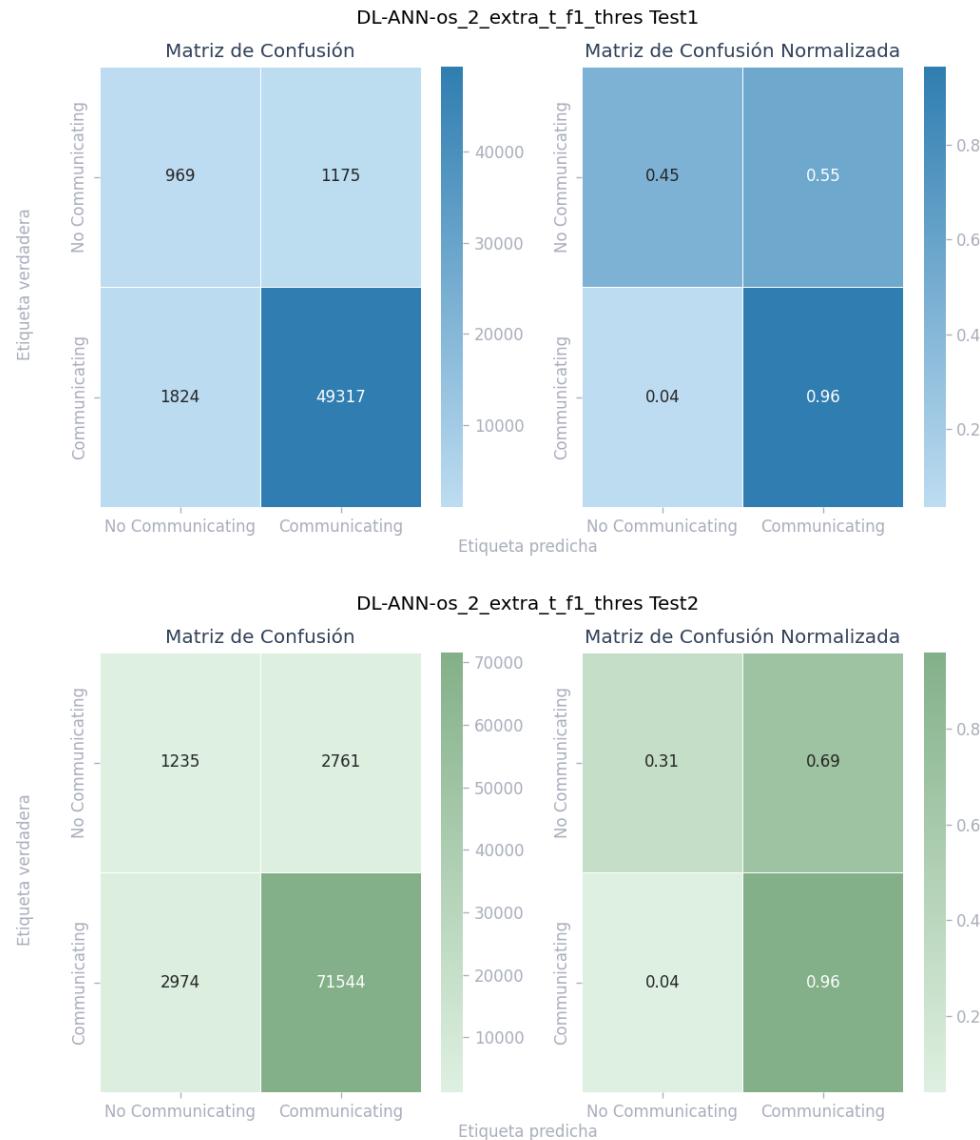
DL ANN Oversampled_1 Extra_US Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 82.

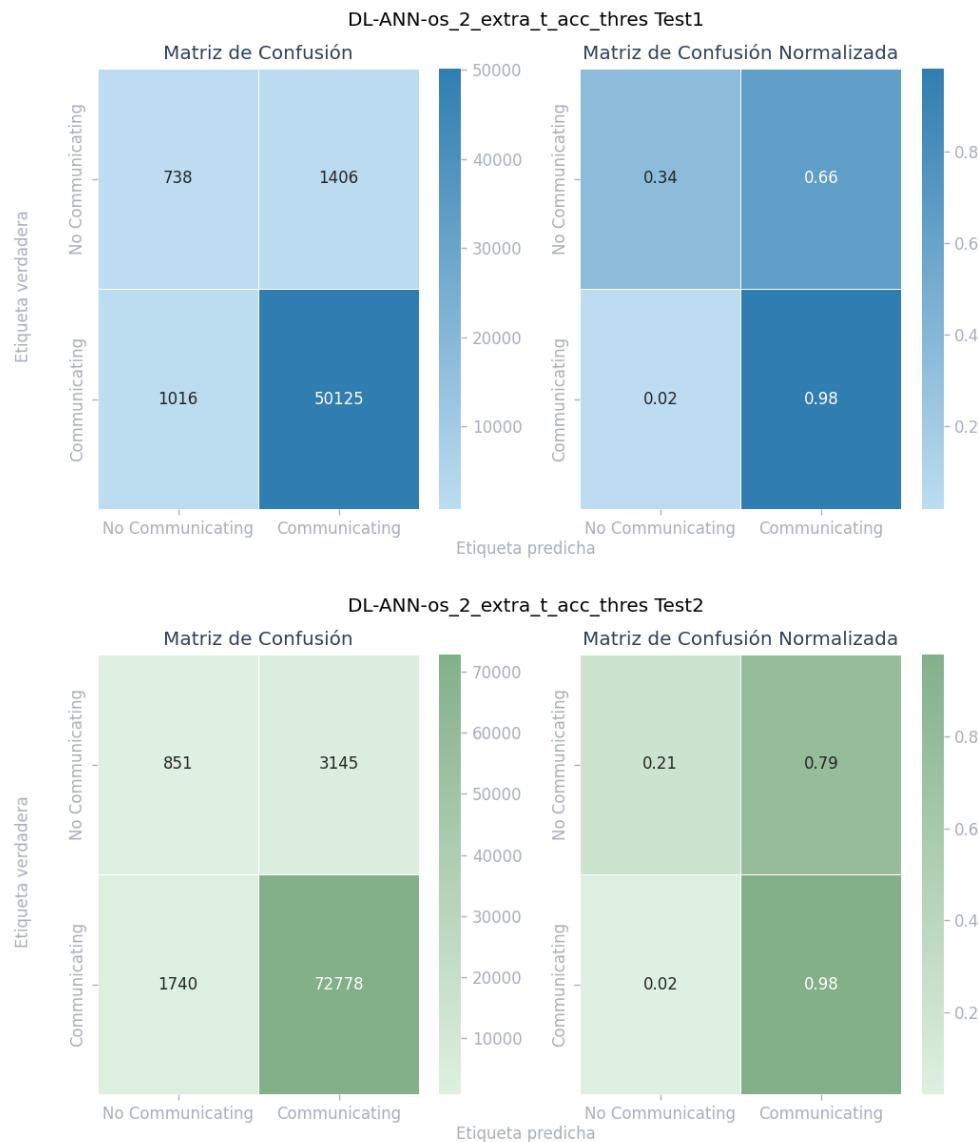
DL ANN Oversampled_2 Extra_T Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 83.

DL ANN Oversampled_2 Extra_T Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 84.

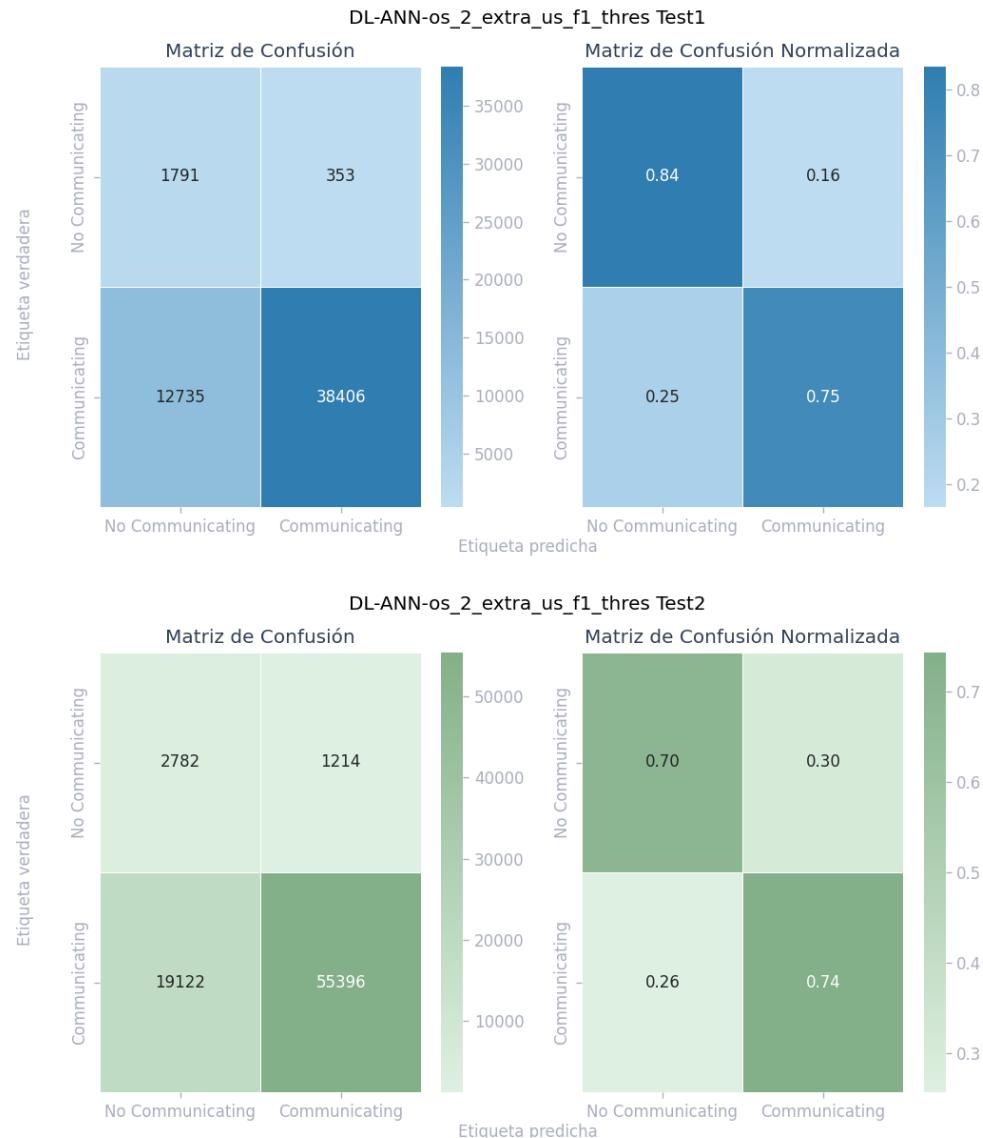
DL ANN Oversampled_2 Extra_T Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 85.

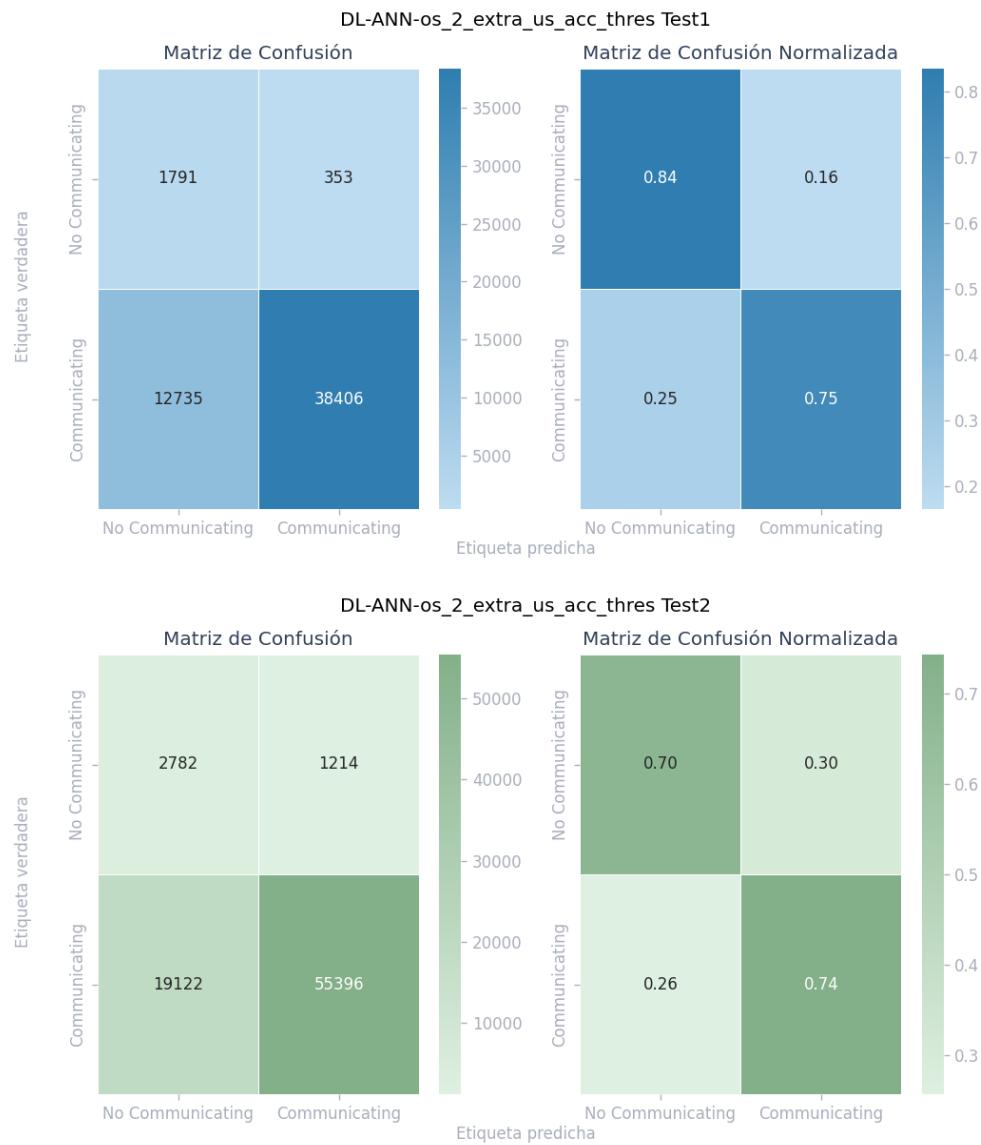
DL ANN Oversampled_2 Extra_US Threshold F1



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 86.

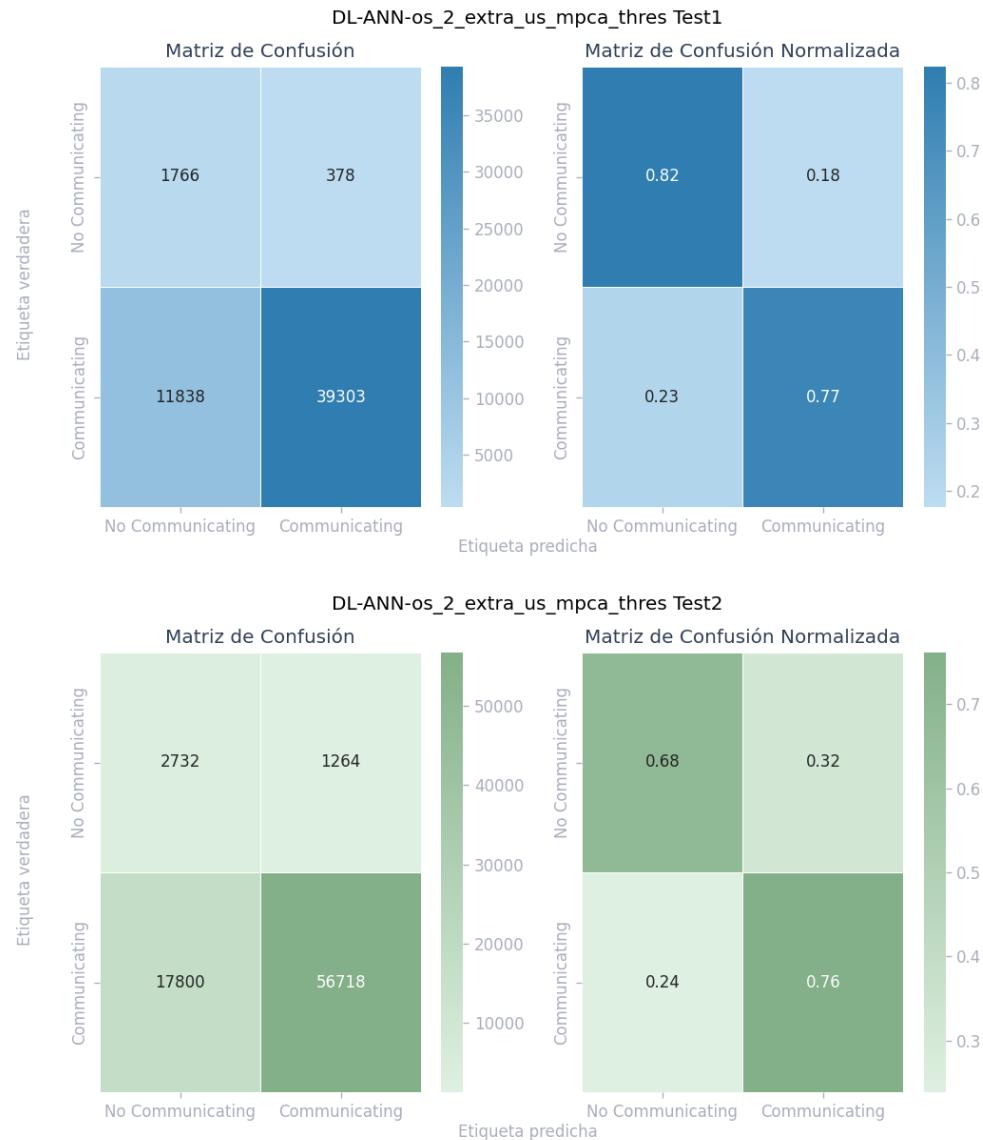
DL ANN Oversampled_2 Extra_US Threshold ACC



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 87.

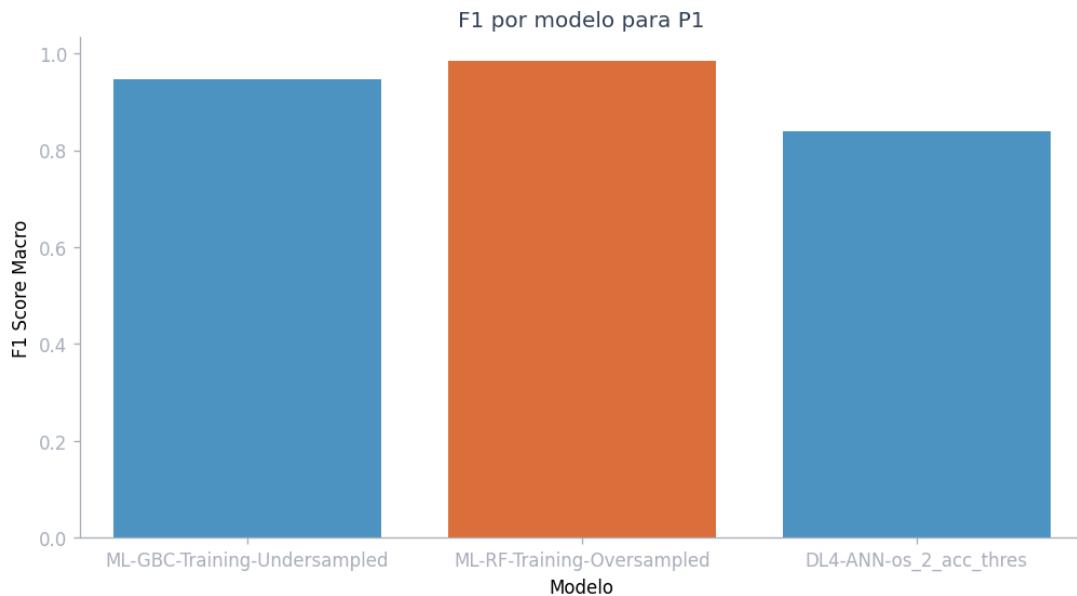
DL ANN Oversampled_2 Extra_US Threshold MPCA



Nota. La figura muestra la visualización de matrices de confusión que es parte de los resultados de los experimentos de *Deep Learning*. Elaboración propia, realizado con Python.

Apéndice 88.

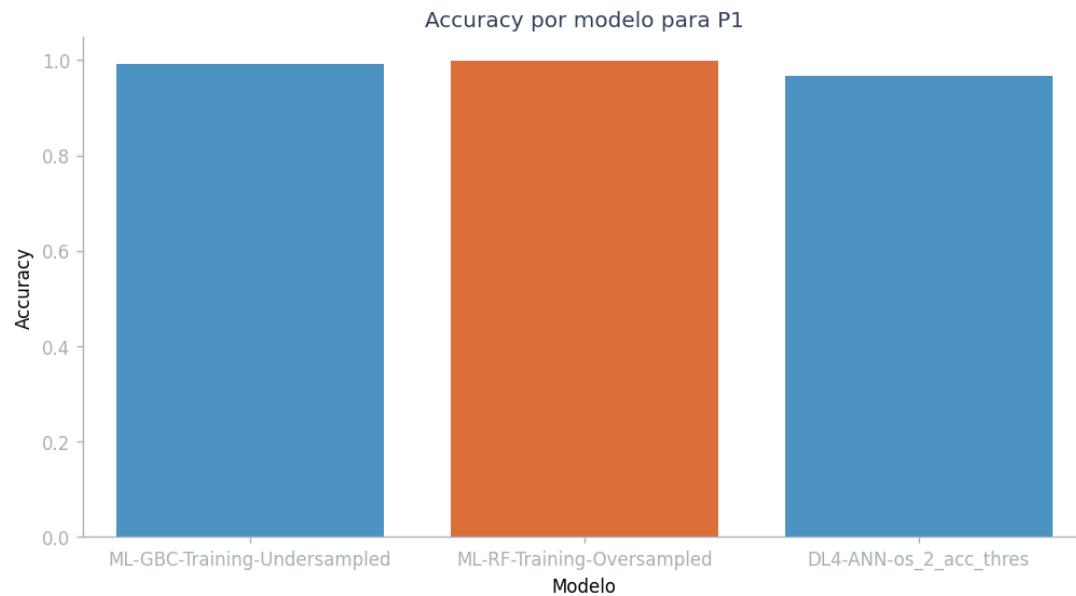
Comparación general $F1$ para P1



Nota. El gráfico muestra la comparación general según métrica que es parte de los resultados generales. Elaboración propia, realizado con Python.

Apéndice 89.

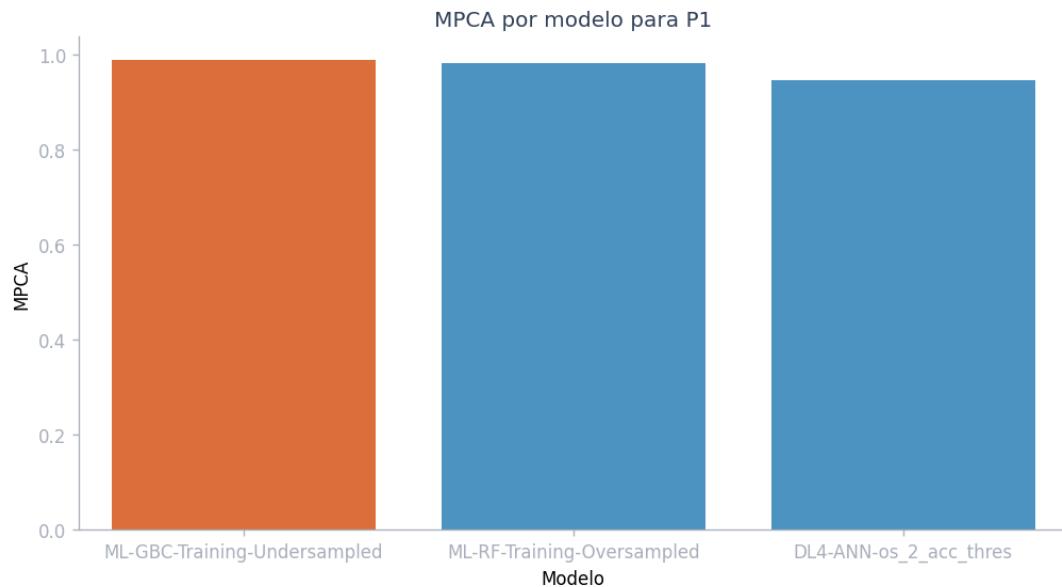
Comparación general ACC para P1



Nota. El gráfico muestra la comparación general según métrica que es parte de los resultados generales. Elaboración propia, realizado con Python.

Apéndice 90.

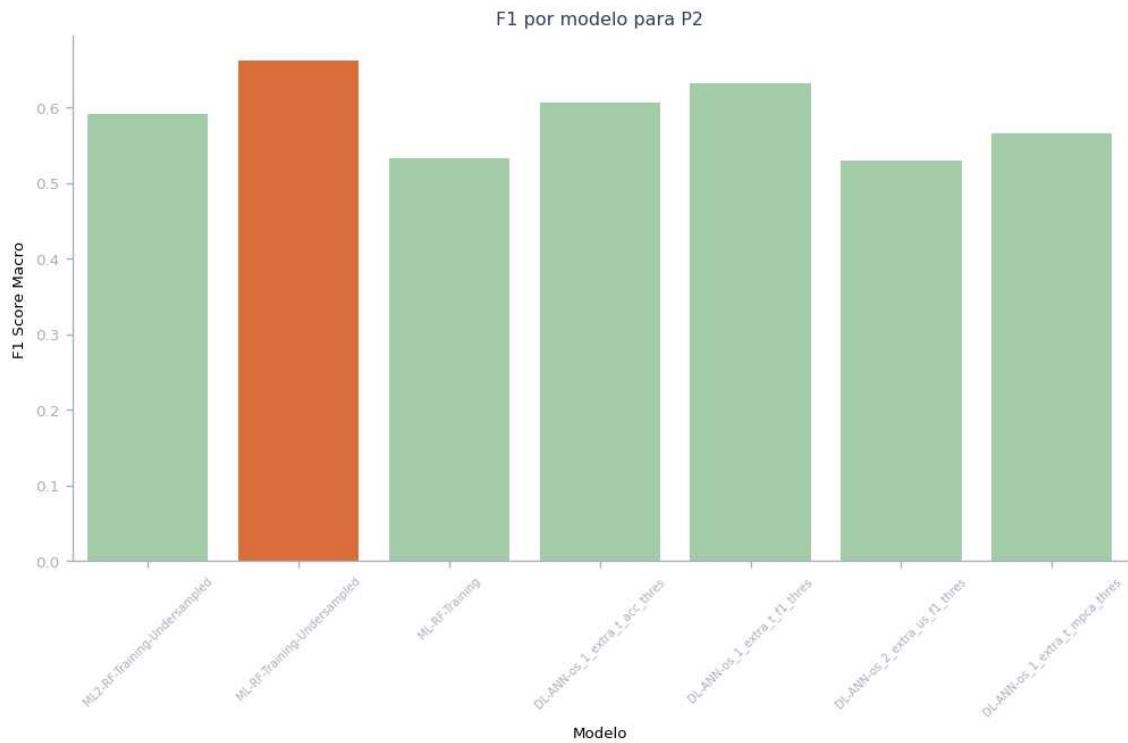
Comparación general MPCAs para P1



Nota. El gráfico muestra la comparación general según métrica que es parte de los resultados generales. Elaboración propia, realizado con Python.

Apéndice 91.

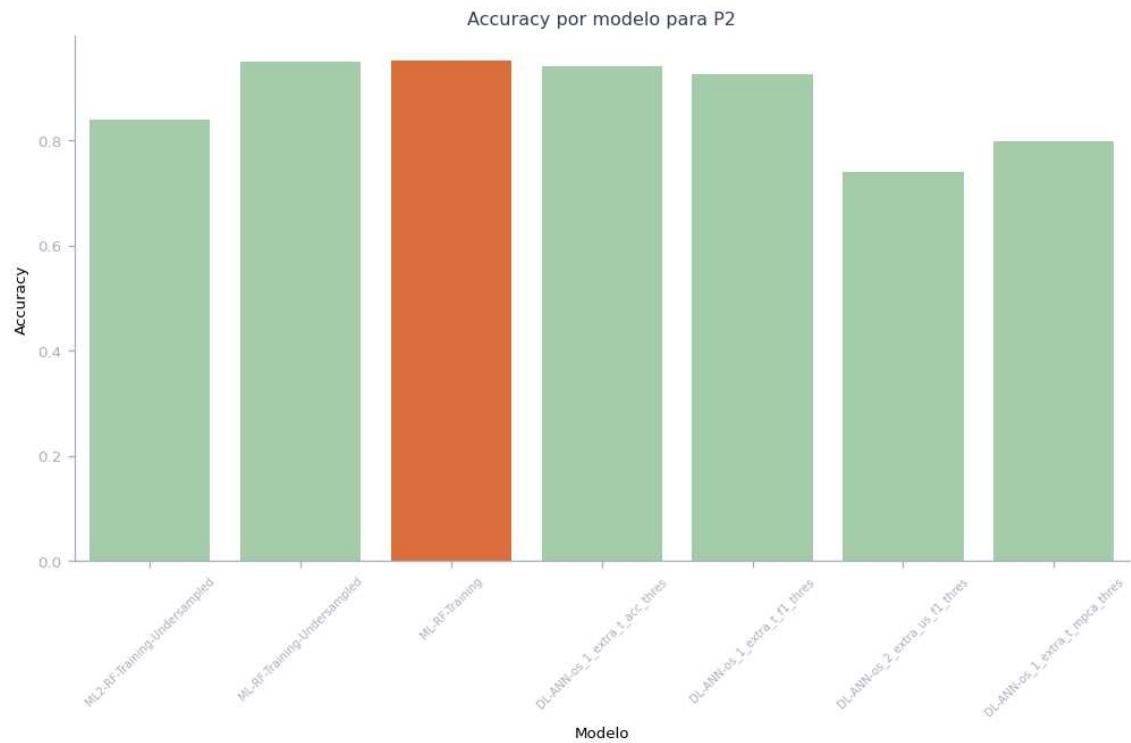
Comparación general *F1* para P2



Nota. El gráfico muestra la comparación general según métrica que es parte de los resultados generales. Elaboración propia, realizado con Python.

Apéndice 92.

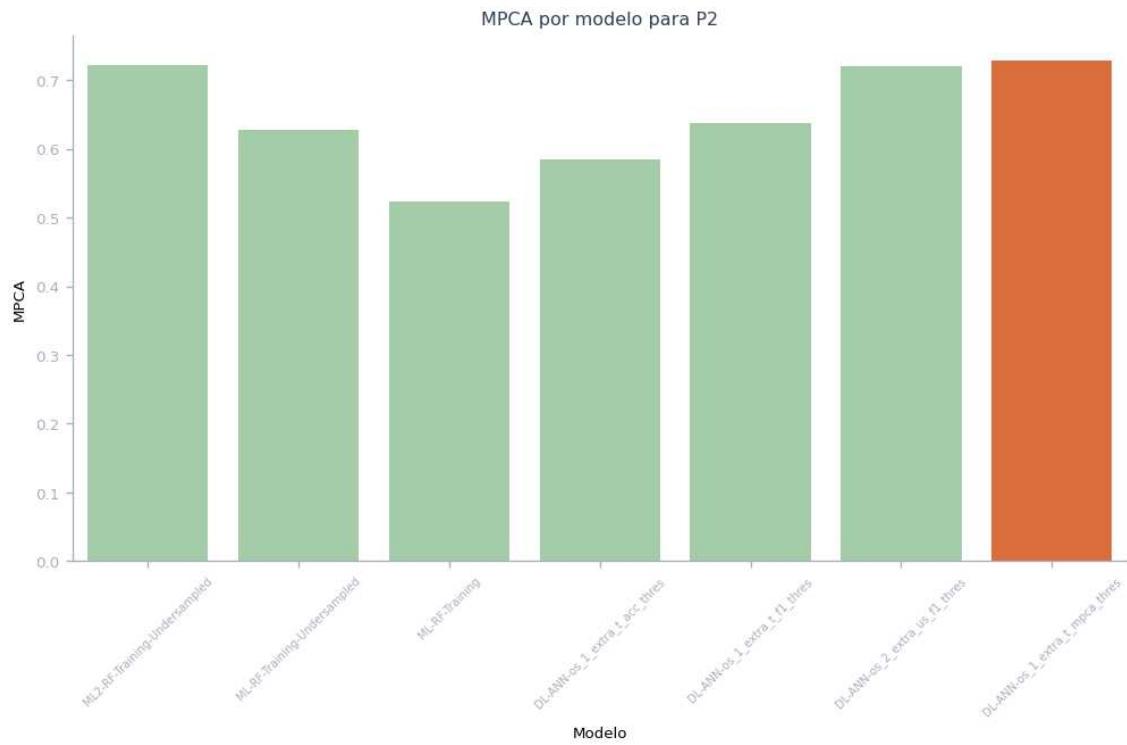
Comparación general ACC para P2



Nota. El gráfico muestra la comparación general según métrica que es parte de los resultados generales. Elaboración propia, realizado con Python.

Apéndice 93.

Comparación general MPCA para P2



Nota. El gráfico muestra la comparación general según métrica que es parte de los resultados generales. Elaboración propia, realizado con Python.

Apéndice 94.

Repository del proyecto

A continuación, se comparte el enlace del repositorio del proyecto, donde es posible ver el código que se ha validado, no tiene información sensible. Cualquier código mencionado en el trabajo escrito y que no esté en el repositorio, no pasó el filtro para poder ser publicado.

<https://github.com/VictorHugoBorrayo/GradPorject-WMN-A>

Nota. Enlace de repositorio del proyecto. Elaboración propia.