

Optimización de redes UMTS soportada en Machine Learning

Autor
Marco Aurelio Ordóñez Ordóñez

Tutor
Juan Carlos Gómez Paredes. Mag

Universidad Distrital Francisco José De Caldas
Maestría en Telecomunicaciones Móviles – Metodología Virtual
Énfasis en optimización de procesos
Bogotá, Colombia
julio de 2021

Dedicatoria

Quiero dedicar este logro primero que todo a DIOS, y por supuesto a mi familia quienes durante este periodo me han brindado todo su apoyo para poder conseguirlo. Al final este logro no es solo mío, sino también de mi esposa y de mis dos hijas que comprendieron el compromiso que involucraba emprender este proyecto. Sé sin lugar a dudas que este logro ahora dará sus frutos y los disfrutare con ustedes como recompensa por todo el tiempo que no pude compartir con ustedes.

Agradecimientos

Los más sinceros agradecimientos a la Universidad Distrital Francisco José de Caldas, ya que, entre sus diversas alternativas de formación en postgrados se incluyó la formación en Maestría de forma virtual. Lo anterior puesto que, muchas personas como yo que trabajamos en tiempo completo y que tenemos otras responsabilidades como padres de familia podemos hacer esto que nos gusta que es estudiar.

Por otro lado, les agradezco a todos los profesores que hicieron parte de este proceso de formación, y en especial a mi tutor el ingeniero magister Juan Carlos Gómez Paredes, quien con su experiencia en docencia y su energía motivacional impulsa a sus estudiantes a lograr la excelencia.

Contenido

INDICE DE FIGURAS v
INDICE DE TABLAS viii
Resumen 1
Palabras Clave 2
1. GENERALIDADES 3
1.1 INTRODUCCIÓN 3
1.2 PROBLEMA DE INVESTIGACIÓN 3
1.2.1 Planteamiento del problema 3
1.2.2 Formulación del problema 4
1.2.3 Sistematización del problema 4
1.3 OBJETIVOS 4
1.3.1 Objetivo general 4
1.3.2 Objetivos específicos 5
1.4 JUSTIFICACIÓN 5
2. marco teÓrico 6
2.1 REDES UMTS 6
2.1.1 KPIs redes UMTS 7
2.2 INTELIGENCIA ARTIFICIAL Y MACHINE LEARNING 12
2.2.1 Aprendizaje supervisado 13
2.2.2 Aprendizaje no supervisado 14
3. ANÁLISIS DE TÉCNICAS Y MODELOS DE MACHINE LEARNING 15
3.1 INTELIGENCIA ARTIFICIAL 15
3.2 INTELIGENCIA COMPUTACIONAL 16
3.3 MACHINE LEARNING 16
3.3.1 Propósito del Machine Learning 17
3.3.2 ¿Cuándo se pueden usar las técnicas de Machine Learning? 17
3.3.3 ¿Cómo funciona el Machine Learning? 18
3.4 TÉCNICAS DE MACHINE LEARNING 19
3.4.1 Aprendizaje No Supervisado 19
3.4.2 Aprendizaje Supervisado 24
3.4.3 Resumen técnicas de Machine Learning 38
4. DESARROLLO MODELOS DE MACHINE LEARNING 40

4.1 ANÁLISIS DATA SET CELDAS UMTS 40
4.2 TÉCNICA ALGORITMO ÁRBOL DE DECISIÓN....44
4.2.1 Desarrollo modelo 1 – Árbol de decisión.....44
4.2.2 Desarrollo modelo 2 – Árbol de decisión.....55
4.3 TÉCNICA RED NEURONAL 64
4.3.1 Desarrollo Modelo 1 – Red Neuronal 64
4.3.2 Desarrollo modelo 2 – Red Neuronal....75
5. anális resultados MODELOS DE MACHINE LEARNING 87
5.1 COMPARACIÓN DE LOS MODELOS DE CLASIFICACIÓN....87
5.1.1 Comparación modelo 1 87
5.1.2 Comparación modelo 2 91
5.2 ESQUEMA FINAL DIAGNÓSTICO CELDAS UMTS....94
5.3 TIEMPOS DE PROCESAMIENTO....95
5.3.1 Tiempo de entrenamiento – Técnica Árbol de Decisión 96
5.3.2 Tiempo de entrenamiento – Técnica Red Neuronal 97
6. Implementanción modelo final de MACHINE LEARNING 99
6.1 IMPLEMENTACIÓN ALGORITMO MODELO 1 99
6.2 IMPLEMENTACIÓN ALGORITMO MODELO 2 102
7. CONCLUSIONES 106
8. TRABJOS FUTUROS 107
9. REFERENCIAS....108

INDICE DE FIGURAS

Figura No. 2-1. Arquitectura 3G-UMTS. Fuente [4] 6
Figura No. 2-2. Esquema comunicación WCDMA UMTS. Fuente propia. 7
Figura No. 2-3. Consumo de potencia celdas UMTS. Fuente propia. 8
Figura No. 2-4. Códigos celdas UMTS. Fuente propia. 9
Figura No. 2-5. Throughput VS número de usuarios. Fuente [7]. 11
Figura No. 2-6. TTI load celda UMTS. Fuente propia. 11
Figura No. 2-7. Técnicas de aprendizaje autónomo. Fuente [13]. 13
Figura No. 3-1. Concepto Inteligencia Artificial. Fuente propia. 15
Figura No. 3-2. Esquema de clasificación IA, IC y ML. Fuente propia. Fuente propia. 16
Figura No. 3-3. Aprendizaje Automático Machine Learning. Fuente propia. 17
Figura No. 3-4. Variables de celda UMTS. Fuente propia. 18
Figura No. 3-5. Técnicas de aprendizaje automático. Fuente propia. 19
Figura No. 3-6. Técnica de Agrupación o Clustering. Fuente propia. 20
Figura No. 3-7. Reducción dimensiones 3D a 2D y 1D. Fuente propia. 21
Figura No. 3-8. Clase selección de características. Fuente propia. 22
Figura No. 3-9. Clase extracción de características. Fuente propia. 23
Figura No. 3-10. Ejemplos de Reglas de Asociación. Fuente propia. 24
Figura No. 3-11. Regresión lineal simple – 2D. Fuente propia. 26
Figura No. 3-12. Regresión lineal múltiple - 3D. Fuente propia. 26
Figura No. 3-13. Clasificación lineal para mensajes de correo electrónico. Fuente [17]. 28
Figura No. 3-14. Clasificación con máquina de soporte vectorial. Fuente propia. 29
Figura No. 3-15. Clasificación k-NN con k=3. Fuente propia. 31
Figura No. 3-16. Árbol de decisión para clasificación. Fuente propia. 33
Figura No. 3-17. Partes de la Neurona. Fuente [18] 35
Figura No. 3-18. Modelo neuronal de McCulloch-Pitts. Fuente [19] 36
Figura No. 3-19. Interconexión entre neuronas artificiales. Fuente propia. 37
Figura No. 3-20. Red neuronal multi-capa. Fuente propia. 38
Figura No. 4-1. Clasificación total de los registros del dataset. Fuente propia. 42
Figura No. 4-2. Clasificación de registros Modelo 1. Fuente propia. 43
Figura No. 4-3. Clasificación de registros Modelo 2. Fuente propia. 43
Figura No. 4-4. Modelos de Clasificación. Fuente propia. 44
Figura No. 4-5. APP Classification Learner. Fuente propia. 44
Figura No. 4-6. Importe Data Set Modelo 1. Fuente propia. 45
Figura No. 4-7. Configuración APP Classification Learner – Modelo 1. Fuente propia. 46
Figura No. 4-8. Opciones árboles de decisión. Fuente propia. 46
Figura No. 4-9. Entrenamiento “Fine Tree” – Modelo 1. Fuente propia. 47
Figura No. 4-10. Desviación estándar – Variables de entrada Modelo 1. Fuente propia. 48
Figura No. 4-11. AUC y curvas ROC Clase OK. Fuente propia. 49
Figura No. 4-12. AUC y curvas ROC Clase NOK. Fuente propia. 49
Figura No. 4-13. Matriz de Confusión (Número de observaciones)– Modelo 1. Fuente propia. 52

Figura No. 4-14. Matriz de Confusión (TPR/FNR)– Sensibilidad Modelo 1. Fuente propia. 53
Figura No. 4-15. Matriz de Confusión (PPV/FDR) – Precisión Modelo 1. Fuente propia. 54
Figura No. 4-16. Árbol de decisión Modelo 1. Fuente propia. 54
Figura No. 4-17. Importe Data Set Modelo 2. Fuente propia. 55
Figura No. 4-18. Configuración APP Classification Learner – Modelo 2. Fuente propia. 56
Figura No. 4-19. Entrenamiento "Fine Tree" – Modelo 2. Fuente propia. 56
Figura No. 4-20. Desviación estándar – Variables de entrada Modelo 2. Fuente propia. 57
Figura No. 4-21. AUC y curvas ROC Clase CAP. Fuente propia. 58
Figura No. 4-22. AUC y curvas ROC Clase COV. Fuente propia. 58
Figura No. 4-23. AUC y curvas ROC Clase CHECK. Fuente propia. 59
Figura No. 4-24. Matrices Multiclase CAP, CHECK y COV. Árbol de decisión Modelo 2. Fuente propia. 60
Figura No. 4-25. Matriz de Confusión (Número de observaciones)– Modelo 2. Fuente propia. 60
Figura No. 4-26. Matriz de Confusión (TPR/FNR)– Sensibilidad Modelo 2. Fuente propia. 61
Figura No. 4-27. Matriz de Confusión (PPV/FDR) – Precisión Modelo 2. Fuente propia. 62
Figura No. 4-28. Árbol de decisión Modelo 2. Fuente propia. 63
Figura No. 4-29. APP Neuronal Net Pattern Recognition. Fuente propia. 64
Figura No. 4-30. Variables de entrada - Red Neuronal Modelo 1. Fuente propia. 65
Figura No. 4-31. Variable de salida - Red Neuronal Modelo 1. Fuente propia. 66
Figura No. 4-32. Configuración APP Neural Pattern Recognition – Modelo 1. Fuente propia. 66
Figura No. 4-33. Asignación datos de entrenamiento, validación y prueba Modelo 1 67
Figura No. 4-34. Arquitectura Red Neuronal – Modelo 1. Fuente propia. 68
Figura No. 4-35. Arquitectura tradicional Red Neuronal – Modelo 1. Fuente propia. 69
Figura No. 4-36. Resultados proceso de entrenamiento Red Neuronal – Modelo 1. Fuente propia. 70
Figura No. 4-37. Entropía cruzada fases de entrenamiento Modelo 1. Fuente propia. 71
Figura No. 4-38. Evolución gradiente y controles de validación – Red Neuronal Modelo 1. Fuente propia. 72
Figura No. 4-39. Histograma – Error en la predicción –Red Neuronal Modelo1. Fuente propia. 72
Figura No. 4-40. Curvas ROC y AUC Red Neuronal Modelo 1. Fuente propia. 73
Figura No. 4-41. Matrices de confusión fases Red Neuronal Modelo 1. Fuente propia. 74
Figura No. 4-42. Variables de entrada - Red Neuronal Modelo 2. Fuente propia. 76
Figura No. 4-43. Variable de salida - Red Neuronal Modelo 2. Fuente propia. 77
Figura No. 4-44. Configuración APP Neural Pattern Recognition – Modelo 2. Fuente propia. 77
Figura No. 4-45. Asignación datos de entrenamiento, validación y prueba Modelo 2. Fuente propia. 78
Figura No. 4-46. Arquitectura Red Neuronal – Modelo 2 78
Figura No. 4-47. Resultados proceso de entrenamiento Red Neuronal – Modelo 2. Fuente propia. 79
Figura No. 4-48. Entropía cruzada fases de entrenamiento Modelo 2. Fuente propia. 80
Figura No. 4-49. Evolución gradiente y controles de validación – Red Neuronal Modelo 2. Fuente propia. 81
Figura No. 4-50. Histograma – Error en la predicción –Red Neuronal Modelo 2. Fuente propia. 81
Figura No. 4-51. Curvas ROC y AUC Red Neuronal Modelo 2. Fuente propia. 83

Figura No. 4-52. Matrices de confusión fases Red Neuronal Modelo 2. Fuente propia. 84
Figura No. 4-53. Matrices Multiclase CAP, CHECK y COV. Red Neuronal Modelo 2. Fuente propia. 85
Figura No. 5-1. Escenarios de Exactitud y Precisión. Fuente propia. 89
Figura No. 5-2. Proceso diagnóstico de celdas UMTS. Fuente propia. 95
Figura No. 5-3. Tiempo de entrenamiento Modelo 1- Técnica Árbol de Decisión. Fuente propia....96
Figura No. 5-4. Tiempo de entrenamiento Modelo 2- Técnica Árbol de Decisión. Fuente propia....97
Figura No. 5-5. Tiempo de entrenamiento Modelo 1 – Técnica Red Neuronal. Fuente propia. 98
Figura No. 5-6. Tiempo de entrenamiento Modelo 2 – Técnica Red Neuronal. Fuente propia. 98
Figura No. 6-1. Aplicación Guide Matlab. Fuente propia. 99
Figura No. 6-2. Diseño interfaz de usuario modelo 1. Fuente propia. 100
Figura No. 6-3. GUI – Modelo 1. Fuente propia. 100
Figura No. 6-4. Tabla de información celdas UMTS. Fuente propia. 101
Figura No. 6-5. Asociación código y árbol de decisión – Modelo 1. Fuente propia. 101
Figura No. 6-6. Resultados clasificación Modelo 1. Fuente propia. 102
Figura No. 6-7. GUI – Clasificación Modelo 2. Fuente propia. 103
Figura No. 6-8. Asociación código y árbol de decisión – Modelo 2. Fuente propia. 104
Figura No. 6-9. Resultados clasificación Modelo 2. 104

INDICE DE TABLAS

Tabla No. 2-1. Asignación modulación vs CQI. Fuente [6]. 10
Tabla No. 3-1. Funciones de activación más comunes en RNA. Fuente [20] 37
Tabla No. 3-2. Características técnicas de Machine Learning. Fuente propia. 39
Tabla No. 4-1. Raw data celdas UMTS. Fuente propia.....40
Tabla No. 4-2. Raw data celdas UMTS con Diagnóstico. Fuente propia. 41
Tabla No. 4-3. Matriz de Confusión. Fuente propia.....50
Tabla No. 4-4. Codificación salidas Modelo 1 - Red Neuronal. Fuente propia.....64
Tabla No. 4-5. Codificación salidas Modelo 2 Redes Neuronales. Fuente propia.....75
Tabla No. 4-6. Asignación clases – Red Neuronal Modelo 2. Fuente propia. 82
Tabla No. 5-1. Comparación de métricas - Técnicas de clasificación modelo 1. Fuente propia.....87
Tabla No. 5-2. Ranking métrica Exactitud Modelo 1. Fuente propia. 88
Tabla No. 5-3. Ranking métrica Tasa de Error Modelo 1. Fuente propia. 88
Tabla No. 5-4. Ranking métrica Sensibilidad Modelo 1. Fuente propia. 88
Tabla No. 5-5. Ranking métrica Especificidad Modelo 1. Fuente propia. 89
Tabla No. 5-6. Ranking métrica Precisión Modelo 1. Fuente propia. 90
Tabla No. 5-7. Ranking métrica Score F1 Modelo 1. Fuente propia. 90
Tabla No. 5-8. Consolidado ranking métricas – Técnicas de Clasificación Modelo 1. Fuente propia. 91
Tabla No. 5-9. Comparación de métricas - Técnicas de clasificación modelo 2. Fuente propia.....91
Tabla No. 5-10. Ranking métrica Exactitud Modelo 2. Fuente propia. 92
Tabla No. 5-11. Ranking métrica Tasa de Error Modelo 2. Fuente propia. 92
Tabla No. 5-12. Ranking métrica Sensibilidad Modelo 2. Fuente propia. 93
Tabla No. 5-13. Ranking métrica Especificidad Modelo 2. Fuente propia. 93
Tabla No. 5-14. Ranking métrica Precisión Modelo 2. Fuente propia. 93
Tabla No. 5-15. Ranking métrica Valor F1 Modelo 2. Fuente propia. 93
Tabla No. 5-16. Consolidado Métricas – Técnicas de Clasificación Modelo 2. Fuente propia. 94

RESUMEN

En esta investigación se lleva a cabo el análisis, la validación y la implementación de un algoritmo tomado del amplio mundo del Machine Learning, para generar uno o varios modelos que puedan apoyar en los procesos de optimización de las redes UMTS (Universal Mobile Telecommunications System - Sistema universal de telecomunicaciones móviles), y más específicamente en la fase de diagnóstico de las celdas. Se presentan algunos de los contadores y KPIs (Key Performance Indicators – Indicadores claves de desempeño) estadísticos de las celdas UMTS que normalmente son utilizados por los ingenieros de optimización para diagnosticar el estado de las celdas, y que servirán para armar un data set que pueda ser utilizado por las técnicas de Machine Learning como información de entrada para entrenar los modelos.

Para esto se realiza el análisis de varias técnicas de Machine Learning y dependiendo de las características y condiciones del data set disponible, se escogen dos de las técnicas, las cuales serán entrenadas y evaluadas para finalmente escoger la técnica de mejor desempeño y sobre la cual se realizará la implementación en un lenguaje de programación que permita realizar diagnósticos futuros de las celdas UMTS.

PALABRAS CLAVE

KPIs, Machine Learning, Data set, Técnica, Machine Learning.

1. GENERALIDADES

1.1 INTRODUCCIÓN

Sin lugar a dudas las redes de telefonía móvil celular han representado un gran avance en las comunicaciones móviles, ya que les facilita a los usuarios el acceso a servicios de voz y datos de manera inalámbrica. Pero una red móvil celular sin importar el operador o vendor que la despliegue, siempre demandará procesos de optimización para una adecuada prestación de los servicios. El objetivo principal del proceso de optimización es que cada celda de la red móvil pueda atender la mayor cantidad de usuarios con la mejor calidad posible. Para esta importante tarea los operadores de telefonía celular asignan grupos especiales conformados por ingenieros expertos que se encargan de realizar un diagnóstico del status de las celdas UMTS y de acuerdo a los resultados implementar planes de acción para ajustar los parámetros de configuración que permitan mejorar el desempeño de las celdas.

En los últimos años una gran cantidad de procesos de diferentes disciplinas se han enfocado en el uso de paradigmas de la inteligencia computacional para dar solución a problemas complejos que no pueden ser abordados mediante los modelos que usan los algoritmos de computación tradicionales. Dentro de este proyecto de investigación se busca desarrollar un modelo de Machine Learning que apoye la fase de diagnóstico de celdas UMTS (Las redes UMTS en Colombia aún seguirán vigentes por varias razones entre las que se destacan, el hecho de que no se podrá apagar una red hasta que no exista otra red totalmente madura que la reemplace, también porque existe un gran número de clientes que no poseen teléfonos con VoLTE y por ende deben cursar las llamadas de voz en la red UMTS y finalmente que los operadores proyectan recuperar la inversión realizada en los equipos de red UMTS a largo plazo) y que entregue como resultado una determinada clasificación de acuerdo a los posibles escenarios de optimización. Como beneficios se conseguirían entre otros, la agilidad en el proceso de optimización de la red con un menor uso de personal calificado. Al final todos los esfuerzos realizados en la optimización de una red se verán reflejados tanto en la satisfacción de los clientes como en un mayor ebitda para los operadores.

1.2 PROBLEMA DE INVESTIGACIÓN

El problema de investigación se define a través del planteamiento, la formulación y sistematización del problema.

1.2.1 Planteamiento del problema

Actualmente los operadores de telefonía celular realizan los procesos de optimización de la red utilizando una gran cantidad de recursos tales como herramientas propietarias, personal experto y tiempo de procesamiento. Dentro del ciclo de optimización se tiene la fase de diagnóstico de las

celdas para la cual son indispensables los KPIs que están asociados a contadores estadísticos tales como número de usuarios, potencia consumida, códigos disponibles, cantidad de tráfico, entre otros. El análisis de estos KPIs se realiza de forma manual por los ingenieros expertos de optimización quienes correlacionan varias de las variables obtenidas en los KPIs para dar finalmente un diagnóstico de las celdas e indicando si requieren o no un plan de acción. Pero podría existir una forma mucho más ágil que permita realizar el diagnóstico de las celdas haciendo uso de la IA (inteligencia artificial), pues muchos de estos modelos o paradigmas son alimentados con data sets que en este caso se podrían armar con los KPIs que se tienen de cada una de las celdas de la red. Dentro de la IA existen varias ramas como la lógica difusa, redes neuronales, sistemas expertos o la combinación de algunas de estas con las cuales se podría abordar el reto de diagnóstico de las celdas UMTS. La posibilidad de desarrollar un modelo que se adecue a las condiciones del proyecto son altas considerando que ya existen algunas aplicaciones que se asemejan tales como las de diagnósticos médicos. Gracias a que se cuenta con varios data sets, será posible evaluar con gran precisión el desempeño de los modelos analizados ya que se tienen tanto las variables de entrada que son los KPIs y las variables de salida que corresponden a los diagnósticos realizados por los ingenieros expertos. Al final de todo el proceso de investigación y análisis se obtendrán importantes conclusiones respecto del desempeño de los modelos de IA para el diagnóstico de celdas UMTS.

1.2.2 Formulación del problema

¿Es posible apropiar una técnica o modelo tomado del mundo de la Inteligencia Artificial, que pueda ser configurado y utilizado en la fase de diagnóstico de celdas UMTS tomando como entrada los KPIs de las celdas?

1.2.3 Sistematización del problema

- ¿Se puede generar un data set con los KPIs de las celdas UMTS?
- ¿Cuáles técnicas de Machine Learning se pueden aplicar para generar los modelos?
- ¿Qué técnica de Machine Learning ofrece mejor desempeño en el diagnóstico de las celdas?

1.3 OBJETIVOS

1.3.1 Objetivo general

Implementar (desarrollar un algoritmo en Matlab) una técnica de inteligencia computacional (engloba numerosas disciplinas de la IA) en particular Machine Learning que, pueda ser utilizada en la fase de diagnóstico (análisis de KPIs) de celdas UMTS apoyando así el proceso de optimización de la interfaz de radio de una red de telefonía móvil celular.

1.3.2 Objetivos específicos

- Identificar técnicas o modelos de inteligencia computacional que sean viables para realizar el diagnóstico de las celdas UMTS.
- Probar los modelos con el data set para determinar la exactitud, tasa de error, sensibilidad, especificidad y precisión de los resultados.
- Comparar los resultados obtenidos a partir de los modelos desarrollados: técnica árbol de decisión y red neuronal.
- Determinar los tiempos de procesamiento que toma el algoritmo para realizar el procesamiento de la información contenido en el data set.
- Implementar el modelo en la herramienta de simulación "Simulink" o mediante "Guide" de Matlab.

1.4 JUSTIFICACIÓN

Según el informe anual "Redes Móviles en Colombia" emitido por la CRC (Comisión de Regulación de Comunicaciones) en el año 2018 Colombia cerró con un total de 131.619 sectores de 3G pertenecientes a los cuatro operadores Claro, Tigo, Movistar y Partners [1]. Este número de sectores representan el 59% del total de sectores 2G, 3G y 4G desplegados por los operadores en todo el País. Para esta gran cantidad de sectores 3G los operadores deben destinar un gran número de recursos orientados al proceso de optimización, en especial a la fase de diagnóstico de las celdas, la cual es realizada por un grupo de ingenieros expertos. Este proceso de diagnóstico en especial demanda una gran cantidad de tiempo, por lo que en muchas ocasiones se actúa tarde sobre las celdas degradadas con los planes de optimización. Los planes de acción tardíos no solo afectan la percepción de los usuarios, sino que adicional podrían generar multas por incumplimiento de los indicadores de desempeño que estable el ente regulador CRC. Por lo tanto, al lograr definir un algoritmo de Machine Learning que apoye a los ingenieros expertos en las tareas de diagnósticos de las celdas, se conseguirá un mayor dinamismo en la definición de planes de acción permitiendo actuar de manera oportuna sobre las celdas con problemas. Adicional se conseguirán varios beneficios entre los que se encuentran los siguientes: agilidad en los procesos de optimización, reducción del OPEX, mejora de la calidad de la red, incremento de los ingresos y el más importante la satisfacción de los usuarios.

2. MARCO TEÓRICO

2.1 REDES UMTS

Los sistemas móviles de tercera generación (3G) dentro de los que se encuentra 3G UMTS, surgen como la evolución de los ya existentes sistemas de segunda generación (2G) GSM (Global System for Mobile Communications) y CDMA (Code Division Múltiple Access). Las principales razones que motivaron el desarrollo de 3G fueron la necesidad de una interfaz en el lado radio acceso que pudiera soportar una mayor cantidad de usuarios y que a su vez también se ofreciera una mayor velocidad de usuario que en teoría llegaría a los 2 Mbps. Todo esto fue recogido por la UIT (Unión Internacional de Telecomunicaciones) en el proyecto IMT-2000 [2]. La primera entrega de las especificaciones en la cual se cumplieron estos objetivos se denominó Release 99 y fue liberada en el año 1999. Los organismos encargados de las especificaciones técnicas del 3GPP (3rd Generation Partnership Project: Proyecto Asociación de Tercera Generación) continuaron los trabajos dando como fruto nuevas liberaciones en UMTS tales como los Releases 4, 5, 6, 7 y 8. En estos releases evolucionaron aún más los sistemas 3G para mejorar la gestión de los recursos de radio con nuevos canales que soportan mayores velocidades. La siguiente figura ilustra la arquitectura básica del sistema 3G-UMTS la cual está basada en la red de radio acceso UTRAN (UMTS Radio Access Network) [3]. Sus principales elementos son: el NodoB, también conocido como estación base y la RNC (Radio Network Controller), la cual concentra un grupo de NodosB. El sistema está conectado al dominio de conmutación de circuitos MSC (Mobile Switching Center) para los servicios de voz y al dominio de conmutación de paquetes SGSN (Service GPRS Node Support) para los servicios de datos.

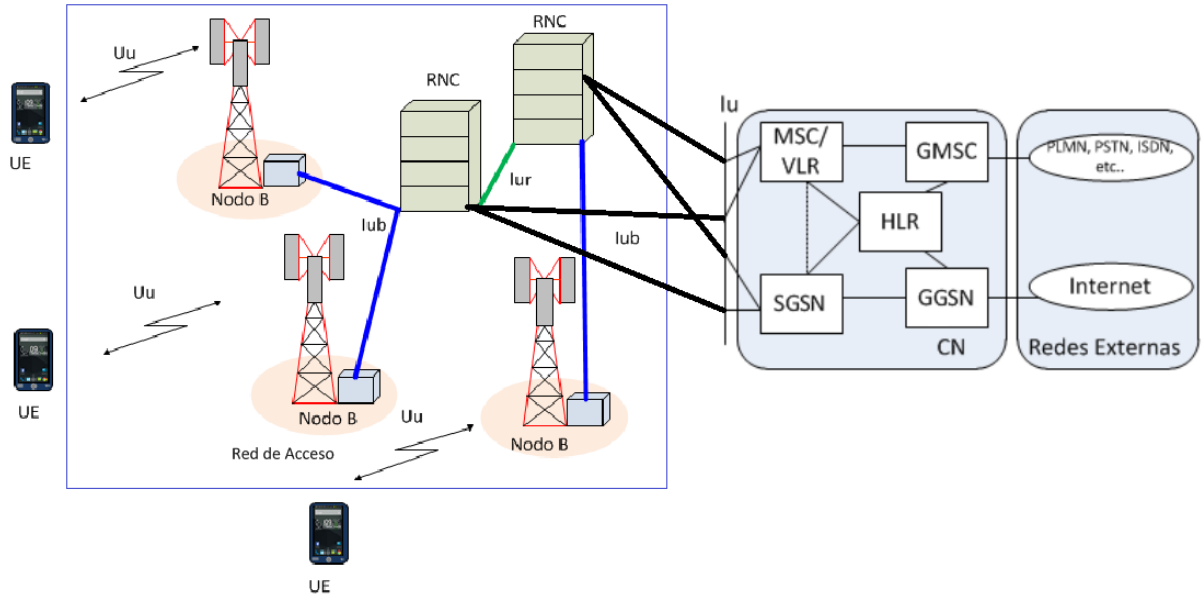


Figura No. 2-1. Arquitectura 3G-UMTS. Fuente [4]

El estándar 3G UMTS se basa en la técnica de acceso WCDMA (Wideband Code División Multiple Access). La forma de diferenciar a cada uno de los usuarios es mediante el uso de códigos. De esta manera todos los usuarios reciben información todo el tiempo, pero solo decodifican la información que corresponde al código asignado.

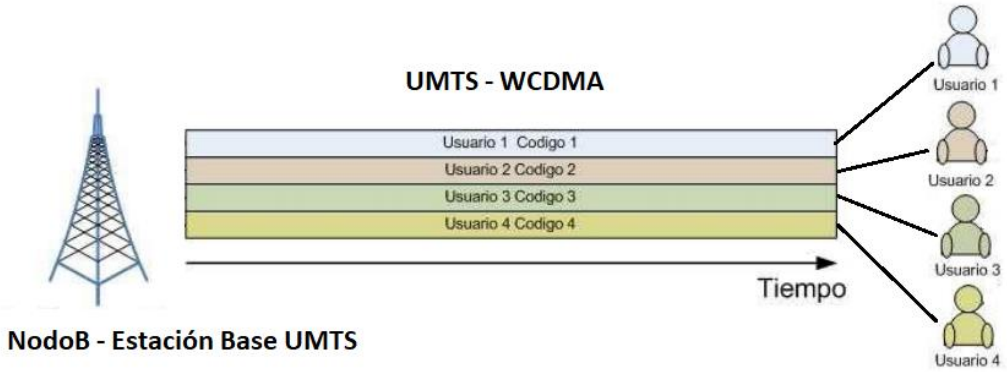


Figura No. 2-2. Esquema comunicación WCDMA UMTS. Fuente propia.

Pero para que los usuarios puedan recibir un óptimo servicio es necesario evaluar una gran cantidad de variables asociadas a cada celda. Estas variables de información son almacenadas normalmente en contadores estadísticos dentro de los sistemas de supervisión y se asocian a lo que se conoce como KPIs (Key Performance Indicators). En el mercado de las telecomunicaciones existen algunas herramientas de monitorización que ofrecen interfaces abiertas para recolectar la información de los gestores y posteriormente post-procesar la información contenida en los contadores con el objetivo de generar reportes, alarmas, etc. Algunas de estas herramientas propietarias son Nastar y Performance Surveillance o también conocida como PRS. Sin embargo, estas herramientas no tienen el alcance de correlacionar diferentes variables de entrada y entregar planes de acción, solo facilitan la visualización de KPIs estándar o definidos por el usuario. En otras palabras, con estas herramientas solo se pueden generar reportes individuales de los KPIs de una forma amigable [5]. A continuación, se relacionan algunos de los contadores y KPIs que se analizan de una celda UMTS.

2.1.1 KPIs redes UMTS

- **Cell Load**

Este KPI hace referencia a que tan cargada esta la celda UMTS de usuarios que demandan un throughput. Para entenderlo mejor, asumamos que una celda UMTS puede entregar en total un throughput de 8000 Kbps entre todos los usuarios que atiende. Si en la celda se encuentran conectados 4 usuarios, cada usuario tendría en promedio 2000 Kbps, pero si son 16 usuarios los conectados, entonces cada usuario tendría en promedio un throughput de 500 Kbps. En otras palabras, entre más usuarios menor throughput se les puede asignar.

Para determinar el Cell Load de una celda se divide el mínimo throughput que debe ofrecer la celda a los usuarios entre el promedio del máximo throughput que están alcanzando los usuarios.

$$\%\boldsymbol{Cell\ Load} = \frac{\boldsymbol{Mínimo\ Throughput\ que\ ofrece\ la\ Celda}}{\boldsymbol{Promedio\ del\ Máximo\ Throughput\ de\ los\ usuarios}}$$

Por ejemplo, si se configura una celda para que ofrezca a sus usuarios al menos un throughput de 875 Kbps y los usuarios en promedio pueden alcanzar un throughput máximo de 1200 Kbps. Esto quiere decir que la celda tiene una carga de aproximadamente del 73%. En otras palabras, la celda podría admitir más usuarios para compartir el throughput total que dispone la celda.

- **Potencia Tráfico R99**

Este KPI está asociado a la potencia que se consumen los usuarios para los servicios de Reléase 99 como la voz. La siguiente imagen ilustra con tres ejemplos como se reparte el consumo de potencia de una celda UMTS.

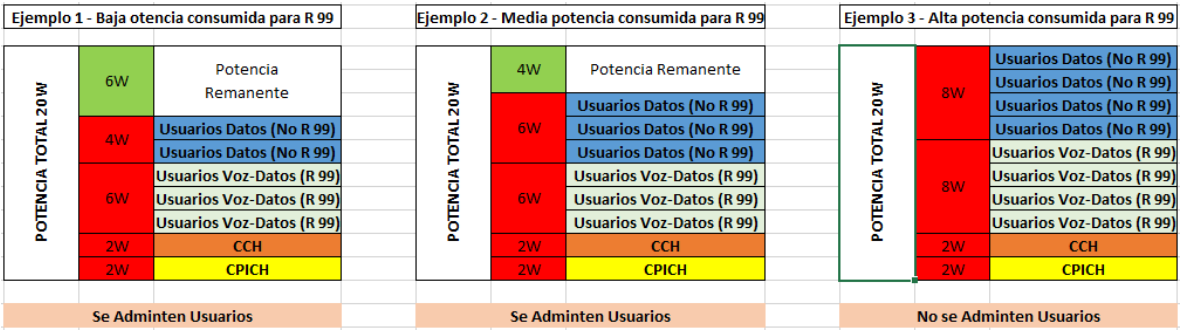


Figura No. 2-3. Consumo de potencia celdas UMTS. Fuente propia.

Se puede decir que la potencia de una celda UMTS es repartida entre tres clases de servicios, la primera corresponde a la potencia entregada a los canales de control, esta potencia es fija y corresponde aproximadamente a un 20% de la potencia total de la celda, sobre las figuras está potencia se representa en los canales de control CCH y el canal piloto CPICH. La siguiente clase corresponde a la potencia entregada para los servicios R99 (Release 99) tales como la voz, este consumo de potencia dependerá de que tantos usuarios de voz o datos estén conectados y a que distancia de la celda. En la tercera clase se ubican los servicios no R99 como por ejemplo los servicios HSDPA, estos servicios no R99 por lo general son los que más consumo de potencia representan en la celda y dependerá de la potencia remanente que le dejen los servicios de R99.

Para realizar su cálculo se divide la potencia consumida por los servicios de R99 entre la potencia total que tiene configurada la celda.

$$\%\boldsymbol{Potencia_R99} = \frac{\boldsymbol{Potencia\ consumidad\ por\ R99}}{\boldsymbol{Potencia\ total\ de\ la\ Celda}}$$

- **Códigos disponibles para HSDPA**

En el estándar UMTS se define un recurso llamado árbol de códigos. Cuando un usuario está conectado a una celda y está realizando una descarga, entre más códigos tenga asignados, mayor será el throughput que puede alcanzar. En otras palabras, el throughput de un usuario es directamente proporcional a los códigos asignados. La idea es que cuando un usuario quiere realizar una descarga existan códigos disponibles en la celda.

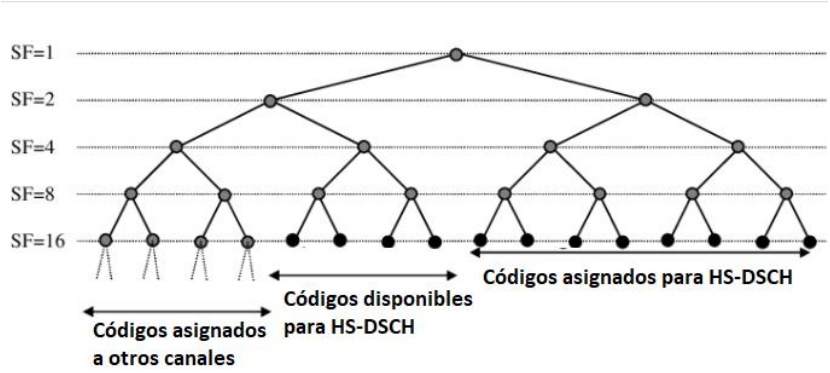


Figura No. 2-4. Códigos celdas UMTS. Fuente propia.

El cálculo es muy sencillo, solo basta restar al total de códigos de la celda los códigos utilizados.

$$\boldsymbol{Códigos\ disponibles\ para\ HSDPA = Códigos\ Totales - Códigos\ usados}$$

- **Calidad Alcanzable en Recepción**

Este contador está asociado al indicador de calidad de señal en recepción que puede alcanzar en promedio un usuario de la celda. En las estadísticas de desempeño que entregan las celdas UMTS se tiene el CQI por sus siglas en inglés (Channel Quality Indicator). Este valor es entregado por la celda directamente y no requiere de cálculos adicionales. Solo para entender como esta variable influye en el desempeño de una celda, se presenta a continuación la tabla tomada del 3GGP TS.25.214 [6], donde se relaciona el valor de CQI con la modulación que se puede utilizar. Entre mayor sea el CQI, mayor será la modulación y por ende mayor el throughput que puede alcanzar el usuario.

Tabla No. 2-1. Asignación modulación vs CQI. Fuente [6].

CQI value	Transport Block Size	Number of HS-PDSCH	Modulation	Reference power adjustment Δ
0	N/A	Out of range		
1	137	1	QPSK	0
2	173	1	QPSK	0
3	233	1	QPSK	0
4	317	1	QPSK	0
5	377	1	QPSK	0
6	461	1	QPSK	0
7	650	2	QPSK	0
8	792	2	QPSK	0
9	931	2	QPSK	0
10	1262	3	QPSK	0
11	1483	3	QPSK	0
12	1742	3	QPSK	0
13	2279	4	QPSK	0
14	2583	4	QPSK	0
15	3319	5	QPSK	0
16	3565	5	16-QAM	0
17	4189	5	16-QAM	0
18	4664	5	16-QAM	0
19	5287	5	16-QAM	0
20	5887	5	16-QAM	0
21	6554	5	16-QAM	0
22	7168	5	16-QAM	0
23	9719	7	16-QAM	0
24	11418	8	16-QAM	0
25	14411	10	16-QAM	0
26	14411	10	16-QAM	-1
27	14411	10	16-QAM	-2
28	14411	10	16-QAM	-3
29	14411	10	16-QAM	-4
30	14411	10	16-QAM	-5

- **Tráfico de voz en Erlang**.

El siguiente contador tiene relación con la cantidad de tráfico en Erlangs que está cursando en promedio la celda. Y al igual que la anterior variable no requiere ningún calculo, pues es entregada directamente por la celda dentro de los contadores que maneja. Para entender un poco como esta variable ayuda a evaluar el estado de la celda, consideremos el siguiente ejemplo. Supongamos que existen unos pocos usuarios conectados a una celda los cuales demandan un tráfico de voz normal, es decir pocos usuarios que generan poco tráfico de voz en Erlangs, sin embargo, estos usuarios demandan demasiada potencia, más de la habitual. Esto quiere decir que la celda está atendiendo usuarios de voz que se encuentran en el borde de la celda y al estar tan lejos demandan más potencia del promedio.

- **Usuarios Activos de HSPS (High Speed Paket Switch)**

Los usuarios Activos de HSPS son los usuarios que tienen datos pendientes por recibir y que se encuentran en el buffer de la radio base. Esta variable al igual que las anteriores no requiere calculo, ya que es entregado por la celda directamente. Para entender mejor esta variable, se presenta la siguiente figura que relaciona el throughput que pueden obtener los usuarios en función de la cantidad de usuarios activos que tiene la celda.

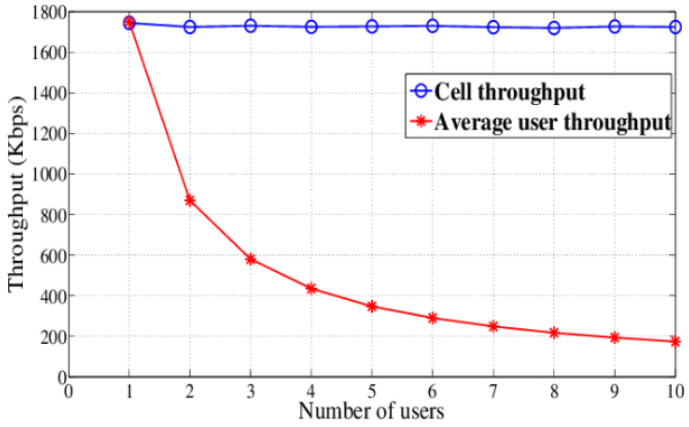


Figura No. 2-5. Throughput VS número de usuarios. Fuente [7].

- **TTI Load – (TTI – Transmission Time Interval)**

El TTI es un parámetro de UMTS relacionado con cada cuanto tiempo se realiza la encapsulación de los datos desde las capas más altas en las tramas para transmisión sobre la capa de radio link. En UMTS se define el TTI normalmente en 2 mili segundos, es decir que cada 2 mili segundos se realiza la asignación de recursos a los usuarios que tienen datos en el buffer de la radio base para ser transmitidos. Para tener una mejor idea del aporte que puede brindar esta variable al desempeño de la celda, en la siguiente figura se ilustra un ejemplo de cómo se asignarían los recursos de una celda a los usuarios que les brinda cobertura. En el primer intervalo solo 1 usuario de los 5 totales se programó en el scheduler de la radio base para trasmitir datos, en el intervalo 2 y 3 ya son 2 usuarios de los 5, pero en el intervalo 7 son 4 usuarios los que tiene la celda programados en su buffer para transmitir datos. De esta manera se tiene que el intervalo 1 el TTI load es del 20% y en el intervalo 7 el TTI load es del 80%.

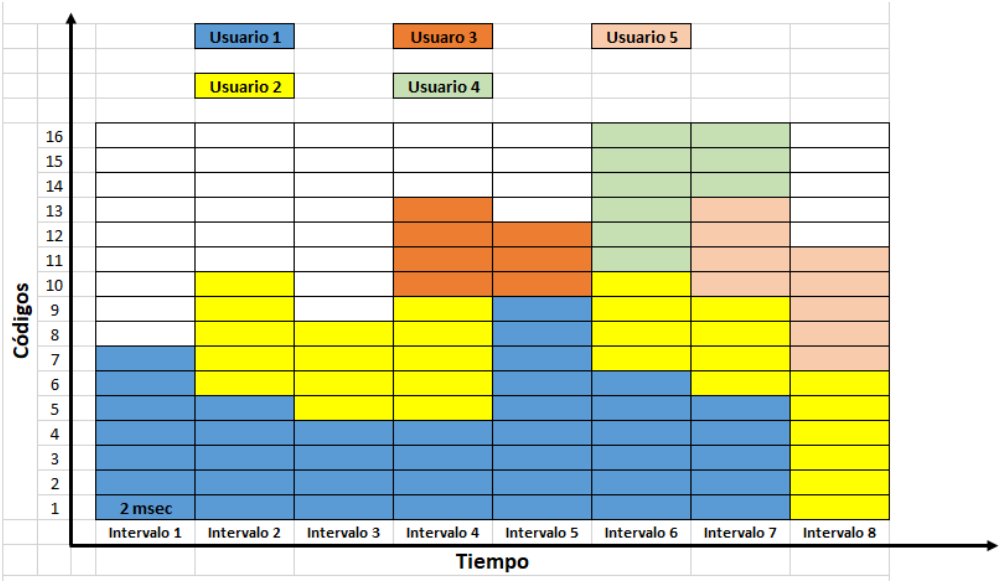


Figura No. 2-6. TTI load celda UMTS. Fuente propia.

Para realizar el cálculo de este KPI se dividen los usuarios activos entre el total de los usuarios que atiende la celda, recordemos que los usuarios activos son los usuarios que tiene datos en el buffer de la radio base.

$$\boldsymbol{TTI\ Load = \frac{Usuarios\ Activos}{Total\ usuarios}}$$

Los KPIs y contadores son indispensables para desarrollar muchos de los procesos de optimización de las celdas UMTS y que se enfocan principalmente en maximizar el uso de las celdas considerando criterios de calidad tales como la accesibilidad, retenibilidad e integridad según la recomendación E.800 del CITT (Comité Consultivo Internacional Telegráfico y Telefónico) de la UIT (Unión Internacional de Telecomunicaciones) [8]. En otras palabras, la optimización de las redes UMTS busca que los usuarios atendidos puedan explotar al máximo los servicios provistos por la red UMTS. El termino optimización de redes celulares puede abarcar una gran cantidad de aspectos, pero en la mayoría de ellos se enfocan a temas de cobertura y capacidad de la red [9]. Para los propósitos de optimizar la cobertura, los ingenieros de optimización hacen uso de varias herramientas propietarias entre las que se encuentran equipos de drive test, servidores de simulación de cobertura e indicadores estadísticos. Los vendors de estas herramientas generan sus propios manuales y procedimientos, y aunque no varían en esencia sus funcionalidades, sí definen algunas características que los diferencian unos de otros, por ejemplo, que algunos puedan realizar un análisis autónomo y otros requieran la intervención de ingenieros expertos que puedan configurar los parámetros y analizar los resultados para tomar las decisiones de optimización requeridas. Estas tareas se enfocan especialmente para tener una visión del estatus de la capa de acceso de una red UMTS. Por su parte la optimización de capacidad se soporta principalmente en los KPI estadísticos que se obtienen de los gestores, estos indicadores son la materia prima que se utilizan como información de entrada para los procesos de diagnóstico de los diferentes elementos e interfaces de la red UMTS. El procesamiento de toda la información de KPIs y su posterior análisis demanda de los operadores de telefonía móvil celular una gran cantidad de recursos como personal calificado, herramientas propietarias y tiempo [10]. Y aunque los procesos pueden ser muy mecánicos y repetitivos, si son tediosos por el procesamiento que demanda para correlacionar toda la información en pro de encontrar los problemas.

2.2 INTELIGENCIA ARTIFICIAL Y MACHINE LEARNING

En relación a la IA se puede decir que existen varias definiciones de un gran número de autores tales como Rich & Knight quienes en 1991 la definieron como *"La inteligencia artificial es el estudio de cómo hacer que las computadoras hagan cosas que por el momento los humanos hacen mejor"*. En este orden de ideas la IA involucra algoritmos especiales corriendo en máquinas las cuales usan una gran variedad de información de entrada para aprender de los datos y posteriormente utilizar todo lo aprendido en la toma de decisiones semejante a como lo haría un ser humano [11].

A nivel mundial se destaca el esfuerzo de la multinacional Vodafone para implementar comercialmente algoritmos de IA en el software de sus redes móviles en España, con el cual se permitirá optimizar y mejorar su rendimiento. En todos estos trabajos han estado involucrados los tres principales proveedores de equipos de red móvil que son Huawei, Ericsson y Nokia. El principal problema que abordaron se enfocaba en la gestión del traspaso de llamadas entre las diferentes bandas de LTE. Lo anterior puesto que se pronostica que para el 2021 llegarían a tener hasta nueve frecuencias haciendo mucho más complejo el proceso de handover [12].

En la actualidad existe una gran variedad de aplicaciones en diferentes ramas y disciplinas que están implementando tecnologías basadas en IA y que son utilizadas para ayudar a los humanos en sus tareas de trabajo. Una de las ramas de la IA es el Machine Learning, la cual se clasifica en dos técnicas dependiendo del tipo de aprendizaje, uno llamado aprendizaje supervisado y el otro aprendizaje no supervisado.

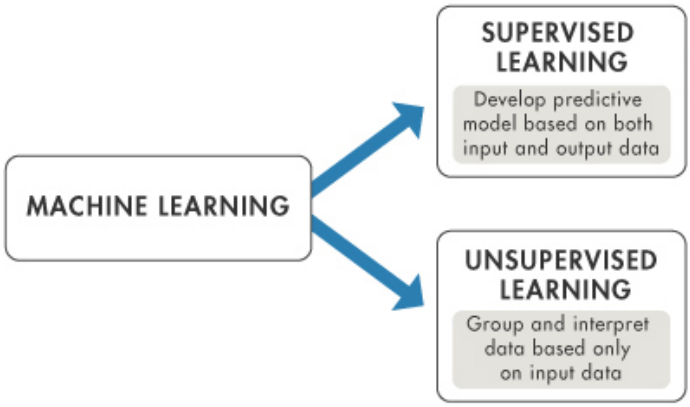


Figura No. 2-7. Técnicas de aprendizaje autónomo. Fuente [13].

2.2.1 Aprendizaje supervisado

Los algoritmos de Machine Learning pueden encontrar patrones en los datos que generan conocimiento y que al final contribuyen en la toma de mejores decisiones. El aprendizaje que realiza el algoritmo puede ser supervisado cuando se crea un modelo que realiza predicciones en función de las pruebas en presencia de incertidumbre [13]. Esta técnica se denomina aprendizaje supervisado porque el entrenamiento del modelo se realiza tanto con la información de entrada como con la información de salida que corresponden a las respuestas a la información de entrada.

Las técnicas de clasificación se enfocan en predecir respuestas discretas como por ejemplo ayudar a diagnosticar si un tumor es cancerígeno o es benigno, o para definir si un correo electrónico es spam o no. Esta técnica se podría utilizar si los datos de entrada se pueden etiquetar, categorizar o dividir en grupos. Algunos de los algoritmos que se usan habitualmente para realizar la clasificación son: Arboles de decisión, máquina de vectores de soporte, vecino más cercano, clasificadores bayesianos, análisis discriminante, regresión logística y redes neuronales.

En el caso de las técnicas de regresión, se podrían predecir respuestas continuas tales como presión atmosférica o consumo de energía. Se utilizaría esta técnica de regresión si la respuesta corresponde a un número real.

2.2.2 Aprendizaje no supervisado

El aprendizaje no supervisado se basa en encontrar patrones ocultos o también estructuras intrínsecas dentro de los datos de entrada. Es útil para inferir información a partir de un conjunto de datos que no tienen respuestas etiquetadas.

La técnica de aprendizaje no supervisado más común es el clustering y se emplea para realizar el análisis de datos de tipo exploratorio, con el propósito de hallar patrones o agrupaciones ocultas en los datos. Dentro de las aplicaciones que se le pueden dar al clustering se encuentran el reconocimiento de objetos, la investigación de mercados y el análisis de secuencias genéticas. En el campo de la telefonía móvil podría ayudar a optimizar las coordenadas donde se deben construir las estaciones base. Algunos de los algoritmos más habituales que se utilizan para realizar el clustering son: k-means y k-medoids, clustering jerárquico, modelos de mezclas gaussianas, modelos de Markov ocultos, mapas autoorganizados, clustering difuso de c-means y clustering sustractivo [13].

3. ANÁLISIS DE TÉCNICAS Y MODELOS DE MACHINE LEARNING

3.1 INTELIGENCIA ARTIFICIAL

Dar una definición de que es la Inteligencia Artificial (IA) es bastante complicado debido a que es un tema muy complejo que comprende e involucra muchas disciplinas tales como matemáticas, psicología, biología, sociología, mecánica, informática, computación, y electrónica. Por lo anterior es posible encontrar en la literatura una gran cantidad de definiciones acerca de ella, dependiendo del punto de vista o enfoque desde el que se aborde.

En los últimos años se ha dado un crecimiento exponencial de las tecnologías que han implementado algún tipo de Inteligencia Artificial, en especial, con el objetivo de ser utilizadas para ayudar en procesos donde se requiere realizar el análisis de una gran cantidad de información.

La Inteligencia Artificial intenta construir maquinas que puedan ser capaces de imitar el comportamiento inteligente de los seres humanos por medio de máquinas computacionales que implementen procesos de aprendizaje, razonamiento y autocorrección. Se puede decir que hace referencia a la inteligencia que las maquinas pueden mostrar.

En la siguiente figura se puede ver como la Inteligencia Artificial corresponde al concepto más amplio en el cual se engloban otros conceptos como Inteligencia Computacional la cual tiene dentro de sus áreas el Machine Learning.

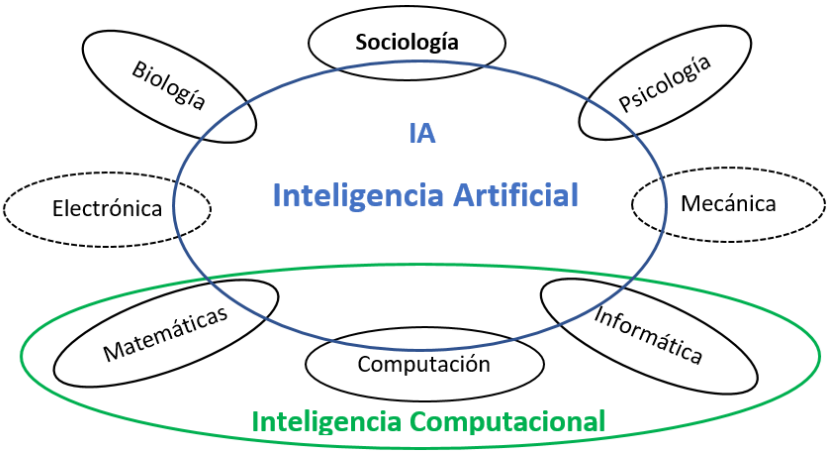


Figura No. 3-1. Concepto Inteligencia Artificial. Fuente propia.

3.2 INTELIGENCIA COMPUTACIONAL

La Inteligencia Computacional es un área muy importante dentro del mundo de la Inteligencia Artificial, la cual se enfoca en el diseño de sistemas informáticos con un cierto grado de inteligencia mediante el aprendizaje, adaptación, razonamiento y evolución. Siempre buscando imitar a la naturaleza y el razonamiento lingüístico del ser humano en la resolución de problemas de gran complejidad.

Dentro de La Inteligencia Computacional se identifican dos propósitos que son: en primer lugar y como objetivo científico el comprender los elementos que dan lugar al comportamiento inteligente, y en segundo lugar el relacionado con el objetivo tecnológico el cual se enfoca en especificar los métodos que se deben considerar en el diseño de los sistemas inteligentes.

Por medio de estas técnicas se crean algoritmos inteligentes con capacidades para tareas tales como clasificar datos, predecir comportamientos, extraer información de grandes bases de datos o controlar sistemas complejos. Una visión de la Inteligencia Computacional corresponde a lo que se conoce como el Machine Learning. La siguiente imagen muestra cómo están relacionados los conceptos Inteligencia Artificial, Inteligencia Computacional y Machine Learning.

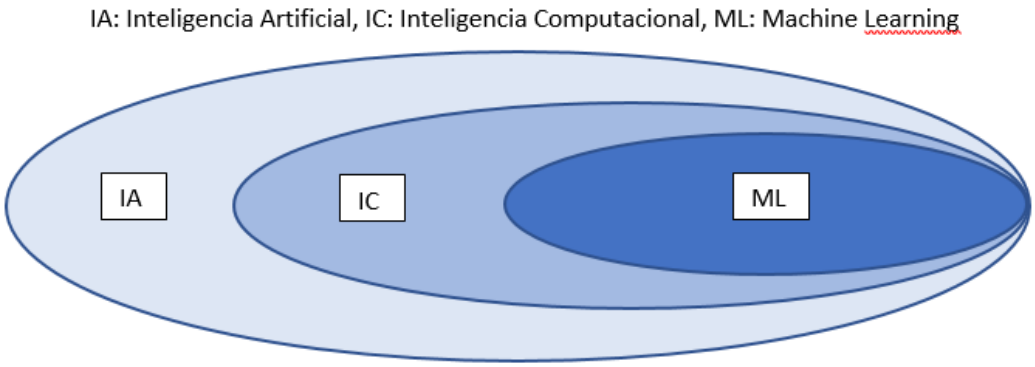


Figura No. 3-2. Esquema de clasificación IA, IC y ML. Fuente propia. Fuente propia.

3.3 MACHINE LEARNING

El Machine Learning conocido también como aprendizaje autónomo involucra a un grupo de técnicas enfocadas en el análisis de datos mediante los cuales se les enseña a los ordenadores a que ejecuten tareas que serían normales para los seres humanos o animales. En el Machine Learning se establecen algoritmos de aprendizaje automático los cuales emplean métodos de cálculo para aprender directamente de los datos de la información sin necesidad de utilizar programación basada en reglas. Los algoritmos pueden incrementar su rendimiento de forma adaptativa si se van incrementando los datos de información de entrada durante el proceso de entrenamiento [13].

En otras palabras, se puede decir que el Machine Learning es la capacidad que tiene una unidad computacional para aprender, predecir y tomar decisiones tomando como base una gran cantidad de información. La eficiencia y precisión de las acciones del Machine Learning dependerá significativamente de la cantidad y calidad de los datos que son proporcionados durante el proceso de aprendizaje [14].

3.3.1 Propósito del Machine Learning

El propósito general del Machine Learning consiste en la creación de algoritmos que puedan ser usados posteriormente por maquinas en la realización de tareas específicas dando solución a problemas complejos o tediosos que normalmente un colaborador experto los resuelve repetitivamente. Los insumos para una técnica de Machine Learning son: Los datos, el algoritmo, la función objetivo y el modelo. La siguiente figura ilustra cómo se da el proceso del aprendizaje automático en el Machine Learning. Los datos son entregados al algoritmo de aprendizaje para que este a su vez pueda aprender de los ejemplos y al final entregar un modelo con el cual se puedan procesar los nuevos casos, ya sea en problemas de optimización, clasificación o predicción.

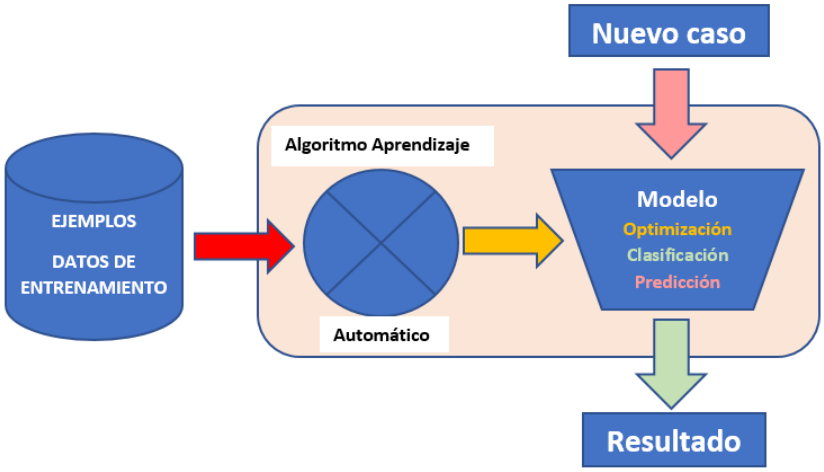


Figura No. 3-3. Aprendizaje Automático Machine Learning. Fuente propia.

3.3.2 ¿Cuándo se pueden usar las técnicas de Machine Learning?

En los últimos años el Machine Learning se ha convertido en una poderosa opción para resolver una gran cantidad de problemas de alta complejidad en los cuales no se disponen de fórmulas o ecuaciones que puedan correlacionar la gran cantidad de datos o variables que se manejan. Para el caso del diagnóstico de celdas UMTS, el Machine Learning es una buena opción ya que se dispone de una gran cantidad de información asociada a las celdas que pueden servir para alimentar el algoritmo. La información disponible de las celdas abarca una muy amplia variedad de escenarios ya que se incluyen todas las celdas UMTS de un operador de telefonía. La siguiente figura relaciona algunas de las posibles variables que pueden ser consideradas para el análisis mediante el uso de técnicas de Machine Learning [13].

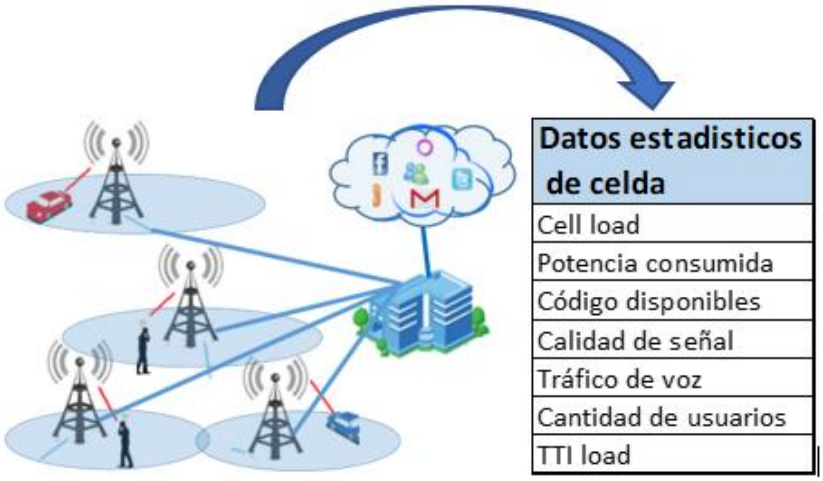


Figura No. 3-4. Variables de celda UMTS. Fuente propia.

El propósito de las técnicas de Machine Learning en este proyecto será tomar toda la información disponible de las celdas para entrenar un modelo que pueda dar solución al problema de diagnóstico de las celdas UMTS. Lo anterior con el propósito de apoyan en esta actividad de diagnóstico que normalmente es realizada por ingenieros expertos de telecomunicaciones. Como resultado del modelo se deberá indicar si una celda requiere o no un plan de acción que mejore su desempeño en la prestación del servicio hacia los usuarios móviles que se encuentran bajo su cobertura [15].

Las variables de información de las celdas UMTS son almacenadas normalmente en contadores estadísticos dentro de los sistemas de supervisión y se asocian a lo que se conoce en el mundo de las telecomunicaciones como los sub-sistemas de gestión.

3.3.3 ¿Cómo funciona el Machine Learning?

Dentro del aprendizaje automático se identifican tres técnicas que son: el **aprendizaje supervisado**, en el cual se entrena un modelo con datos conocidos tanto de entrada como de salida para poder predecir las salidas futuras, la segunda técnica corresponde al **aprendizaje no supervisado**, en el cual se buscan patrones ocultos o estructuras intrínsecas contenidos en los datos de entrada, es decir que no tienen en cuenta los datos de salida para generar el modelo, y la tercer técnica que es el **aprendizaje por refuerzo** la cual se puede ubicar en medio de las dos técnicas anteriores, es decir que solo se dispone de una parte de los datos esperados como si fuera un aprendizaje supervisado parcial. A continuación, se desarrollan con mayor profundidad las dos principales técnicas. En la siguiente imagen se ilustra la clasificación general de las diferentes técnicas del Machine Learning.

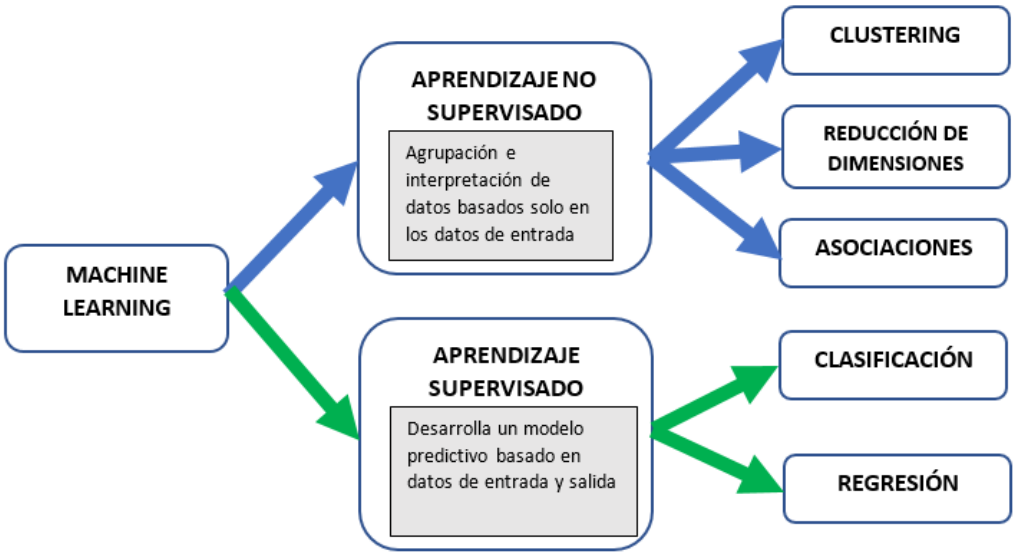


Figura No. 3-5. Técnicas de aprendizaje automático. Fuente propia.

3.4 TÉCNICAS DE MACHINE LEARNING

3.4.1 Aprendizaje No Supervisado

Con las técnicas del aprendizaje no supervisado se busca identificar estructuras en los datos de entrada, ya que no se dispone de una respuesta conocida a cada escenario de los datos de entrada, por lo tanto, se requiere que estos algoritmos busquen las características y relaciones entre los datos de entrada para crear el modelo. Se puede decir que el desafío de estos algoritmos consiste en descubrir las relaciones implícitas que pueden existir en un conjunto de datos que no está etiquetado, es decir que sus elementos no se han asignado previamente. Dentro de este conjunto de algoritmos de aprendizaje no supervisado se tienen tres tipos de problemas que son: Agrupamiento, Reducción de Dimensiones y Asociaciones.

3.4.1.1 Agrupamiento

Esta técnica también conocida en inglés como Clustering es la más utilizada del grupo de aprendizaje no supervisado. Se enfoca en el análisis exploratorio de datos con el objetivo de encontrar agrupaciones o patrones ocultos en los datos. Las máquinas con este tipo de algoritmos realizan la agrupación de los objetos o variables de acuerdo a su mejor criterio ya que al no estar etiquetados los datos, lo que hace es dividir los objetos en función de características desconocidas. La siguiente figura ilustra gráficamente el propósito de la técnica ante una serie de datos dada.

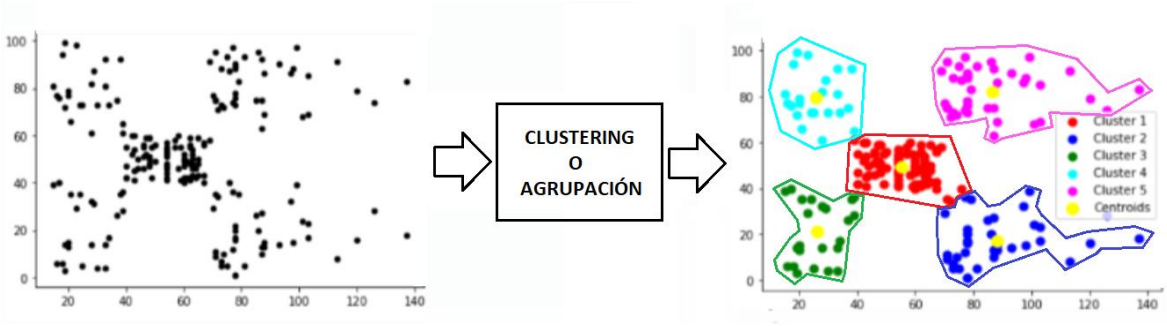


Figura No. 3-6. Técnica de Agrupación o Clustering. Fuente propia.

Estos algoritmos de agrupación durante el análisis de los datos de entrada intentaran hacer conjuntos de objetos que sean mucho más similares entre ellos que a los de otros grupos. De esta manera los objetos que tienen similares características se fusionan en una sola clase.

En el proceso de clustering se identifican varios pasos para generar los clusters, su nivel de dificultad es alto ya que la eficiencia del mismo dependerá de que tan precisos sean los criterios y reglas que se definan para el algoritmo. El primer paso para realizar el clustering consiste en definir el número de agrupaciones que se desean hacer del conjunto de datos. A continuación, se deben definir las formas de los grupos de similaridades junto con la asignación de un centro para tomarlo como punto de partida desde donde se pueda arrancar el recorte del agrupamiento. Asociado a estos parámetros se debe precisar también un margen de error para poder empezar a definir los diferentes clusters del conjunto de datos.

Estableciendo dentro del modelo una métrica o un nivel de error, se pueden delimitar los niveles aceptables de fallo que el algoritmo aplicará posteriormente en las formulas. Con esto se puede determinar un error general del modelo el cual deberá ser incorporado en el algoritmo de entrenamiento. Algunos de los algoritmos más habituales para realizar el clustering son: k-means y k-medoids, clustering jerárquico, modelos de mezclas gaussianas, modelos de Markov, mapas auto organizados y redes neuronales.

3.4.1.2 Reducción de Dimensiones

En el Machine Learning cobra gran importancia el número de dimensiones que se manejan en el modelo, las dimensiones se pueden equiparar al número de características o variables que se están considerando para el análisis. Lo anterior puesto que a medida que van aumentando el número de variables es decir el número de dimensiones, ocurre que las distancias se vuelven menos discriminativas. Este efecto es conocido como la "Maldición de la Dimensión" e impacta drásticamente a las técnicas de aprendizaje automático que están basadas en distancias tales como k-means y KNN (los k vecinos más cercanos). En muchos casos el hecho de ir aumentando el número de variables o características de los objetos puede favorecer para alcanzar una mayor precisión del modelo, pero se puede llegar a un cierto punto en el que el rendimiento del modelo disminuirá con el creciente aumento de características, lo anterior debido a que la densidad de la muestra

disminuye de manera exponencial con el aumento de la dimensionalidad. En otras palabras, cuando se siguen aumentando las variables o características sin que se realice un aumento en el número de muestras para el proceso de entrenamiento, se provocaría que la dimensionalidad del espacio de características crezca y se vuelva mucho más dispersa. Cuando esto ocurre es muy probable que se dé un sobreajuste, dado a que el modelo se correspondería en gran medida con un conjunto particular de datos y no se lograría una buena generalización. La siguiente imagen ilustra gráficamente un ejemplo sencillo de cómo se daría una reducción de dimensionalidad pasando de un espacio de tres dimensiones a un espacio de una sola dimensión.

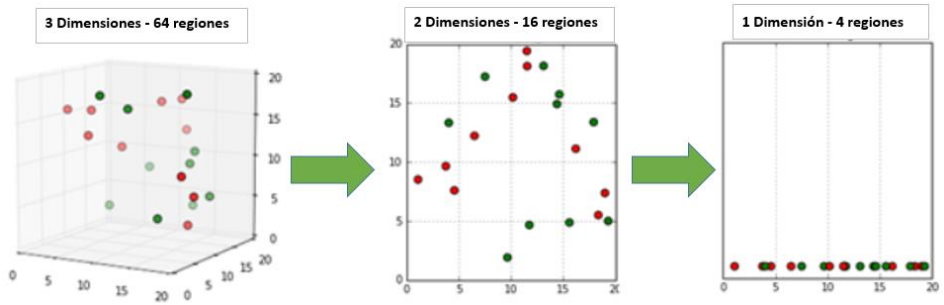


Figura No. 3-7. Reducción dimensiones 3D a 2D y 1D. Fuente propia.

Para poder enfrentar la maldición de la dimensionalidad y evitar el sobreajuste en los casos donde se tienen demasiadas características o variables de entrada con muy pocas muestras de entrenamiento, se han definido dos clases que son: la eliminación de características y la extracción de características.

- **Eliminación de características**: como su nombre lo indica, consiste en eliminar por completo algunas de las variables siempre y cuando se cumpla alguna de las siguientes dos condiciones: que estas sean redundantes con otra variable del conjunto de datos o que no aporten ninguna información nueva sobre el conjunto de datos. La ventaja de esta estrategia es que el conjunto de datos sea más pequeño solo con las variables de interés. Como desventaja se tendría la posibilidad de la perdida de información al dejar de evaluar las variables. En la siguiente imagen se ilustra un ejemplo con seis variables o características de entrada en el que después del proceso de selección de características se reducen a solo tres variables, las cuales serían las más representativas para el modelo. En otras palabras, gracias al proceso de reducción de dimensiones se estaría pasando de un espacio con seis dimensiones a uno de tres dimensiones.

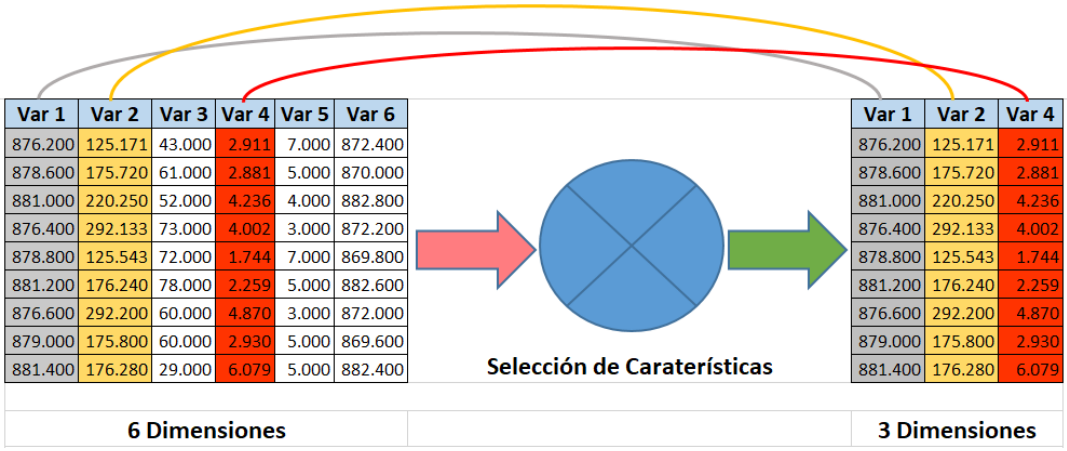


Figura No. 3-8. Clase selección de características. Fuente propia.

Es importante indicar que en estos métodos no realiza ninguna trasformación de la información de las variables originales, solo se seleccionan las variables o características que se consideran más relevantes de los objetos. Existen tres estrategias con las cuales se pueden llevar a cabo los métodos de selección de características: Filtrado, envolvente e incrustados.

- **Extracción de características**: su objetivo principal es el de poder proveer nuevas variables con un nivel más alto de información a partir de ciertas variables específicas. Al contrario de los métodos de la anterior clase, en estos métodos si se da una transformación completa de la información de las variables originales [16]. En esta técnica, si se parte de un total de 10 características, al final se crearán 10 nuevas características independientes y cada nueva característica será el resultado de la combinación de las 10 características iniciales. El proceso de creación de las nuevas variables se realizará de una manera tal que se puedan clasificar en un orden de mejor a peor de acuerdo a su relevancia para predecir la variable dependiente. Inicialmente se puede presentar que no existe una reducción de dimensiones dado a que se inicia y termina con el mismo número de variables, sin embargo, de ser necesario se puede prescindir de algunas de las nuevas características y al eliminarlas no se perderá toda la información inicial puesto que todas las nuevas variables contienen información útil de todas las variables iniciales. En la siguiente imagen se representa gráficamente el proceso de extracción de características, en donde se inicia con un espacio de seis variables y se termina con otro espacio de seis variables, pero clasificadas con un orden de importancia. Para este ejemplo puntual solo se consideran las tres primearas nuevas variables.

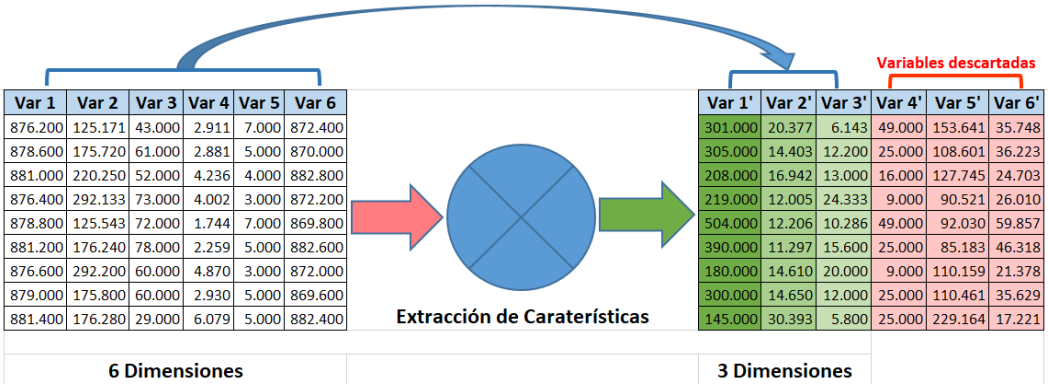


Figura No. 3-9. Clase extracción de características. Fuente propia.

Algunas de las tecinas más utilizadas para la reducción de dimensión mediante extracción de características son: Análisis de componentes principales (PCA), Análisis de correspondencia (AC), Análisis factorial(AF) o algunas variantes de estos.

3.4.1.3 Asociaciones

En muchas ocasiones los datos proporcionados a un modelo no tienen una relación directa entre sí, sin embargo, se pueden dar reglas de asociación entre las variables analizadas que busquen entre ellas un mismo objetivo. Estos algoritmos son muy comunes para realizar el análisis de los llamados carritos de compra en las ventas on-line con el objetivo de automatizar las estrategias de marketing. Para entender un poco mejor el propósito de estos algoritmos de asociación se presentan los siguientes ejemplos. Considérese que usted está planeando ir de vacaciones hacia un lugar costero, cuando inicia la busque de los tiquetes aéreos, de inmediato le aparecen ofertas de hoteles y planes turísticos. En este caso se está asociando el tiquete aéreo con opciones de hotel en el destino. Otro ejemplo común ocurre en el proceso de compra de vestuario, para este caso si un cliente está buscando comprar una sudadera para realizar deporte, en asociación a la sudadera le podrán aparecer otras ofertas como zapatillas y camisetas que pueden complementar la compra en conjunto con la sudadera. En la siguiente figura se ilustran algunos ejemplos de posibles asociaciones que normalmente se dan para un propósito final.

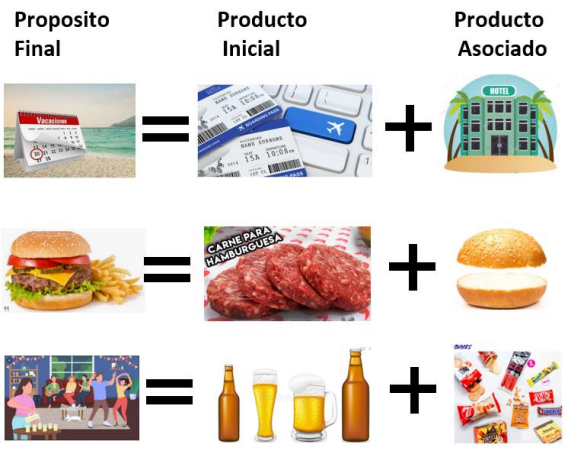


Figura No. 3-10. Ejemplos de Reglas de Asociación. Fuente propia.

Algunos de los algoritmos más populares que implementan estas técnicas son: A priori, Euclat, FP-growth.

3.4.2 Aprendizaje Supervisado

Estas técnicas serán utilizadas cuando se dispone de un conjunto de datos con información tanto de entrada como de salida, es decir que se conoce previamente cuál es la respuesta o resultado correcto para cada combinación de los datos de entrada. Por lo tanto, el problema que se tiene con estos casos es que no se sabe cómo llegar de los datos de entrada a la respuesta y para eso se requiere de un algoritmo. El objetivo de estas técnicas es el poder deducir una función basada en un proceso de entrenamiento en el que se consideran tanto los datos de entrada como las respuestas a cada uno de los escenarios. La siguiente expresión puede expresar en forma general el aprendizaje supervisado.

$$y = f(x)$$

Donde ***y*** corresponde a la variable resultado y *x* es el conjunto de información que se le introduce al modelo. Es importante mencionar que *x* puede ser un arreglo de variables. Dentro de este grupo de aprendizaje supervisado se tienen dos clases de problemas que son: Regresión y Clasificación.

3.4.2.1 Regresión

Los problemas de Regresión se dan cuando se intenta predecir una respuesta o resultado el cual puede variar en un rango continuo. Al final lo que se desea tener a la salida será un valor numérico dentro de un conjunto infinito de posibles resultados. Algunos ejemplos en los cuales se puede aplicar la regresión son:

1) Predecir el precio final de venta de un inmueble.
2) Estimar el tiempo de viaje que tardará un vehículo para llegar su destino.
3) Estimar la edad de una persona basados en una fotografía.
4) Predecir cuanto tiempo permanecerá un empleado en una compañía.

5) Predecir la temperatura a una hora determinada del día.

Como se ha mencionado antes, el aprendizaje supervisado hace referencia al aprendizaje computacional que se enfoca en el desarrollo de modelos que se soportan en el análisis de datos que son etiquetados por usuarios para posteriormente realizar predicciones a datos no vistos. Los modelos de regresión hacen uso de la función inferida para estimar los valores de salida dados unos datos de entrada.

Se parte de un conjunto de datos de entrenamiento.

$$D = \{(x_i, y_i): x_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}N_{i=1}$$
$$D = (x, y)$$

Dónde el conjunto *D* hace referencia al espacio de las características o variables de entrada, las $x_i \in \mathbb{R}^d$ son denominados como vectores de características y las $y_i \in \mathbb{R}$ corresponden a las respuestas.

Se asume que existe una función f y un posible ruido aleatorio ∈ tal que.

$$y_i = f(x_i) + \epsilon = f_i + \epsilon$$

Donde f: $\mathbb{R}^d \rightarrow \mathbb{R}$ es la función subyacente a los datos, y $\in$ hace referencia al posible ruido aleatorio que puede estar en los datos. El propósito aquí es poder construir un estimador asociado a la función subyacente.

$$\bar{f} \approx f$$

Algunos de los algoritmos más comunes con los cuales se pueden abordar esta técnica de Machine Learning son: regresión lineal simple y regresión lineal múltiple.

- **Regresión Lineal Simple**: dado un conjunto de puntos en un plano bidimensional, el algoritmo de regresión lineal establecerá un modelo para ajustar la relación de dependencia entre una sola característica independiente (valor de la variable de entrada *x*) y un solo valor de respuesta (valor de la variable dependiente *y*). Para establecer la relación se establece de manera arbitraria una línea recta y posteriormente se calcula la distancia entre la recta y cada uno de los puntos del conjunto (x_i, y_i). En esta distancia, las líneas verticales corresponden a lo que se conoce como residuo o error de predicción. La siguiente imagen ilustra un ejemplo de regresión lineal con una serie de puntos dispersos en el plano y para los cuales se ha definido una línea recta representada por la ecuación *y = mx +b*. Donde *y* es la variable dependiente, *m* es la pendiente de la recta, *x* es la variable dependiente y *b* es el intersecto con el eje y cuando *x* = 0. La expresión *mx + b* representa el modelo para que *y* sea el valor predicho.

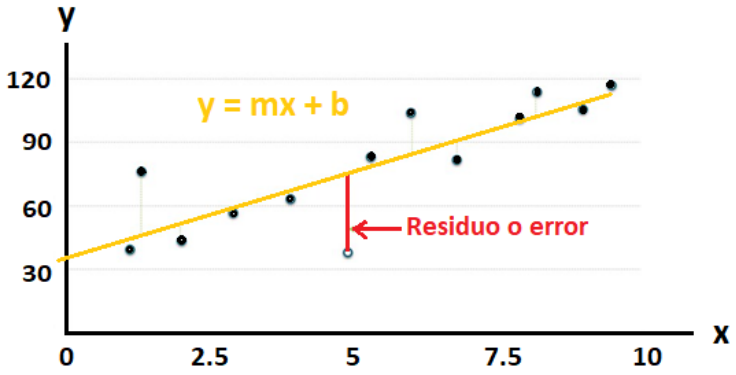


Figura No. 3-11. Regresión lineal simple – 2D. Fuente propia.

El objetivo que se persigue con la regresión lineal consiste en minimizar la distancia vertical que existe entre cada uno de los puntos del conjunto y la línea recta. El algoritmo recalculara la ecuación de la recta con cada iteración, es decir que la recta se desplazará en el plano hasta encontrar aquella que mejor se ajuste al conjunto de puntos dados. Para ello se debe minimizar la distancia entre todos los puntos y la línea recta, es decir conseguir la línea recta que sea más cercana a la mayor cantidad de puntos.

- **Regresión Lineal Múltiple**: este tipo de regresión se da cuando la salida o resultado que se intenta predecir depende de más de una variable de entrada. En otras palabras, para una variable dependiente existen varias variables independientes. Para esto será necesario definir un modelo más complejo que pueda realizar el análisis de un espacio con más dimensiones. Si en la regresión lineal simple se busca definir los parámetros para una línea recta mediante la ecuación $y = mx + b$, en la regresión lineal múltiple con dos variables independientes se pretende definir el modelo mediante el ajuste de un hiperplano. En este caso se tendría un espacio de tres dimensiones y la ecuación que las relaciona es $y = m_1x_1 + m_2x_2 + b$. La siguiente imagen ilustra un ejemplo de regresión lineal múltiple con dos variables de entrada independientes y una sola variable de salida dependiente.

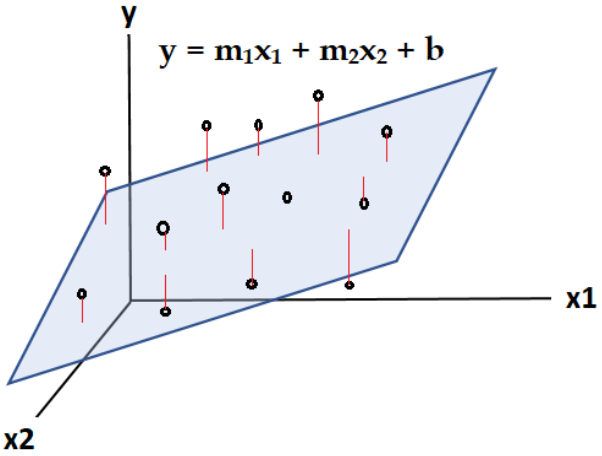


Figura No. 3-12. Regresión lineal múltiple - 3D. Fuente propia.

La visualización es sencilla cunado se tienen solo 2 o 3 dimensiones, pero a medida que se aumentas las dimensiones se hace más complicada su visualización.

3.4.2.2 Clasificación

Las tareas de clasificación se podrían considerar como procesos sencillos para los seres humanos, pues somos capaces de clasificar de una forma casi automática todo lo que vemos: como por ejemplo tipos de vehículos, animales, plantas, colores y hasta desechos o residuos entre otros. Por el contrario, este proceso que es tan sencillo para los humanos no lo es para las maquinas. Sin embargo, existen una serie de métodos que pueden ayudar en la generación de modelos para el desarrollo de estas tareas y que incluso podrían llegar a ser mucho más eficientes si se considera que las maquinas podrían analizar una mayor cantidad de información de forma continua sin descansar.

En las técnicas de regresión se intenta predecir una salida o respuesta continua, pero en las técnicas de clasificación se busca predecir unas respuestas discretas, es decir que, se puede utilizar las técnicas de clasificación si los datos pueden ser etiquetados, categorizados o divididos en grupos o clases concretas. Algunos ejemplos en los cuales se puede hacer uso de la clasificación son: en el diagnóstico de un tipo de cáncer, clasificándolo como maligno o benigno, en la clasificación de un correo como spam o normal o en la clasificación en los tipos de vinos.

Para expresarlo de una manera formal, se considera el conjunto de ejemplos provistos por el usuario para realizar el proceso de entrenamiento como $D=\{(x_1,y_1), (x_2,y_2),\ldots., (x_T,y_T)\}$, donde cada pareja (x_i,y_i) corresponde a un ejemplo debidamente etiquetado por el usuario (e.g., mensaje de correo electrónico y su correspondiente etiqueta), donde las variables $x_i \in \mathbb{R}^d$ y son la codificación numérica de la información de entrada del *i*-ésimo objeto, y_i corresponde a la clase en la cual fue clasificado el objeto, con $y_i \in C$ siendo $C=\{C_1, C_2, \ldots. C_T\}$ el conjunto de posibles clases en las cuales se pueden clasificar los objetos. Para el caso en el que K sea igual a dos (K=2) se tendría un problema de clasificación binaria con dos posibles casos de salida, es decir $y_i \in C = \{-1, +1\}$.

El conjunto *D* que relaciona las variables de entrada y sus respectivas respuestas o etiquetas, es usado para la generación del modelo, el cual puede venir de una distribución de probabilidad *P*(X,y) pero que es desconocida, puesto que si se conociera la distribución de probabilidad no se tendría necesidad de recurrir a las técnicas del Machine Learning. Es por lo anterior, que el problema de clasificación consiste en encontrar una función *f** que sea lo más aproximada posible a la función $f : \mathbb{R}^d \rightarrow C$, tal que $\hat{y}_j=f^*(x_j)$ sea igual o muy parecida a y_j, para todas las posibles combinaciones (x_i,y_i) que pertenezcan al conjunto *T*, considerando a *T* como el conjunto de todos los posibles escenarios que se podrían presentar dentro de la misma distribución *P*(X,y), es decir que se incluya dentro de este conjunto *T* tanto los casos etiquetados como los casos no etiquetados.

Algunos de los algoritmos que más se utilizan para la clasificación son: máquina de vectores de soporte (SVM-Support Vector Machine), clasificadores bayesianos (Naive Bayes), k-vecino más cercano (k-NN, k-Nearest Neighbors), Árboles de Decisión y Redes Neuronales Artificiales (RNA).

- **Máquina de Vectores de Soporte**: el principio de funcionamiento de este método consiste en definir un híper plano que divida los objetos en dos clases diferentes. Se parte de los objetos ejemplo que ya se encuentran debidamente etiquetados en un determinado plano de dimensión d, posterior se busca una línea dentro del espacio donde se encuentran los ejemplos de tal manera que divida los ejemplos en las diferentes clases a las que pertenecen. La siguiente imagen muestra un sencillo ejemplo en el que se busca clasificar los mensajes de correo electrónico en las clases spam y no spam.

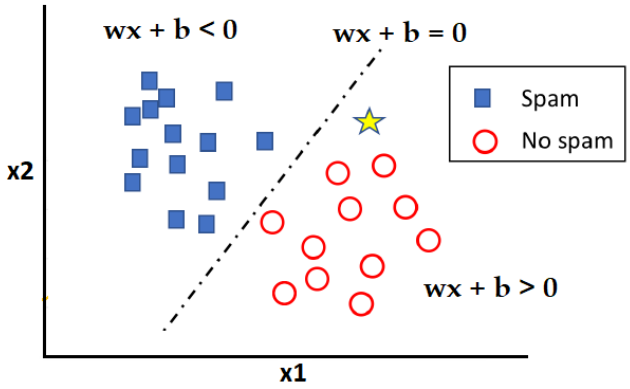


Figura No. 3-13. Clasificación lineal para mensajes de correo electrónico. Fuente [17].

Cuando la dimensión del espacio es dos (d=2), el hiperplano corresponde a una línea recta que está dada por $wx + b = 0$, donde $w \in \mathbb{R}^d$ y corresponde al vector de pesos, b es el termino de sesgo. Todas estas variables son aprendidas por el modelo gracias al conjunto de datos que ya están etiquetados. Para clasificar un nuevo objeto x_j que no hace parte del conjunto de entrenamiento, solo basta con determinar en qué lado del híper plano se encuentra. La clase en la cual se clasificaría el nuevo objeto x_j puede estar dada por la siguiente expresión [17].

$$f(x_j) = sign(wx_j + b)$$

Es decir que $f(x_i)$ puede tomar uno de tres valores -1, 0 o 1. El resultado -1 corresponde a la clase "Spam" y el 1 para la clase "No spam". Los ejemplos frontera más cercanos al híper plano de cada una de las clases corresponden a los denominados vectores soporte y coinciden con los extremos negativo y positivo del híper plano. La siguiente imagen nos ayuda a entender mejor lo anteriormente descrito.

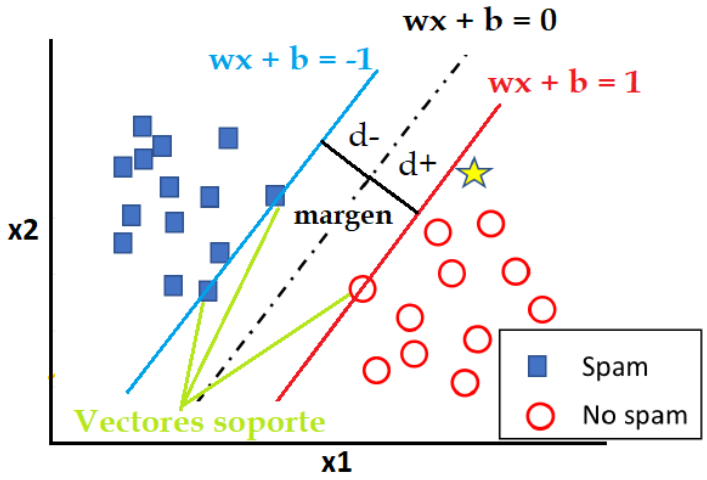


Figura No. 3-14. Clasificación con máquina de soporte vectorial. Fuente propia.

De la imagen se pueden identificar tres rectas, la recta azul corresponde al hiperplano frontera de la clase Spam y en ella se encuentran dos vectores soporte que son los dos ejemplos que coinciden con la recta, de igual manera se tiene una recta de color rojo que es el hiperplano frontera de la clase No spam, sobre esta recta solo se tiene un vector soporte, es decir que un solo ejemplo coincide con la recta. El problema que se busca en estos algoritmos es maximizar el margen de separación que existe entre los ejemplos de las dos clases. Si la distancia entre los dos extremos wx + b = 1 y wx + b = -1 es $\frac{2}{\|w\|}$, para lograr maximizarla lo que se debe hacer es minimizar ‖w‖.

- **Clasificadores Bayesianos:** estos clasificadores son probabilísticos y asumen independencia entre los diferentes atributos descriptivos de las variables predictoras en un conjunto dado. Este tipo de clasificadores tienen su fundamento en el teorema de Bayes junto con algunas hipótesis simplificadoras adicionales. En otras palabras, se puede decir que el modelo asume que la presencia o ausencia de un atributo particular no tiene relación con la presencia o ausencia de cualquier otro atributo o característica en la variable.

  El problema de clasificación supervisada desde el enfoque bayesiano consiste en asignarle a un objeto que tiene ciertas características o atributos X1, X2, X3, …, Xn, una de las m clases posibles, C1, C2, C3, …, Cm, de modo tal que la probabilidad de la clase sea máxima dados los atributos del objeto. Usando el teorema de Bayes lo anterior se puede presentar de una manera formal de la siguiente manera.

$$Arg_C[Max\ P(C|X_1, X_1, .., X_n)]$$

  Si se considera al conjunto de atributos del objeto como X = {X_1, X_2, X_3, …, X_n}, entonces la expresión se puede escribir como:

$$Arg_C[Max\ P(C|X)]$$

Como se puede ver a continuación, la formulación del clasificador bayesiano se soporta en la regla de Bayes para calcular la probabilidad posterior de la clase dados los atributos del objeto.

$$P(C|X_1, X_1, .., X_n) = \frac{P(C)P(X_1, X_1, .., X_n|C)}{P(X_1, X_1, .., X_n)}$$

O de forma compacta como:

$$P(C|X) = \frac{P(C)P(X|C)}{P(X)}$$

Por lo tanto, el problema de calificación que busca maximizar la probabilidad de la clase se puede representar de la siguiente manera.

$$Arg_C[Max\left[P(C|X) = \frac{P(C)P(X|C)}{P(X)}\right]]$$

En la practica el denominador *P(X)* se considera una constante, ya que no depende de C y los valores de Xi son datos. Adicional y dado que lo que nos interesa es maximizar la probabilidad de la clase, entonces se puede expresar como:

$$Arg_C[Max\,[P(C|X) = \alpha P(C)P(X|C)]]$$

Para poder resolver un problema de clasificación soportado en el enfoque bayesiano es necesario considerar la anterior expresión, donde se requiere la prioridad a priori de cada clase *P(C),* y la probabilidad de los atributos dada la clase, *P(X|C)*, también conocida como verosimilitud, con estos valores se puede conocer la probabilidad posterior *P(C|X).*

Con base en lo anterior se puede decir que para que el clasificador pueda aprender de un conjunto de datos, es necesario estimar las probabilidades a priori y verosimilitud, a partir de los datos, es decir que estas probabilidades serán los parámetros del clasificador.

Es importante indicar que, realizar la aplicación directa de la expresión ArgC[Max[P(C|X)=αP(C)P(X|C)]] implica un sistema muy complejo para desarrollarlo sobre una computadora puesto que el termino P(X1, X2, X3, …, Xn|C) se incrementa de manera exponencial de tamaño a medida que se aumentan el número de atributos, al final el resultado se tendría un requerimiento muy alto tanto de memoria para almacenar la información como de procesamiento para realizar un alto número de operaciones en el cálculo de la probabilidad [17].

- **K-vecinos más cercanos:** este es uno de los métodos de clasificación más sencillos y comprensibles. Su funcionamiento se basa en la comparación del objeto que se desea clasificar con los demás objetos que hacen parte del conjunto de datos de entrenamiento. De manera intuitiva lo que hace para clasificar un nuevo objeto *xj* en una determinada clase, es analizar las etiquetas de todos los k ejemplos que se utilizaron para el entrenamiento y que más se parecen al nuevo objeto desconocido *xj*. Continuando con el ejemplo de clasificación de correos spam, un nuevo mensaje de correo electrónico será comparado con los almacenados en el set de

entrenamiento, como resultado de las comparaciones se seleccionarán los k mensajes que tengan las características o atributos más parecidos, al final la clase o etiqueta asignada será la que más se repita de los mensajes seleccionados en el rango de comparación.

La siguiente figura ilustra de manera gráfica como se da el proceso de clasificación utilizando el método de k-NN con k=3. En este caso el objeto a clasificar es la estrella de color amarillo que por sus características es ubicado en dicha posición dentro del plano, ahora dentro del circulo concéntrico en la estrella se encuentran tres vecinos que son los más cercanos al objeto que se desea clasificar, de estos tres vecinos, dos pertenecen a la clase "No spam" y solo un vecino esta categorizado como "Spam". Por mayoría el nuevo objeto se clasificaría como No spam.

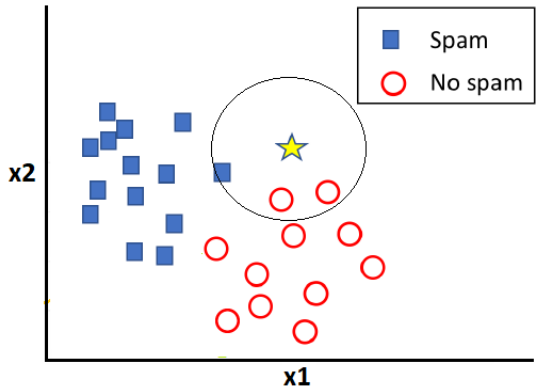


Figura No. 3-15. Clasificación k-NN con k=3. Fuente propia.

Como se puede observar la "*k*" significa la cantidad de vecinos más cercanos que se deben considerar para realizar la clasificación. Se recomienda escoger valores impares para *k*, dado que si es par y los vecinos más cercanos están repartidos en cantidades iguales de las clases a clasificar entonces se daría un empate.

Se dice que este algoritmo está basado en instancia dado a que no aprende explícitamente un modelo como ocurre con otros algoritmos, sino que memoriza todas las instancias de entrenamiento para usarlas como base de conocimiento para poder realizar la predicción.

De manera formal para la clasificación de un nuevo objeto *xj,* se puede representar la función de decisión del algoritmo k-NN mediante la siguiente expresión.

$$f(x_j) = arg^{max}_{C_k \in C} \sum_{i=1}^{k} 1_{v=f(x_i^N)}$$

Donde los $x_1^N, \ldots, x_k^N$ corresponden a los k vecinos que son más cercanos al objeto *xj.* Y $1_{a=b}$ =1 si a=b o 0 en caso contrario. Para conocer cuáles son los vecinos más cercanos al objeto bajo análisis *xj*, se recurre al cálculo de la distancia entre el objeto que se quiere clasificar y todos los elementos del conjunto de entrenamiento que ya se encuentran etiquetados. Los *k* objetos que se encuentren a menor distancia vienen a ser los vecinos más cercanos. Para el cálculo de la distancia existen varios métodos, la elección del método dependerá del tipo de problema. El

método para el cálculo de la distancia que más se utiliza corresponde a la forma Euclidiana la cual se declara mediante la siguiente expresión.

$$d(x_j, x_i) = \sqrt{\sum_{i=1}^{k} (x_j - x_i)^2}$$

Otras formas de calcular la distancia entre los diferentes objetos son "Manhattan" y "Minkowski" y su uso dependerá de las condiciones del problema y aplicación.

Distancia Manhattan:

$$d(x_j, x_i) = \sum_{i=1}^{k} |x_j - x_i|$$

Distancia Minkowski:

$$d(x_j, x_i) = [\sum_{i=1}^{k} (|x_j - x_i|)^4]^{\frac{1}{4}}$$

En resumen, se puede decir que el proceso de clasificación tiene tres pasos que son:

1. Calcular la distancia
2. Encontrar sus vecinos más cercanos
3. Asignar la clase basado en las mayorías

Es importante precisar que el usuario es quien debe ajustar el valor de *k* en *k-NN*, los valores altos de *k* son habituales en problemas con clases linealmente separables, por el contrario, los valores pequeños de *k* son recomendable para problemas con clases donde la superficie de clasificación no es lineal [17].

- **Árboles de Decisión:** estas técnicas son unas de las más antiguas y utilizadas en el Machine Learning, son ampliamente usadas en los procesos de clasificación gracias a su arquitectura que facilita su interpretación e implementación. Los árboles de decisión se encuentran dentro del grupo de los modelos predictivos, y para estos modelos en especial se basan en reglas binarias de decisión (si y no) con las cuales se logra distribuir las observaciones en función de las características de cada una de las variables independientes para predecir de esta manera la variable dependiente o categoría resultado.

  La estructura básica de un árbol de decisión se basa como su nombre lo indica en un árbol con ramas y hojas. Dentro del modelos se definen los siguientes tipos de nodos:

  - Nodo raíz: es el primer nodo del árbol y es por lo tanto donde se produce la primera división en función de la variable más importante.

- Nodos intermedios o internos: estos nodos aparecen después del nodo raíz y se utilizan para seguir dividiendo el conjunto de datos en función de las variables. Estos nodos aparecerán hasta que ya no sea necesario tomar decisiones de división.

- Nodos hojas o terminales: son los últimos nodos que aparecen en las ramificaciones y su función es indicar la clasificación final que se tiene para el objeto evaluado.

La siguiente imagen ilustra de manera general la estructura de un árbol de decisión con todos sus nodos. Como se puede apreciar en este ejemplo, cada condición evaluada en los nodos raíz e intermedios genera una división de los datos, mirado en la imagen de la derecha, con cada condición se generan nuevas regiones, al final el número de regiones concuerda con el número de nodos hoja que tenga el árbol. Otra observación importante que se puede destacar, es que las diferentes clases en las cuales se van a categorizar los objetos, están repartidas en varias de las regiones en las que se han divido los datos.

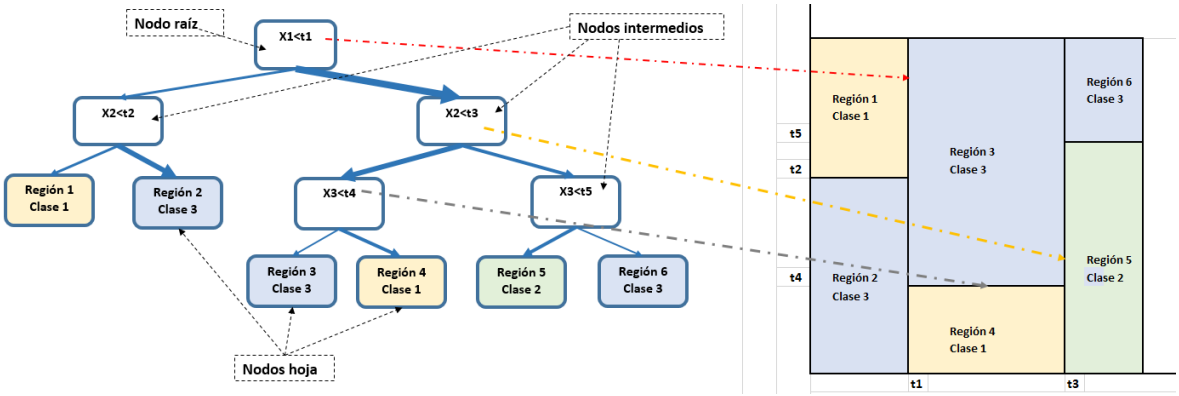


Figura No. 3-16. Árbol de decisión para clasificación. Fuente propia.

Otro concepto que se debe tener presente en estos modelos es la **profundidad del árbol,** la cual se define por el número máximo de nodos que puede llegar a tener una rama. La lectura del árbol debe realizarse de arriba hacia abajo.

Para la construcción de un árbol de decisión es importante la elección de las variables que se deben dividir primero, esto con el objetivo de obtener una separación óptima. Pues se obtendrán resultados diferentes si se cambia el orden en que se dividen las variables. Para entender un poco mejor este punto considérese un árbol de decisión que tiene un conjunto de registros con dos atributos como género con valores “hombre ó mujer” y edad en rangos inferiores y superiores o iguales a 18 años. Se podría crear un árbol que primero divida por género y luego subdivida por edad. La otra opción es que sea primero por edad y luego por género. La tarea de escoger la mejor opción se la dejamos a los algoritmos, los cuales mediante el análisis de los datos de entrada junto con los datos de salida decidirán la mejor forma de realizar las divisiones en cada uno de los nodos para que al final se logre la predicción con la mayor probabilidad de acierto. A primera instancia podría parecer sencillo, pero la verdad es que no lo es, si se tuviera un conjunto con 10 atributos y cada uno de ellos con 2 o más valores,

las posibles combinaciones para seleccionar el mejor árbol serían de cientos o miles. Es por lo anterior que aparecen estos algoritmos los cuales tomaran la mejor decisión desde un punto de vista probabilístico.

El funcionamiento detrás del algoritmo consiste en medir de algún modo las predicciones que se pueden alcanzar con cada posible opción, esto con el objetivo de compararlas y definir la mejor opción. Para esta tarea los algoritmos disponen de varias alternativas tales como: el índice de Gini o la ganancia de información.

- **Índice de Gini:** este método mide el grado de impureza del nodo. En otras palabras, se puede decir que indica que tan desordenados o mezclados quedan los nodos una vez divididos. Entre mayor sea el índice Gini mayor será la impureza del nodo por lo que se buscara seleccionar la variable con menor Gine ponderado. La solución al problema será minimizar el índice Gini.

  El índice de Gini se define como: la métrica con la cual se puede medir la frecuencia con la que un objeto escogido al azar seria clasificado incorrectamente y se utiliza la siguiente formula.

$$GINI_m = \sum_{k=1}^{K} Pmk(1 - Pmk) = \sum_{k=1}^{K} Pmk - Pmk^2 = \sum_{k=1}^{K} Pmk - \sum_{k=1}^{K} Pmk^2 = 1 - \sum_{k=1}^{K} (Pmk)^2$$

  Donde *Pmk* corresponde a la probabilidad de cada clase en el nodo m, K es el número de categorías o clases.

- **Ganancia de Información**: este cálculo de ganancia de información se asocia a la propiedad estadística que mide que tan efectivo es un atributo para separar los ejemplos de entrenamiento basados en sus clasificaciones objetivo. En otras palabras, lo que busca esta forma de clasificación es estimar la información que aporta cada atributo y para esto se recurre al cálculo de la "entropía", la cual va de cero a uno. La medida de la entropía es útil para cuantificar el desorden de un sistema, cuando un nodo es puro se dice que su entropía es cero y por lo tanto se tendrán observaciones para una sola clase. Por el contrario, si la entropía es uno, entonces se tiene la misma frecuencia de objetos para cada una de las clases en observación. Se debe maximizar esta ganancia.

  Cuando se obtiene la medida de entropía para cada atributo, se puede calcular la ganancia de información del árbol. Para el cálculo de la entropía se utiliza la siguiente función.

$$E = -\sum_{k=1}^{K} Pmk * \log_2(Pmk)$$

  La ganancia de la información se da como resultado de la diferencia entre la entropía antes de la división, es decir en el nodo padre, y la entropía promedio que se tiene

después de que se ha realizado la división del conjunto de datos en función de los valores de atributo dados.

Ganancia de información = Entropía Nodo Padre – Promedio Entropía nodos hijos.

- **Redes Neuronales Artificiales:** se pueden definir como un modelo matemático que está inspirado en el comportamiento biológico que presentan las neuronas. Y no solo considera el funcionamiento de una neurona, sino también, en cómo se organizan y comunican dentro de la estructura del cerebro. En cuanto al funcionamiento, el cerebro puede ser visto como un sistema inteligente capaz de desarrollar un gran número de tareas, pero de una forma distinta a como lo hace actualmente las conmutadoras.

  Si bien en el cerebro existen varios tipos de neuronas, la siguiente imagen ilustra un esquema simplificado de una neurona típica. En ella se pueden identificar varias partes que son comunes a todas las neuronas.

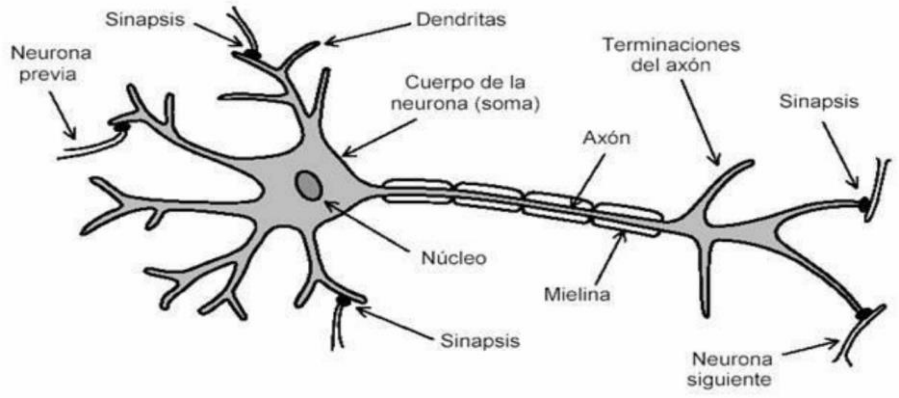


Figura No. 3-17. Partes de la Neurona. Fuente [18]

El cuerpo central de la neurona se llama **soma**, este contiene en su interior al **núcleo,** adicional se identifican una prolongación larga del soma que se llama **axón**. Al conjunto de ramificaciones se les denomina **dendritas**, y cuando estas se conectan a otras neuronas se da lo que se conoce como **sinapsis.**

La función principal de una neurona es la transmisión de impulsos nerviosos, estos impulsos viajan desde las dendritas hasta llegar a las últimas terminaciones del axón, donde por medio de la sinapsis pasa a otra neurona.

La forma en cómo se responde a los diferentes estímulos del mundo exterior y el aprendizaje que se realiza del mismo, está muy relacionado con las conexiones neuronales que se dan en el cerebro. De aquí es que aparecen las RNA como un intento de emular este proceso.

El siguiente modelo neuronal corresponde al primer modelo matemático que se creó para una neurona artificial. Este modelo fue presentado en 1943 como resultado del trabajo conjunto del psiquiatra y neuroanatomista Warren McCulloch y el matemático Walter Pitts [19].

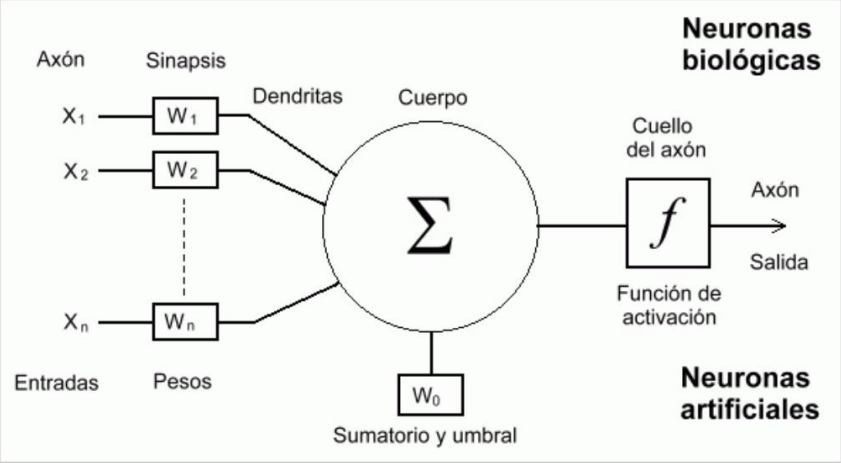


Figura No. 3-18. Modelo neuronal de McCulloch-Pitts. Fuente [19]

Dentro del modelo se identifican n entradas $X_1, X_2, \ldots, X_n$. A cada entrada se le asocia un peso sináptico $W_1, W_2, \ldots, W_n$. Dentro del cuerpo se tiene la función de agregación, ∑. Antes de la salida, sobre el cuello del axón se tiene una función de activación, *f*.

Los estímulos son representados por las entradas que recibe la neurona artificial, y la respuesta es la salida de la neurona. Lo magnifico de las neuronas es que pueden adaptarse a su entorno y aprender de él cuando se modifican los valores de los pesos sinápticos. Estos pesos también son conocidos como parámetros libres del modelo, debido a que pueden ser modificados y adaptados para que se pueda cumplir una determinada tarea.

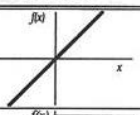
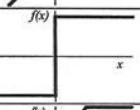
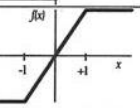
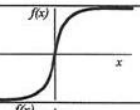
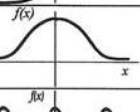
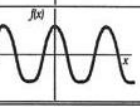
Para este modelo la salida Y de la neurona está dada por la siguiente expresión:

$$Y = f\left[\left(\sum_{i=1}^{n} W_i X_i\right) + W_0\right]$$

Donde *Xi* corresponde a las entradas de la neurona, *Wi* son los pesos asociados a cada una de las entradas y por último W_0 representa el bias (sesgo) o umbral que sirve para controlar que tan dispuesta esta la neurona para asignar un 1 o un 0 de forma independiente de los pesos. Un bias alto provoca que la neurona demande una entrada alta.

Tanto en las redes neuronales biológicas como en las artificiales se da un paso importante en la transmisión de la información, para el caso de la neurona biológica se la conoce como tasa de potencial de acción de disparo en el cerebro y para la neurona artificial se tiene lo que se conoce como función de activación. Esta función de activación se escoge de acuerdo a la tarea objetivo que tiene que realizar la neurona. La siguiente tabla relaciona las funciones de activación más empleadas en el campo de las redes neuronales [19].

Tabla No. 3-1. Funciones de activación más comunes en RNA. Fuente [20]

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

Al igual como ocurre en las conexiones de las neuronas en el cerebro, en las redes neuronales artificiales también se interconectan entre ellas para formar una red que potencializa la capacidad para la solución de problemas. En la siguiente imagen se puede observar como la salida de la neurona pre-sináptica i se conecta a una de las entradas de la siguiente neurona post-sináptica.

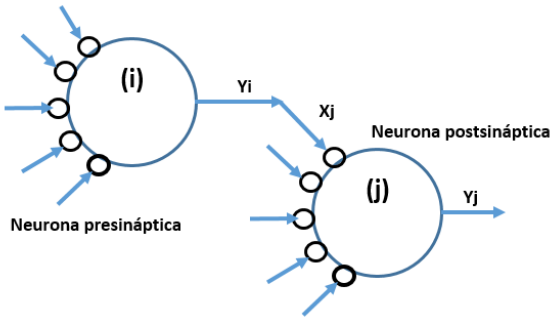


Figura No. 3-19. Interconexión entre neuronas artificiales. Fuente propia.

Gracias a estas interconexiones se pueden formar redes neuronales multicapa. La siguiente figura ilustra un ejemplo típico de estos arreglos de neuronas artificiales. En la red se identifican tres tipos de capas, la capa de entrada, las capas ocultas y la capa de salida. La capa de entrada se encarga de recibir los valores o señales de entrada y distribuirlos a los elementos de procesamiento de la siguiente capa. En la capa oculta se encarga de sopesar cada una de las entradas para aprender las diferentes características de los datos. La capa de salida es la responsable de proporcionar la información de las etiquetas asociadas a las diferentes clases.

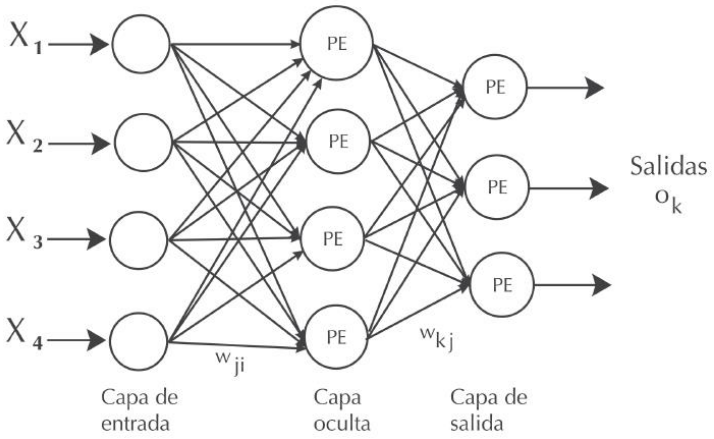


Figura No. 3-20. Red neuronal multi-capa. Fuente propia.

Para los problemas de clasificación utilizando redes neuronales se pensaría en primera instancia que una sola neurona de salida podría ser configurada con varios rangos de valores que correspondieran a las diferentes categorías. Pero esto sería bastante complejo ya que el funcionamiento normal de una neurona se enfocaría a una clasificación binaria que sería útil solo para clasificar dos clases o categorías. Pero si el problema demanda la categorización de tres o más clases se recomienda adaptar la red neuronal para que cada neurona de salida corresponda con una determinada categoría, de esta manera si en el problema se deben clasificar los elementos en 6 categorías será necesario configura la red en la capa de salida con 6 neuronas.

3.4.3 Resumen técnicas de Machine Learning

Con el objetivo de facilitar la escogencia de alguna técnica de Machine Learning para la resolución de algún problema, a continuación, se presenta una tabla en la que se resumen las principales características con algunos de los algoritmos más utilizados.

Tabla No. 3-2. Características técnicas de Machine Learning. Fuente propia.

Machine Learning				
Aprendisaje Supervisado		**Aprendizaje No Supervisado**		
Regresión	**Clasificación**	**Asociaciones**	**Reducción Dimensiones**	**Clustering**
-Relacionado con modelos predictivos. -El dataset de entrenamiento debe contener datos de entrada y salida. -La maquina aprende explicitamente de los datos de ejemplos. -La salida debe estar dentro de un rango de datos continuos. -Su objetivo es predecir un valor númerico dentro de un rango continuo. -Algoritmos: Redes Neuronales, Árboles de decisión, SVR, GPR, Regresión Lineal.	-Entrega modelos predictivos. -El dataset de entrenamiendo debe contener datos de entrada y salida. -La maquina aprende explicitamente de los datos de ejemplos. -La salida debe estar dentro de un rango de opciones discretas o categorias. -Su objetivo es determinar una categoria o clase para cada una de las observaciones. -Algoritmos: Árboles de decisión, SVM, Redes Neureonales, Naive Bayes	-El dataset solo contiene los datos de entrada. -La salida debe estar dentro de un rango de opciones discretas. -Su objetivo es encontrar items que esten involucrados en las transaciones de un determinado conjunto. -Se establecen reglas de asociación entre los items de un determinado conjunto de datos. -Algoritmos usados: Apriori, FP-Growth, Euclat.	-El dataset solo contiene los datos de entrada. -Se parte de un conjunto de datos de espacion M y se mapean a un subconjunto de espacio N derivado del espacio original pero de menor dimensión. -Su objetivo es el de reducir el número de variables aleatorias de entrada. -Se seleccionan las variables o caracteristicas más relevantes. -Algoritmos: ACP, AC, AF.	-El dataset solo contiene los datos de entrada. -Su objetivo es agrupar los objetos en subgrupos de tal manera que sus caracteristicas sean muy similares entre si. -Al no tener conocimiento de las salidas o respuestas se realiza un analisis exploratorio de los datos para encontrar agrupaciones o patrones ocultos. -Algoritmos:K-meas y K-modoids, mezclas gaussianas, Markov, redes neuronales.

4. DESARROLLO MODELOS DE MACHINE LEARNING

4.1 ANÁLISIS DATA SET CELDAS UMTS

Conocidas ya las principales técnicas de Machine Learning presentadas en el capítulo anterior, se puede proceder ahora a enmarcar el problema de optimización de celdas UMTS en alguna de las técnicas que mejor se acomoden a las características del mismo. Para esto es necesario analizar la estructura del dataset con el cual se dispone para resolver el problema.

A continuación, en la siguiente tabla se presenta una muestra del raw data estadístico que se puede obtener de los gestores de desempeño iManager U2020-Huawei y del cual se puede armar el dataset con todas las variables necesarias para entrenar el modelo. Este raw data corresponde a la información atomizada de una red 3G-UMTS de uno de los operadores móviles en Colombia. Cada celda UMTS está asociada a un sector, un sitio, una ciudad, una región y finalmente una RNC.

Tabla No. 4-1. Raw data celdas UMTS. Fuente propia.

RNC	CLUSTER	carrier type	SITE	WCDMA_CELL	SECTOR	Cell Load	% P_Traffic R99	Available codes to HSDPA	RxQualAchievable Achievable	Voice Erlangs	Active_HSPS_Users_#	TTI Load
RNCMED0	Noroccidente	ANT	ANT1085	ANT1085:0	1	1019.16%	146.7%	9.4906	15	3.6416	16.7668	99.60%
RNCIBA01	Centro	IBA	IBA0014	IBA0014:7	2	430.37%	71.8%	9.8598	18	2.2166	9.968	94.49%
RNCBOG01	Centro	BOG	BOG0028	BOG0028:14	3	117.53%	47.1%	6.501	20	11.6284	3.4392	91.94%
RNCBOG04	Centro	IBA	IBA0018	IBA0018:0	1	261.78%	71.1%	6.2936	18	4.3668	6.0632	96.64%
RNCIBA01	Centro	IBA	IBA0014	IBA0014:1	2	218.66%	66.3%	9.484	21	2.4534	7.1262	94.83%
RNCMED02	Noroccidente	ANT	ANT1049	ANT1049:1	2	142.28%	41.5%	10.69918	19	4.3416	3.73764	94.81%
RNCBOG01	Centro	BOG	BOG0028	BOG0028:8	3	143.26%	41.4%	7.9992	20	14.3934	4.1922	95.02%
RNCMED02	Noroccidente	ANT	ANT0010	ANT0010:0	1	209.28%	59.9%	8.8328	16	9.8984	3.6998	94.94%
RNCBOG07	Centro	CUN	CUN0017	CUN0017:9	4	214.42%	57.8%	8.074	20	6.8998	6.2746	96.85%
RNCBOG07	Centro	CUN	CUN0450	CUN0450:0	1	101.00%	57.6%	11.2768	20	1.8616	2.9554	87.73%
RNCCAL01	Suroccidente	VAC	VAC0120	VAC0120:0	1	21.79%	36.4%	12.132	23	4.1952	1.054	62.66%
RNCBOG07	Centro	CUN	CUN0115	CUN0115:6	1	73.51%	54.7%	11.7524	17	1.3234	1.5284	74.24%
RNCSIN01	Costa	COR	COR0022	COR0022:0	1	73.63%	54.2%	8.44564	18	1.54368	1.70538	61.93%
RNCMED02	Noroccidente	MED	MED1082	MED1082:1	2	38.07%	35.2%	12.6386	19	0.9232	0.6152	44.92%

Para cada celda UMTS el gestor de red entrega una gran cantidad de contadores estadísticos con los cuales se pueden conformar varios KPIs que son útiles para definir el estado de la celda. Del lado derecho de la anterior tabla y resaltados en color amarillo se encuentran los KPIs que se han seleccionado para realizar el diagnóstico y que por lo tanto se tomaran como variables de entrada

para el entrenamiento del modelo, estor KPIs son los mismos que se explicaron en el capítulo inicial, es decir el Cell load, porcentaje de potencia para trafico Release 99, códigos disponibles para HSDPA, calidad alcanzable en recepción, tráfico de voz en Erlangs, usuarios activos de HSPS (High Speed Packet Swtich) y carga de TTI.

Personal experto en optimización de redes UMTS toma todos estos KPIs y realizan un proceso de análisis de forma manual para cada una de las celdas, en otras palabras, esto corresponde a un diagnóstico de celda con el cual los expertos determinan el estado de la celda. La siguiente tabla muestra en la columna final resaltada en color verde algunos de los ejemplos de diagnóstico que se pueden dar para las diferentes celdas, entre ellos se encuentran las que tiene un estado normal etiquetadas como OK, las que presentan deficiencias de cobertura etiquetadas como COV, las que tiene deficiencias de capacidad etiquetadas como CAP y finalmente las que tienen problemas de hardware o configuración que son etiquetadas como CHECK.

Tabla No. 4-2. Raw data celdas UMTS con Diagnóstico. Fuente propia.

RNC	CLUSTER	carrier type	SITE	WCDMA_CELL	SECTOR	Cell Load	% P_Traffic R99	Available codes to HSDPA	RxQualAchievable Achievable	Voice Erlangs	Active_HSPS_Users_#	TTI Load	Diagnostico
RNCMED0	Noroccidente	ANT	ANT1085	ANT1085:0	1	1019.16%	146.7%	9.4906	15	3.6416	16.7668	99.60%	COV
RNCIBA01	Centro	IBA	IBA0014	IBA0014:7	2	430.37%	71.8%	9.8598	18	2.2166	9.968	94.49%	COV
RNCBOG01	Centro	BOG	BOG0028	BOG0028:14	3	117.53%	47.1%	6.501	20	11.6284	3.4392	91.94%	CHECK
RNCBOG04	Centro	IBA	IBA0018	IBA0018:0	1	261.78%	71.1%	6.2936	18	4.3668	6.0632	96.64%	COV
RNCIBA01	Centro	IBA	IBA0014	IBA0014:1	2	218.66%	66.3%	9.484	21	2.4534	7.1262	94.83%	COV
RNCMED02	Noroccidente	ANT	ANT1049	ANT1049:1	2	142.28%	41.5%	10.69918	19	4.3416	3.73764	94.81%	CAP
RNCBOG01	Centro	BOG	BOG0028	BOG0028:8	3	143.26%	41.4%	7.9992	20	14.3934	4.1922	95.02%	CHECK
RNCMED02	Noroccidente	ANT	ANT0010	ANT0010:0	1	209.28%	59.9%	8.8328	16	9.8984	3.6998	94.94%	COV
RNCBOG07	Centro	CUN	CUN0017	CUN0017:9	4	214.42%	57.8%	8.074	20	6.8998	6.2746	96.85%	CAP
RNCBOG07	Centro	CUN	CUN0450	CUN0450:0	1	101.00%	57.6%	11.2768	20	1.8616	2.9554	87.73%	CAP
RNCCAL01	Suroccidente	VAC	VAC0120	VAC0120:0	1	21.79%	36.4%	12.132	23	4.1952	1.054	62.66%	OK
RNCBOG07	Centro	CUN	CUN0115	CUN0115:6	1	73.51%	54.7%	11.7524	17	1.3234	1.5284	74.24%	OK
RNCSIN01	Costa	COR	COR0022	COR0022:0	1	73.63%	54.2%	8.44564	18	1.54368	1.70538	61.93%	OK
RNCMED02	Noroccidente	MED	MED1082	MED1082:1	2	38.07%	35.2%	12.6386	19	0.9232	0.6152	44.92%	OK

De esta manera la información del raw data con los siete KPIs y su correspondiente diagnóstico será utilizada como data set para el entrenamiento del modelo. Como se puede ver, el data set de entrenamiento estará compuesto por siete variables de entrada que son los KPIs y una variable de salida que corresponde al diagnóstico. A continuación, se relacionan diferentes características que tiene el data set y que serán útiles para escoger la técnica de Machine Learning más adecuada.

1. El data set dispone de datos tanto de entrada como de salida.

2. Las variables de entrada corresponden a valores continuos en diferentes rangos.

3. Los datos de salida son categorías que se pueden asociar a diferentes clases.

Dado a que el data set dispone de datos tanto de entrada como de salida, la técnica de Machine Learning a escoger se debe encontrar dentro del grupo de aprendizaje supervisado, por otro lado y considerando que los datos de salida no son continuos sino que son categorizados, nos lleva a que se debe utilizar una de las técnicas del sub grupo de clasificación.

Para el desarrollo de este proyecto de investigación se utilizarán los algoritmos de Árboles de Decisión y Redes Neuronales que son comúnmente utilizadas para resolver problemas de clasificación.

La siguiente figura ilustra la distribución de los 28.091 registros de las celdas UMTS que componen el data set en cada una de las clases en las cuales se pueden clasificar.

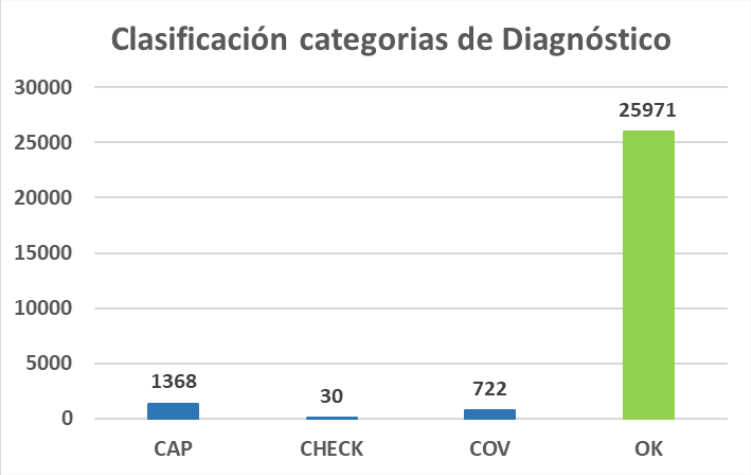


Figura No. 4-1. Clasificación total de los registros del dataset. Fuente propia.

Como se puede observar, más del 92% de las muestras (celdas UMTS) de ejemplo son categorizadas o clasificadas como OK, el resto de muestras se reparten en las tres posibles anomalías que puede presentar una celda. En otras palabras, se puede decir que un poco menos del 8% de las celdas UMTS presentan un diagnóstico negativo que demandará algún plan de acción dentro del proceso de optimización. En esta distribución de los ejemplos se puede ver que existe un desbalance entre los registros de ejemplos con diagnóstico negativo (CAP, CHECK, COV) y los de diagnóstico positivo (OK).

Con el objetivo de facilitar el proceso de aprendizaje se divide el gran problema de diagnóstico de celdas UMTS en dos problemas más pequeños. El primer problema consistirá en buscar un modelo que solo clasifique las muestras en dos categorías OK y NOK, donde esta última abarca las clases de diagnóstico negativo asociado a CAP, COV y CHECK. La siguiente figura muestra la distribución que se tiene para cada una de las clases OK y NOK en el primer problema.

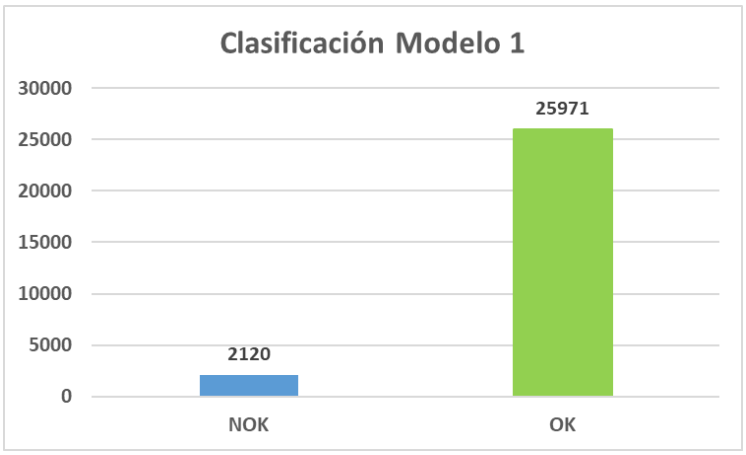


Figura No. 4-2. Clasificación de registros Modelo 1. Fuente propia.

El segundo problema se encargará de encontrar un modelo solo para clasificar las muestras de la categoría NOK en las tres posibles clases de diagnóstico negativo. Para el entrenamiento solo se contará con un data set de 2120 muestras. De esta manera se elimina en gran medida el desbalance que se tiene entre las muestras de las celdas que se encuentran bien y las que requieren un plan de acción por tener un diagnostico negativo. La siguiente figura ilustra la distribución de las muestras que pertenecen a la clase NOK del primer modelo y que se clasifican como CAP, COV y CHECK para el segundo problema.

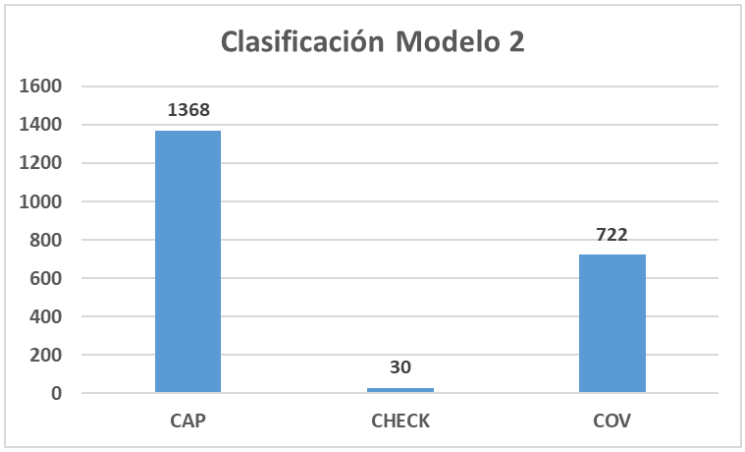


Figura No. 4-3. Clasificación de registros Modelo 2. Fuente propia.

La siguiente figura ilustra cómo se acota el problema desde una clasificación con cuatro clases (CAP, CHECK, COV y OK) pasando a un primer modelo con solo dos clases (OK y NOK) y terminando con un segundo modelo que solo se encarga de categorizar los registros en tres clases (CAP, CHECK y COV).

Clase	# Muestras
CAP	1368
CHECK	30
COV	722
OK	25971
Total	28091

Modelo 1

Clase	# Muestras
NOK	2120
OK	25971
Total	28091

Modelo 2

Clase	# Muestras
CAP	1368
CHECK	30
COV	722
Total	2120

Figura No. 4-4. Modelos de Clasificación. Fuente propia.

4.2 TÉCNICA ALGORITMO ÁRBOL DE DECISIÓN

4.2.1 Desarrollo modelo 1 – Árbol de decisión.

Este primer algoritmo es uno de los más utilizados en el Machine Learning para la resolución de problemas complejos de clasificación. Para abordar el desarrollo de los modelos basados en árboles de decisión se recurre a la aplicación de Matlab llamada "Classification Learner" la cual permite entrenar un modelo usando Machine Learning supervisado. La siguiente figura muestra la ubicación en Matlab de esta APP.

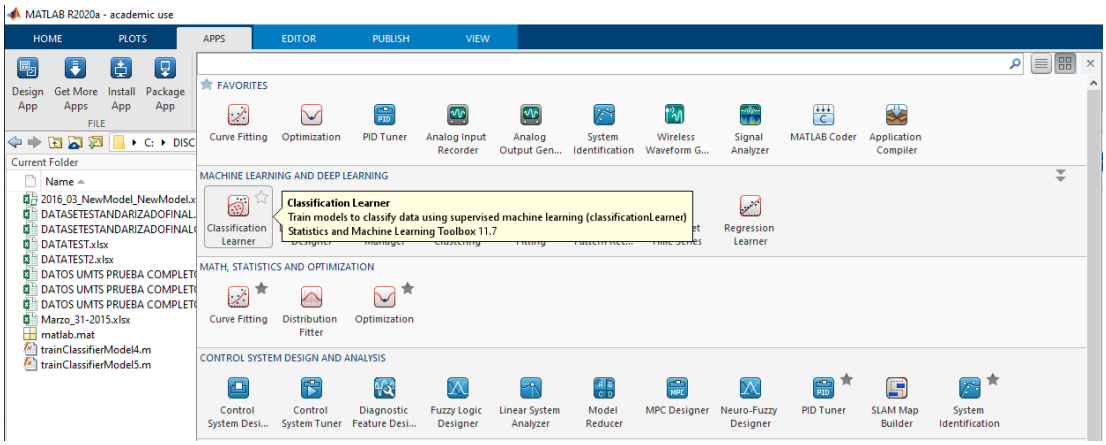


Figura No. 4-5. APP Classification Learner. Fuente propia.

Antes de iniciar la APP de clasificación se procede a importar a Matlab la tabla de datos del data set que se armó con los KPIs de las celdas UMTS y el diagnostico dado por los expertos en optimización para cada una de ellas. La siguiente figura ilustra el data set que se importó, y el cual se muestra como una matriz de 28091x8 nombrada como DTM1.

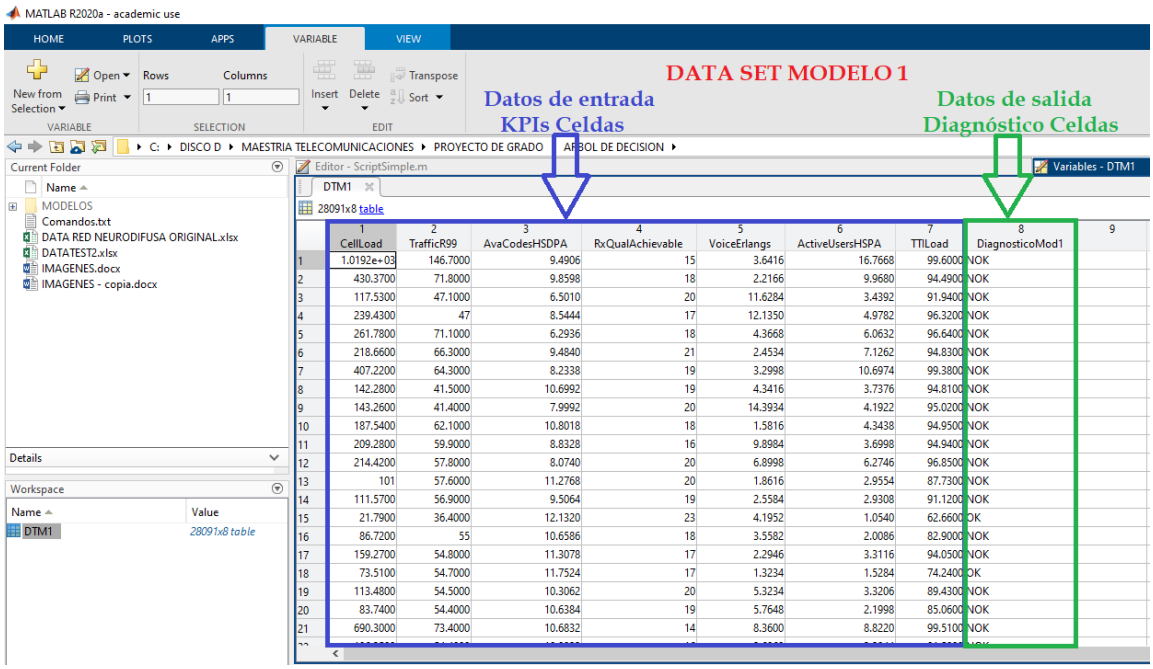


Figura No. 4-6. Importe Data Set Modelo 1. Fuente propia.

Una vez importado el data set se inicia una nueva sesión de la APP "Classification Learner". A continuación, en la siguiente figura se resalta cada uno de los parámetros que deben ser fijados para el entrenamiento del modelo. En primera instancia se selecciona el data set que previamente se importó, esta opción se remarca en color verde. A continuación, se define de donde se toman los datos de salida, para nuestro caso se toman del mismo data set en la columna final llamada "DiagnósticoMod1". En este punto la APP ya identifica el número de categorías en la cuales se deben clasificar cada uno de los registros. Como las posibles respuestas son OK y NOK, serán dos únicas clases. La variable de salida o respuesta también se conoce como variable dependiente. Finalmente se deben seleccionar las variables predictoras con las cuales se van a entrenar el modelo, es decir, se deben seleccionar todas las variables de entrada que se ubicaron en las columnas 1 a 7 en el data set.

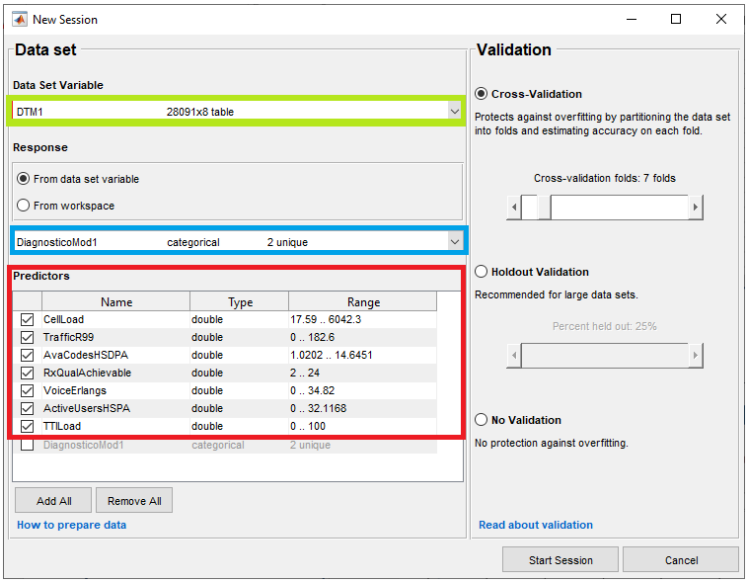


Figura No. 4-7. Configuración APP Classification Learner – Modelo 1. Fuente propia.

Dentro de los árboles de decisión existen varias alternativas las cuales se enmarcan en la siguiente figura con color rojo, los tipos de árboles se diferencian por la cantidad de hojas y ramificaciones o divisiones que pueden tener, con el objetivo de tener varias distinciones finas se decide seleccionar la opción “Fine Tree” que ofrece un máximo de 100 divisiones.

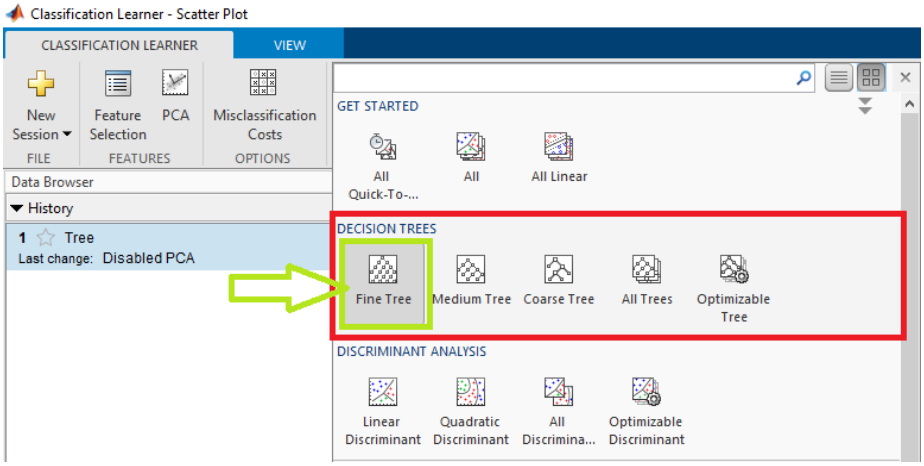


Figura No. 4-8. Opciones árboles de decisión. Fuente propia.

Una vez cargado el data set DTM1 con los datos de entrada y salida, y habiendo seleccionado el tipo de árbol, se procede a realizar el proceso de entrenamiento tal como se muestra en la siguiente figura.

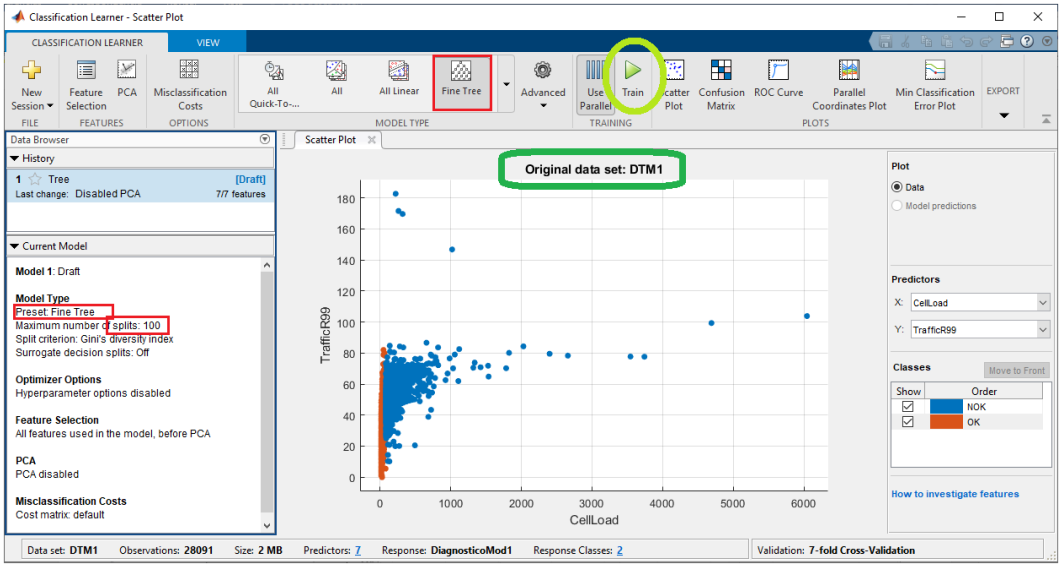


Figura No. 4-9. Entrenamiento "Fine Tree" – Modelo 1. Fuente propia.

A continuación, se presentan los resultados del entrenamiento con su correspondiente análisis. En primer lugar, se puede observar del resultado de entrenamiento que la precisión que tendrá el modelo1 es muy cercana al 100%, en toras palabras, se puede decir que será muy preciso en la valoración de las siete variables de entrada para definir la categoría correcta. Por otro lado, se puede observar que la velocidad de predicción es de aproximadamente 400.000 observaciones por segundo, y que para analizar los 28.091 registros solo gasto un tiempo de 5,0103 segundos, lo cual se puede considerar como bastante rápido.

Una forma amigable que ofrece Matlab para el análisis grafico de un set de datos complejos tanto por cantidad de registros como por cantidad de variables, es la función "Parallel Coordinates Plot", la cual crea una gráfica de coordenadas paralelas de todos los datos con sus múltiples características. Sobre el eje X se presentan cada una de las variables de entrada, que para nuestro caso corresponden a los siete KPIs de celda UMTS, y sobre el eje Y se presenta la desviación estándar que tiene cada registro para cada variable o característica de entrada. Al final se tendrán graficadas todas las coordenadas armadas por característica y desviación estándar.

De la siguiente figura se puede observar que los registros clasificados como OK tienen una menor dispersión que los registros clasificados como NOK, lo cual indica que el modelo funcionará mucho mejor para esta categoría OK. Al validar la dispersión de los datos en todas sus variables o características se puede ver que la que está asociada al KPI Cell Load tiene la menor dispersión de todas para los registros categorizados como OK. Lo anterior nos podría llevar a inferir que esta es una de las variables más puras y que por lo tanto será la variable que se asigne al nodo de partida en el árbol de decisión.

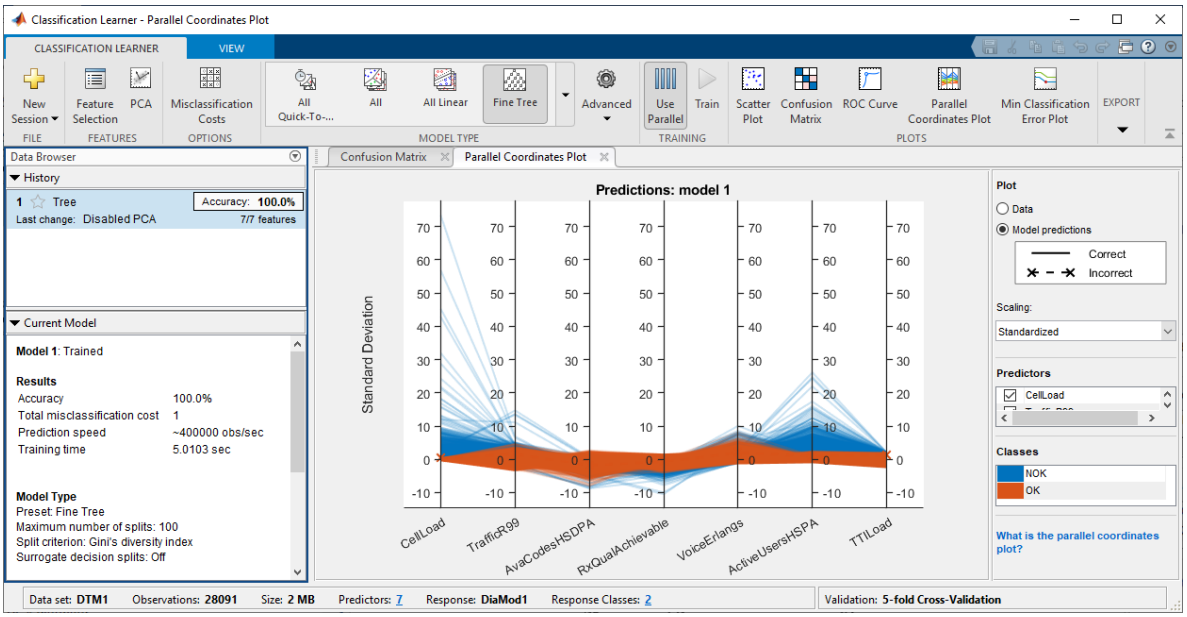


Figura No. 4-10. Desviación estándar – Variables de entrada Modelo 1. Fuente propia.

Respecto de los registros clasificados como NOK se puede observar de la figura anterior que tiene una mayor dispersión en los datos, en especial para las variables de entrada Cell Load y ActiveUsersHSPA. Por su mayor dispersión es posible que el modelo presente algún error en el momento de la predicción.

Otra función que nos ayuda a evaluar el rendimiento del modelo durante la fase de aprendizaje es la función AUC (Area Under the Curve) la cual en términos de probabilidad define que tan bueno es el modelo para la predicción de dicha categoría y las curvas ROC (Receiver Operating Characteristic) que muestra la relación entre la tasa lograda por el modelo clasificador entre los verdaderos positivos y los falsos positivos conocidos como TPR (True Positive Rate) y FPR (False Positive Rate) respectivamente. En las gráficas entregadas por la aplicación de Matlab se mezclan en un solo grafico tanto la función de las curvas ROC como la función de AUC, uno por cada categoría o clase que predijo el modelo durante el proceso de aprendizaje. Las siguientes dos figuras muestras estas funciones, la primera para la clase OK y la segunda para la clase NOK.

Para las dos categorías OK y NOK se tienen similares resultados en el cálculo de la AUC con un valor muy cercano a 1, esto quiere decir que existe una muy alta probabilidad de que un nuevo registro sea clasificado correctamente como un verdadero OK o un verdadero NOK. Por otro lado, y dado a que el AUC es 1 se puede inferir que las clases OK y NOK están debidamente separadas sin traslape entre ellas.

Por su parte las curvas ROC tienen de igual manera un resultado similar en las dos categorías el cual es muy cercano a 1. Esto quiere decir que se tiene un 100% de tasa para verdaderos positivos y 0% en la tasa de falsos positivos.

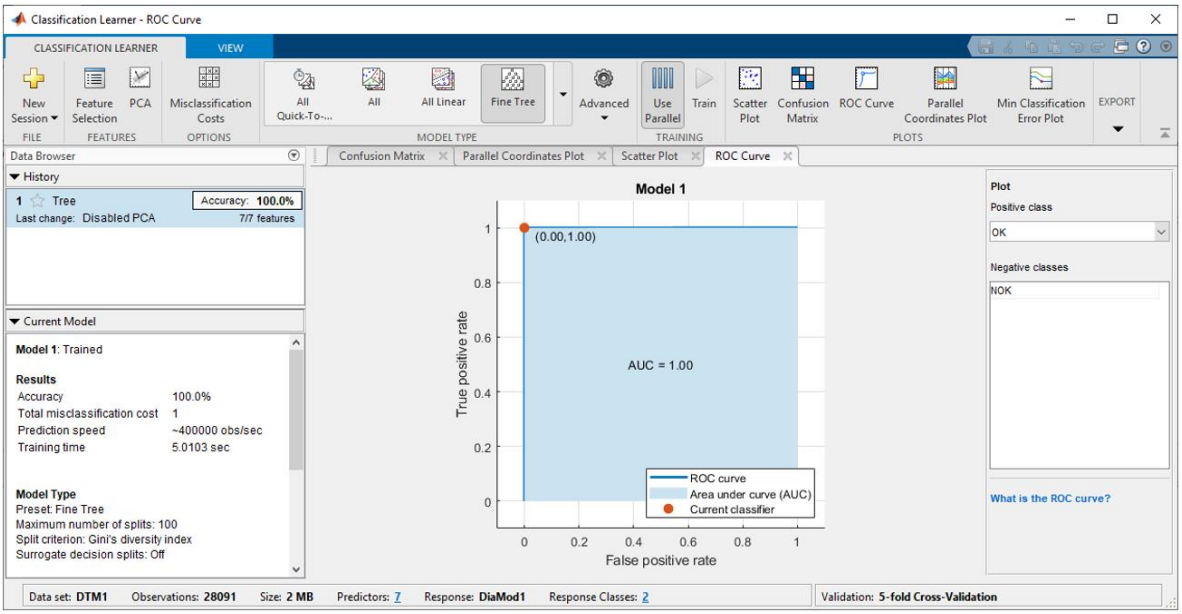


Figura No. 4-11. AUC y curvas ROC Clase OK. Fuente propia.

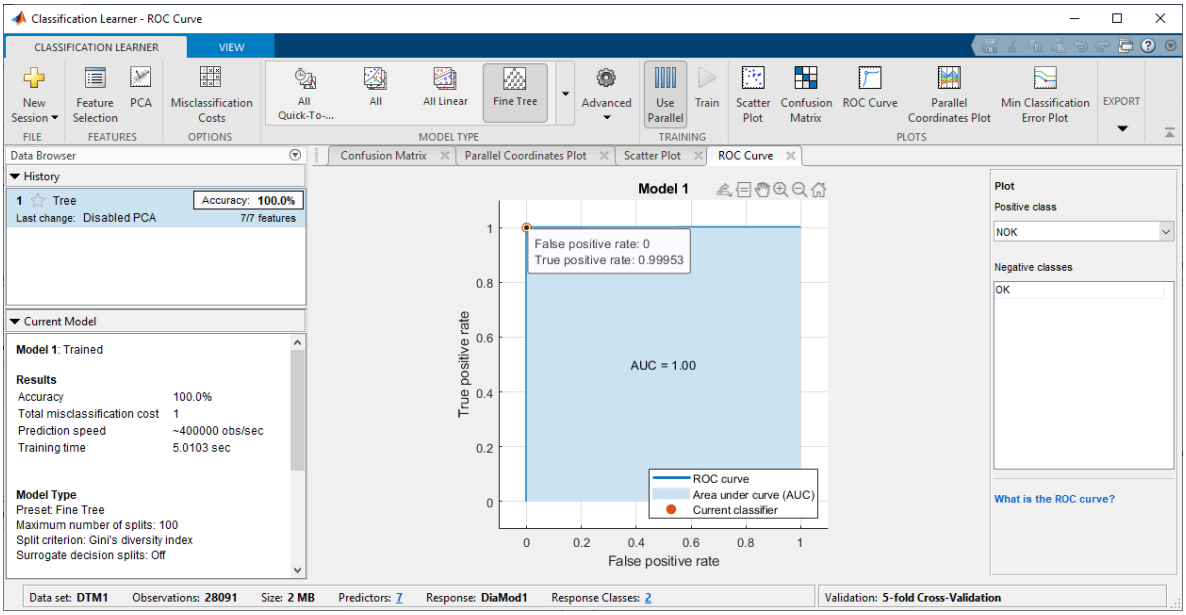


Figura No. 4-12. AUC y curvas ROC Clase NOK. Fuente propia.

4.2.1.1 Métricas matrices de confusión – Técnica Árbol de Decisión Modelo 1

La siguiente función de la cual se hará uso para el análisis de los resultados en cada modelo es la llamada matriz de confusión. Esta función es muy útil ya que además de facilitar el análisis de todas las categorías de manera gráfica, también proporciona los datos numéricos necesarios para el

cálculo de las diferentes métricas que califican o evalúan el desempeño del algoritmo en el proceso de entrenamiento.

Una matriz de confusión para un problema de clase n es una matriz nxn donde n es el número de clases o categorías que el modelo debe predecir. Las filas se nombran según las clases reales y las columnas de acuerdo a las clases que predijo el modelo. En otras palabras, se puede decir que por medio de la matriz de confusión se puede ver cuáles son los aciertos y errores que está presentado el modelo durante el proceso de aprendizaje.

Se tiene tres formas de representación de la matriz de confusión. La primera en forma numérica entre las clases verdaderas y clases predichas. En la segunda y tercera forma los resultados son expresados en forma de porcentajes considerando la tasa de verdaderos positivos (TPR), tasa de falsos negativos (FNR), valor predictivo positivo (PPV) y la tasa de falso descubrimientos (FDR).

Para el cálculo de las diferentes métricas que nos ayudaran a evaluar el desempeño del modelo es necesario definir las expresiones de cálculo para cada una de ellas. La siguiente tabla muestra los términos que la conforman con su correspondiente ubicación.

Tabla No. 4-3. Matriz de Confusión. Fuente propia.

Clase Real	**P**	**TP**	**FN**
	N	**FP**	**TN**
		P´	**N´**
		Clase Predicha	

Donde,

N→ es la condición negativa, está dado por el número real del total de casos negativos.

P→ es la condición positiva, está dado por el número real del total de casos positivos.

N'→ es la predicción negativa, está dado por el número total de casos negativos que predijo el algoritmo.

P'→ es la predicción positiva, está dado por el número total de casos positivos que predijo el algoritmo.

TN→ son los verdaderos negativos, corresponden al número de casos que siendo negativos se clasificaron correctamente como negativos.

FP→ son los falsos positivos, corresponden al número de casos que siendo negativos se clasifican erróneamente como positivos. También son conocidos como error I o subestimación.

FN→ son los falsos negativos, corresponden al número de casos que siendo positivos se clasifican erróneamente como negativos. También son conocidos como error II o sobrestimación.

TP→ son los verdaderos positivos, corresponden al número de casos que siendo positivos se clasifican correctamente como positivos.

Una vez definidas las anteriores convenciones, a continuación, se presentan las diferentes expresiones que definen las métricas de las matrices de confusión.

- **Exactitud:** útil para determinar qué porcentaje del total de datos fue clasificado correctamente. Se calcula mediante la siguiente expresión.

$$Exactitud = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Tasa de error:** indica que porcentaje del total de datos se clasifico incorrectamente. Para su cálculo se utiliza la siguiente expresión.

$$Tasa\ de\ error = \frac{FP + FN}{P + N} = \frac{FP + FN}{TP + TN + FP + FN}$$

- **Sensibilidad:** también conocida como tasa de verdaderos positivos (TRP), cuando la clase es positiva, que porcentaje se logra clasificar correctamente. Se calcula mediante la siguiente expresión.

$$Sensibilidad = \frac{TP}{P} = \frac{TP}{TP + FN}$$

- **Especificidad:** también conocida como tasa de verdaderos negativos (TNR). Se trata de los casos negativos clasificados correctamente por el algoritmo del total de verdaderos negativos. Se calcula mediante la siguiente expresión.

$$Especificidad = \frac{TN}{N} = \frac{TN}{TN + FP}$$

- **Precisión:** también conocida como valor predictivo positivo (PPV). Se refiere a la dispersión del conjunto de predicciones positivas obtenidas por el algoritmo. Cuanto menor sea la dispersión mayor será la precisión. Se calcula mediante la siguiente expresión.

$$Precisión = \frac{TP}{P'} = \frac{TP}{TP + FP}$$

La siguiente figura muestra el resultado de la matriz de confusión para el primer modelo del algoritmo árbol de decisión.

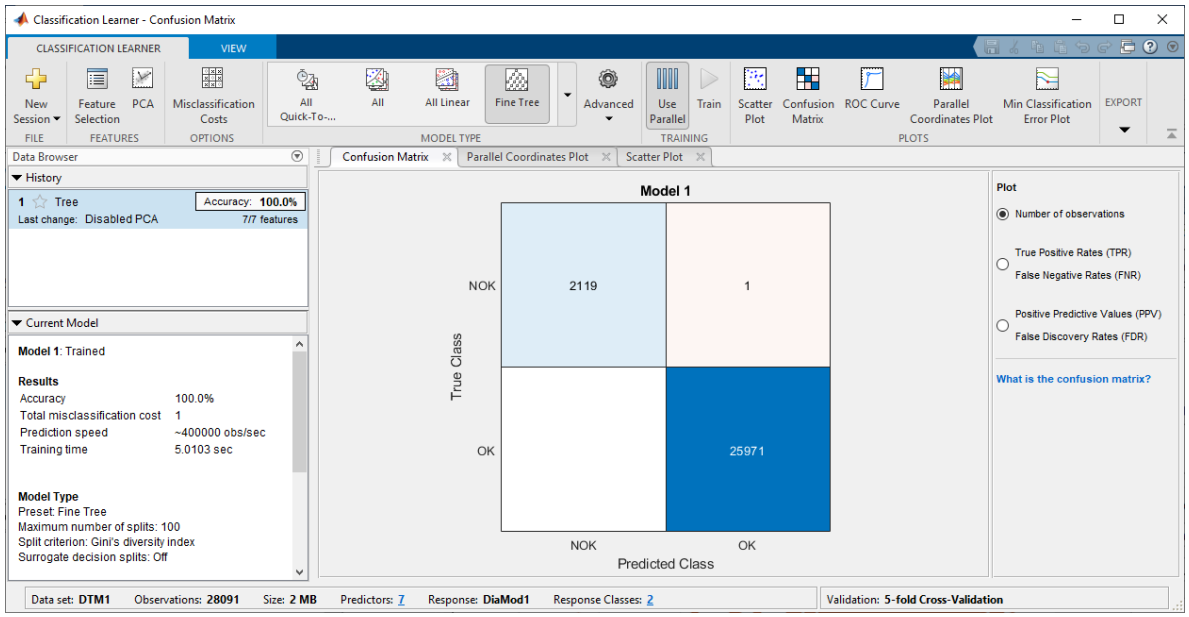


Figura No. 4-13. Matriz de Confusión (Número de observaciones)– Modelo 1. Fuente propia.

A continuación, se realizan los cálculos de todas las métricas que nos permiten evaluar el modelo.

$$\boldsymbol{Exactitud} = \frac{2119 + 25971}{28091} = \frac{28090}{28091} = 0.999964$$

El modelo ofrece una muy buena exactitud ya que su valor es muy cercano al 100%, con esto se tiene que los resultados de la clasificación de los registros están muy cerca del valor verdadero para cada una de sus clases OK y NOK. En otras palabras, se puede decir que prácticamente no existe diferencia entre el valor real y el valor esperado.

$$\boldsymbol{Tasa\ de\ error} = \frac{1 + 0}{28091} = \frac{1}{28091} = 0.000035598$$

La tasa de error es demasiado baja con un valor muy cercano a 0%, por lo tanto, se puede decir que el modelo tiene un muy bajo porcentaje de registros clasificados incorrectamente considerando todas las clases OK y NOK.

$$\boldsymbol{Sensibilidad} = \frac{2119}{2119 + 1} = \frac{2119}{2120} = 0.999528$$

En relación a la sensibilidad, se puede decir que el algoritmo es bastante alto ya que se logra porcentaje cercano al 100%. Por lo tanto, el modelo tiene una alta capacidad para discriminar los casos de celdas UMTS con problemas que se deben clasificar como NOK.

$$\boldsymbol{Especificidad} = \frac{25971}{25971 + 0} = \frac{25971}{25971} = 1$$

Por su parte la especificidad logra un porcentaje perfecto del 100%, con esto se puede decir que el modelo tiene una muy alta capacidad para identificar las celdas UMTS que no tienen problema entre todas las celdas que están OK.

$$\boldsymbol{Precisión} = \frac{2119}{2119 + 0} = \frac{2119}{2119} = 1$$

La precisión que ofrece el modelo es muy alta, con un porcentaje perfecto del 100%, es decir, que ninguno de los registros que realmente pertenecen a la clase OK se clasificaron como NOK.

Las siguientes figuras entregadas por la aplicación de Matlab ilustran gráficamente los cálculos de sensibilidad y precisión respectivamente. Con estas graficas se pueden corroborar los cálculos realizados para cada métrica.

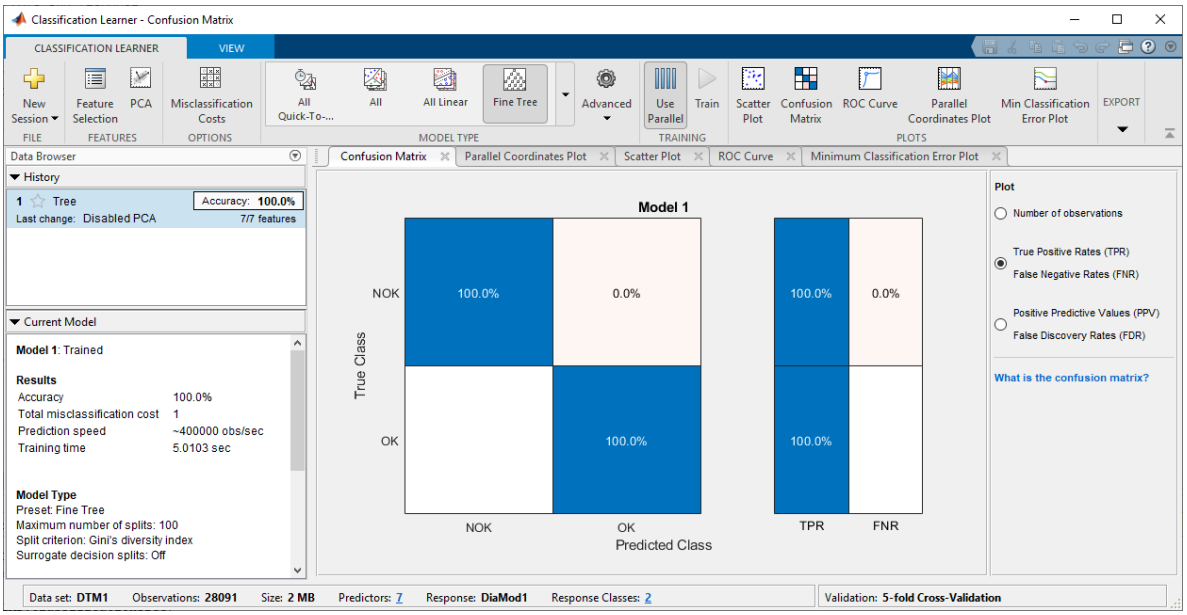


Figura No. 4-14. Matriz de Confusión (TPR/FNR)– Sensibilidad Modelo 1. Fuente propia.

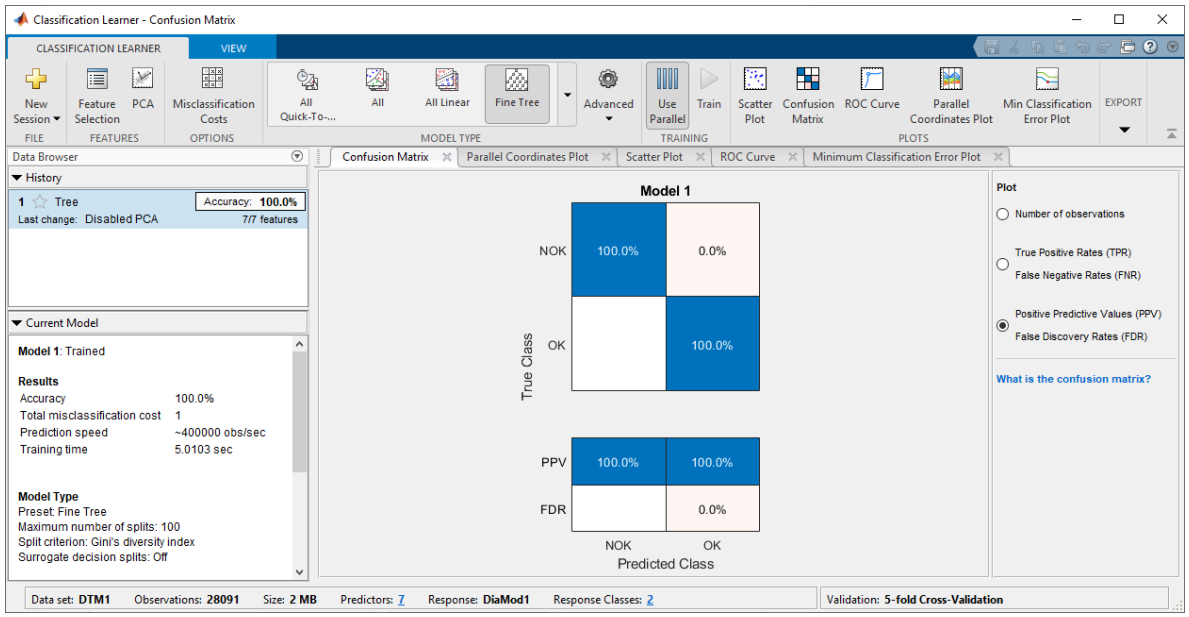


Figura No. 4-15. Matriz de Confusión (PPV/FDR) – Precisión Modelo 1. Fuente propia.

4.2.1.2 Resultado árbol de decisión modelo 1

Al final cuando el proceso de entrenamiento ha terminado, se puede graficar el árbol de decisión que el algoritmo tendrá en cuenta para clasificar los registros en las dos posibles categorías del modelo 1 que son OK y NOK. Como se puede ver en la siguiente figura, el algoritmo determina que para realizar el proceso de clasificación solo considera la primera variable del data set que es el CellLoad. De esto se puede concluir que en el análisis que realiza el algoritmo con todas las posibles opciones de árboles de decisión, la más efectiva para la tarea de clasificación es esta opción.

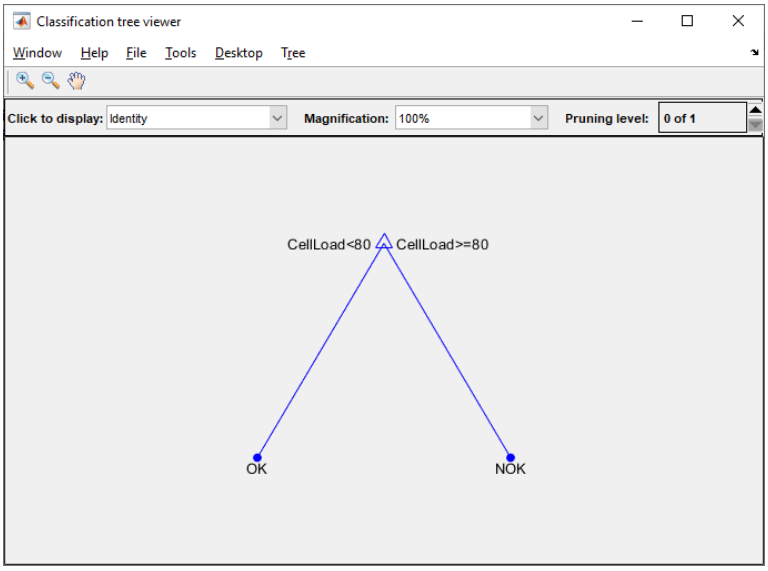


Figura No. 4-16. Árbol de decisión Modelo 1. Fuente propia.

4.2.2 Desarrollo modelo 2 – Árbol de decisión.

Igual como se realizó con el modelo 1, primero se procede a importar a Matlab el data set con el cual se realizará el proceso de entrenamiento, pero para este modelo el data set se reduce ya que solo se consideran los registros de las celdas que se pertenecen realmente a la categoría NOK, dentro de esta gran categoría se encuentran las categorías COV, CAP y CHECK. A continuación, en la siguiente figura se ilustra el data set cargado en Matlab como una matriz de 2120x8 llamado DTM2. Es importante indicar que para el entrenamiento de este segundo modelo también se utilizan las mismas siete variables que se utilizaron en el modelo 1, lo que cambia además del número de registros de ejemplo es que las salidas se clasifican en tres categorías, por lo tanto y como se verá más adelante, la matriz de confusión será de 3x3.

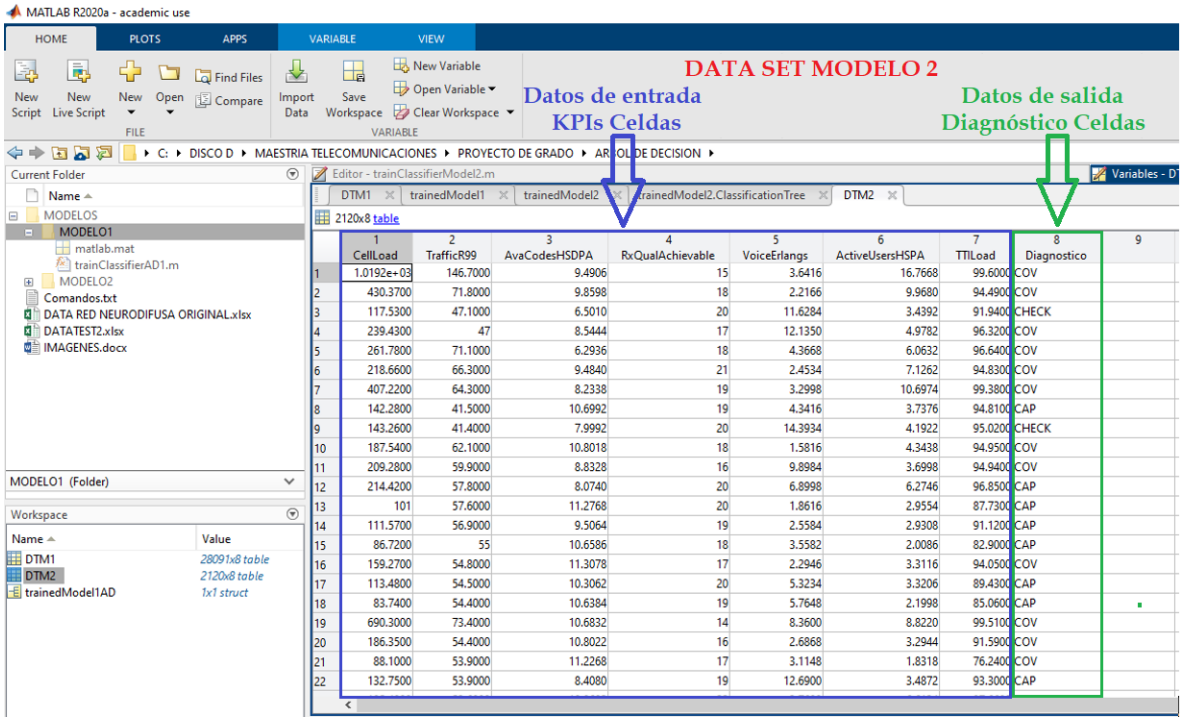


Figura No. 4-17. Importe Data Set Modelo 2. Fuente propia.

Con el data set DTM2 cargado, se realiza el mismo procedimiento que se realizó en el modelo 1, seleccionado el data set DTM2, la variable de salida o dependiente y las variables de entrada o predictoras. La siguiente figura muestra como quedaron configurados estos parámetros.

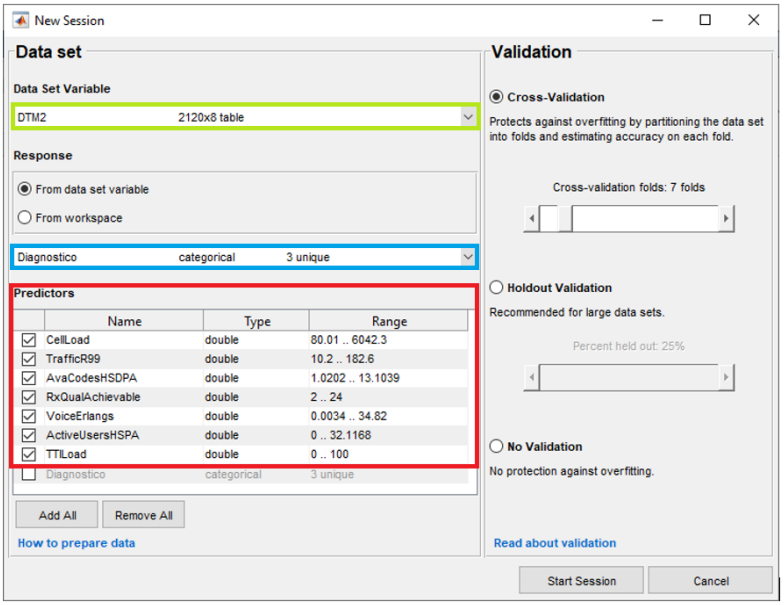


Figura No. 4-18. Configuración APP Classification Learner – Modelo 2. Fuente propia.

Para el desarrollo del modelo 2 también se escoge la opción "Fine Tree", y así tener un mayor detalle en los procesos de bifurcación y cantidad de hojas del árbol. Es importante tener presente que para este modelo se tiene tres posibles clases o categorías en las que se pueden clasificar los registros de ejemplo.

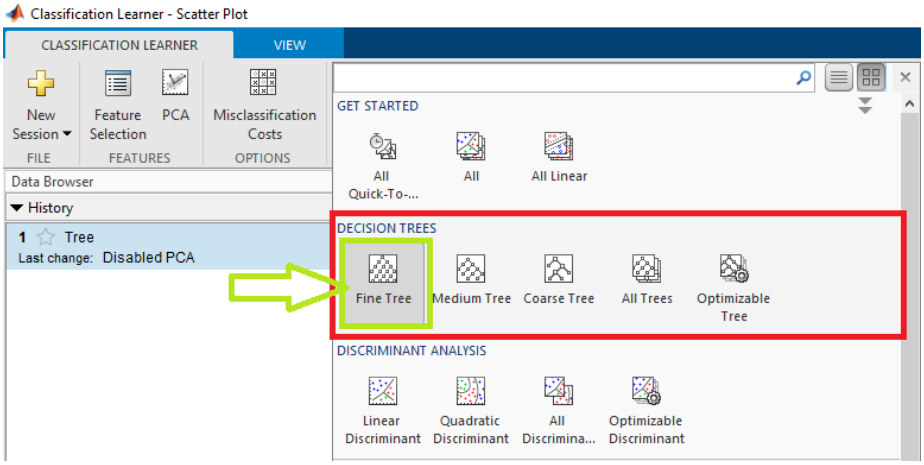


Figura No. 4-19. Entrenamiento "Fine Tree" – Modelo 2. Fuente propia.

Después del proceso de entrenamiento para el modelo 2 se obtienen los siguientes resultados. En la siguiente figura se puede observar los diferentes datos de cada una de las variables que se transponen para las tres clases COV, CAP y CHECK. La desviación estándar es mínima para la gran mayoría de las coordenadas (Nombre de Variable, Valor Desviación Estándar). De todas las variables, talvez la que presenta una mayor desviación en sus muestras, es la quinta variable de entrada denominada VoiceErlangs. El gran grueso de las muestras se encuentra dentro de una desviación

estándar inferior a 2.5, las muestras por fuera de este rango son muy pocas y podrían tener algún efecto durante el proceso de entrenamiento generando varios nodos de decisión.

En relación al tiempo de entrenamiento podemos ver que, aunque se tienen solo 2120 registros que es mucho menor que la cantidad de registros del modelo 1 (28091 registros), se incrementa el tiempo de este proceso a un total de 28,633 segundos, esto se explica por el hecho de que el algoritmo ya no tiene que generar el modelo para clasificar 2 categorías, sino que en el modelo 2 ya se incrementan a 3 categorías. De este análisis se puede concluir que el tiempo de entrenamiento dependerá en mayor medida del número de categorías que se deben clasificar, a mayor categorías mayor tiempo de entrenamiento.

Pasando a la precisión se puede ver que esta disminuye a un 99.6%, y aunque es menor de la que se logró en el modelo 1, se sigue considerando un muy buen porcentaje de precisión. El costo total de la clasificación errónea es de 9, esto indica que 9 registros del total de ejemplos no se pudieron clasificar correctamente.

Finalmente se puede ver que el algoritmo realiza en promedio un aproximado de 62.000 observaciones por segundo. Al compararlo con el algoritmo de modelo 1, se disminuyeron el número de observaciones por segundo en aproximadamente un 84%.

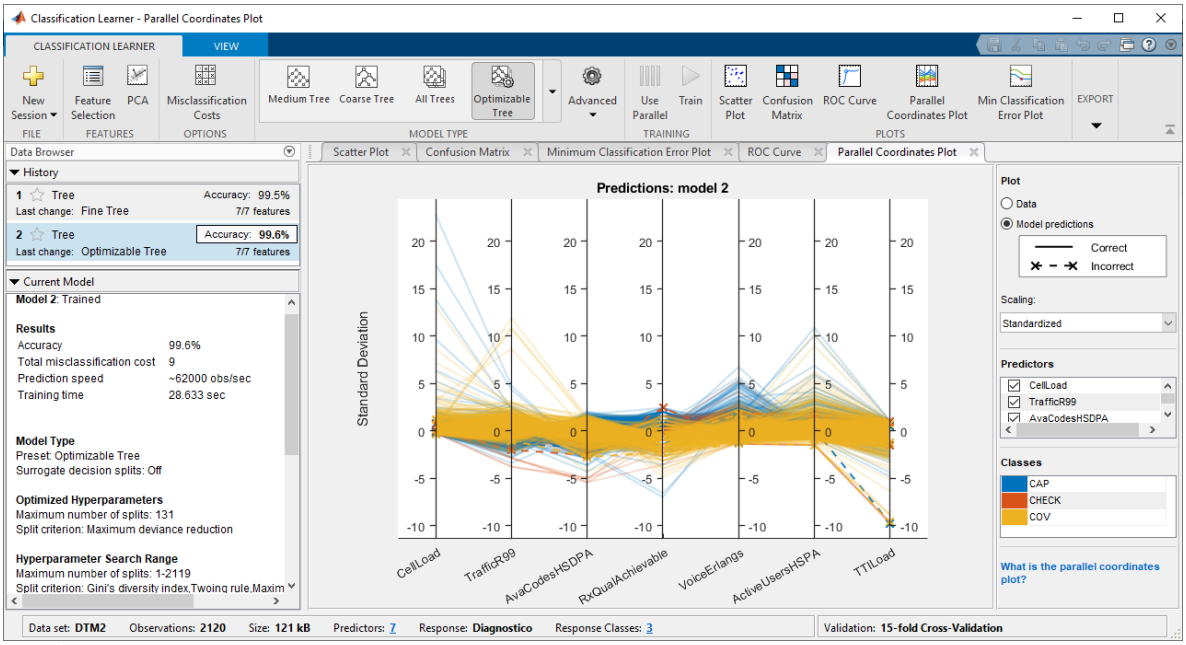


Figura No. 4-20. Desviación estándar – Variables de entrada Modelo 2. Fuente propia.

Para este modelo que tiene tres categorías se tendrán tres resultados de las funciones curvas ROC y AUC. En las categorías COV y CAP se logra una AUC de 1 indicando una muy alta probabilidad de que se clasifiquen correctamente los registros de estas categorías. De igual manera las curvas ROC de estas dos categorías tienen un 100% de tasa para verdaderos positivos y un 0% en la tasa de falsos positivos. En resumen, se puede decir que el modelo 2 presenta un buen desempeño durante

el proceso de clasificación para estas dos categorías. Las siguientes dos figuras muestra los resultados obtenidos de estas dos funciones para las clases CAP y COV.

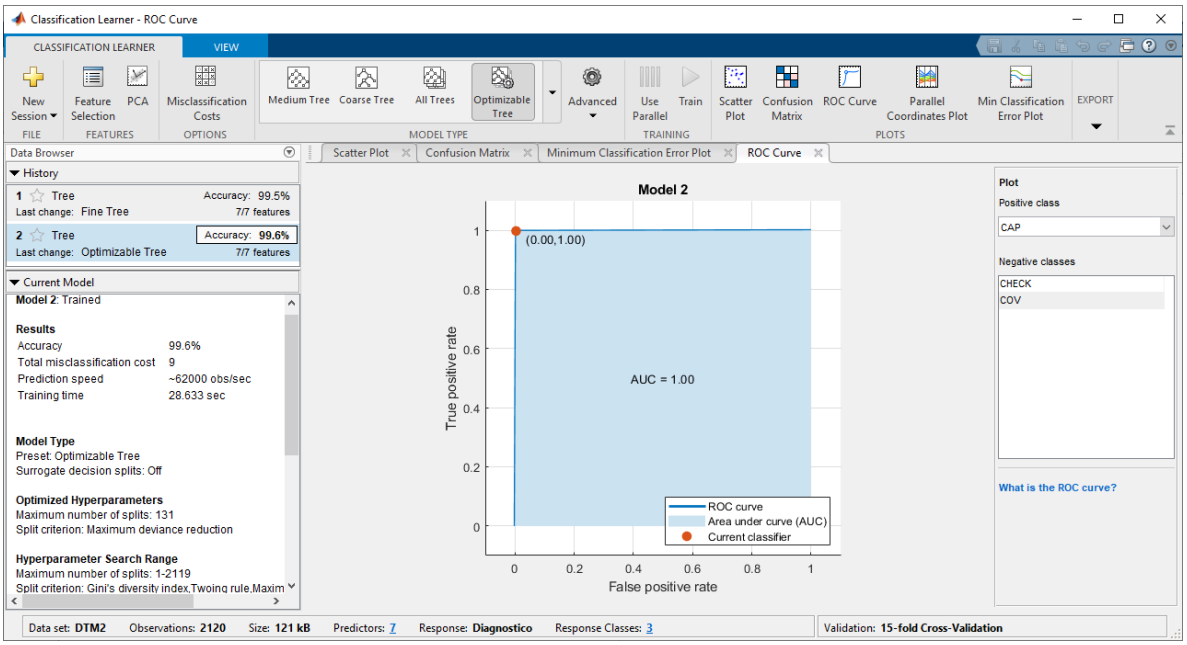


Figura No. 4-21. AUC y curvas ROC Clase CAP. Fuente propia.

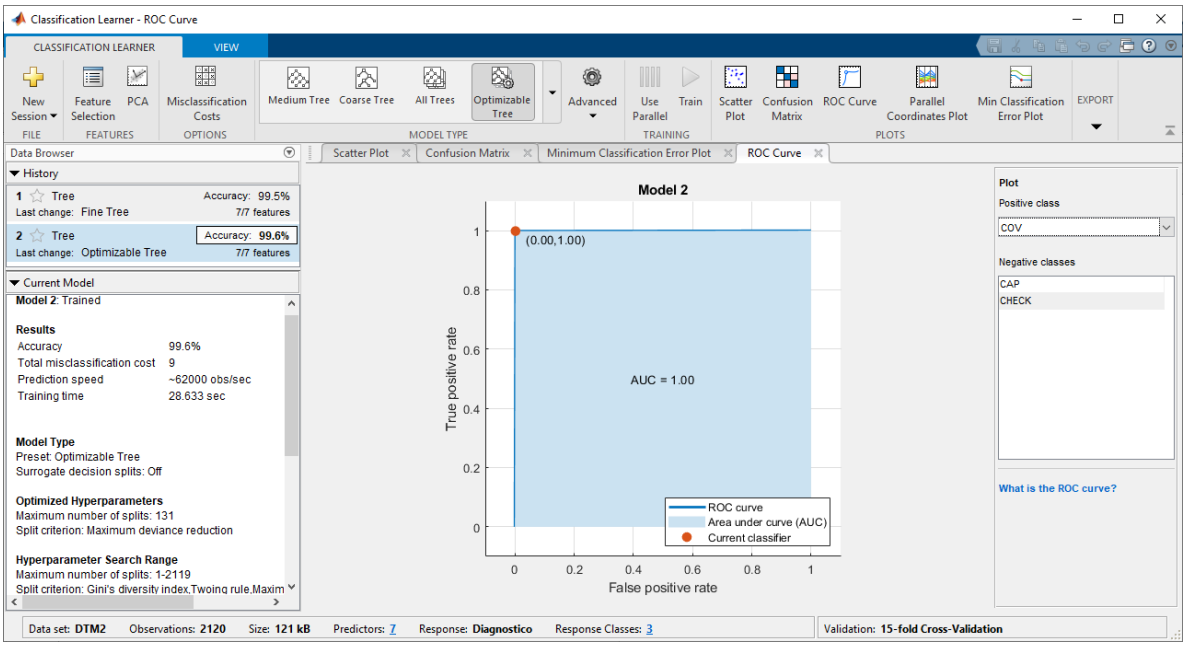


Figura No. 4-22. AUC y curvas ROC Clase COV. Fuente propia.

Por otro lado, se puede ver que los resultados de las funciones AUC y curvas ROC obtenidos para la clase CHECK no son tan excelentes como en las clases anteriores que fueron muy próximos al 100%. El AUC llego a un 0.92 lo que indica que hay un 92% de probabilidad de que el modelo clasifique una instancia positiva elegida aleatoriamente más alta que una instancia negativa. Este valor de ACU

inferior al resto de clases tendría relación con el bajo número de registros que se utilizaron para entrenar el modelo, pues como se observó en la distribución de las muestras, solo 30 registros pertenecen a la clase CHECK de un total de 2120. En otras palabras, solo el 1.4% del total de los datos se utilizaron para entrenar el modelo en la clase CHECK. Sin embargo, un valor de AUC superior a 0.9 puede considerarse como bueno.

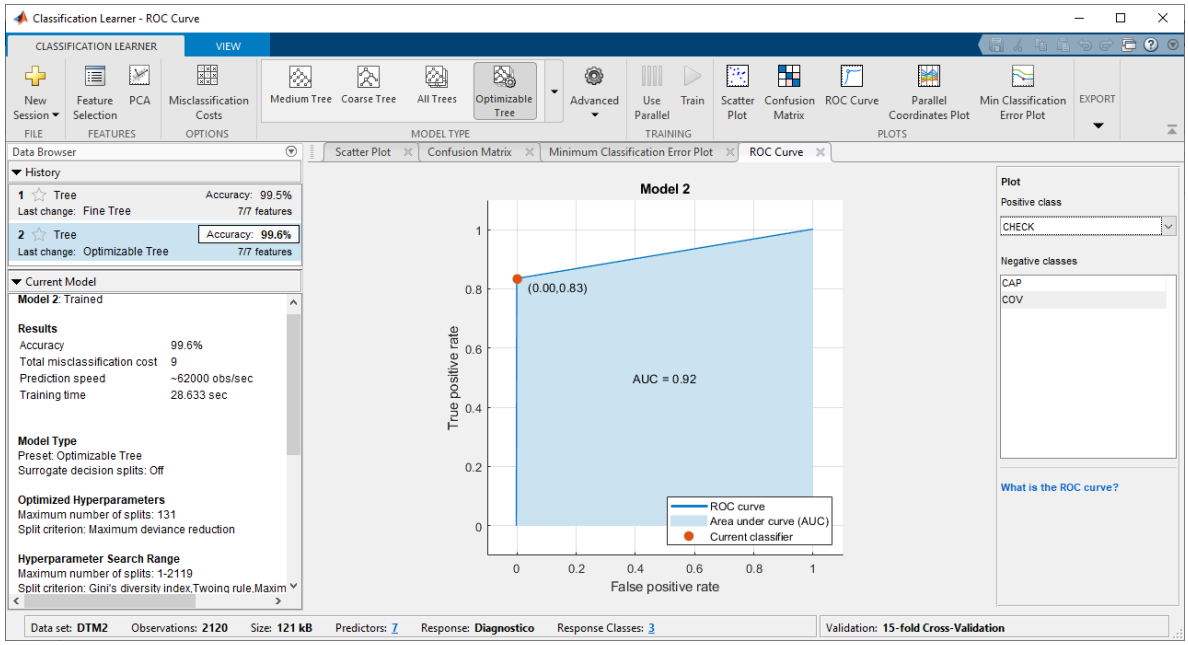


Figura No. 4-23. AUC y curvas ROC Clase CHECK. Fuente propia.

4.2.2.1 Métricas matrices de confusión – Técnica Árbol de Decisión Modelo 2

Finalmente se presenta la matriz de confusión la cual ya no corresponde a una clasificación binaria, sino que corresponde a una clasificación multiclase dado que el modelo debe predecir tres tipos de categorías o clases. En este caso el cálculo de las métricas sigue considerando los valores de TP, TN, FP y FN, pero sus posiciones cambian dependiendo para cuál de las clases se está realizando el cálculo. Para las métricas exactitud y tasa de error, el cálculo se realiza de la misma forma como se realiza para una clasificación binaria, esto dado a que se pueden considerar como métricas generales del modelo. Por otro lado, para las métricas sensibilidad, especificidad y precisión, el cálculo se deberá realizar de acurdo a la clase que se quiere analizar y la distribución de sus componentes quedaría tal como se muestra en la siguiente figura.

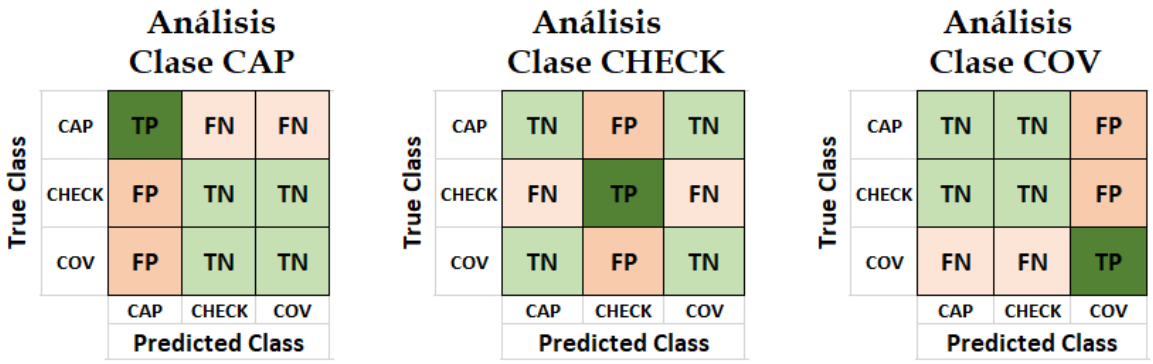


Figura No. 4-24. Matrices Multiclase CAP, CHECK y COV. Árbol de decisión Modelo 2. Fuente propia.

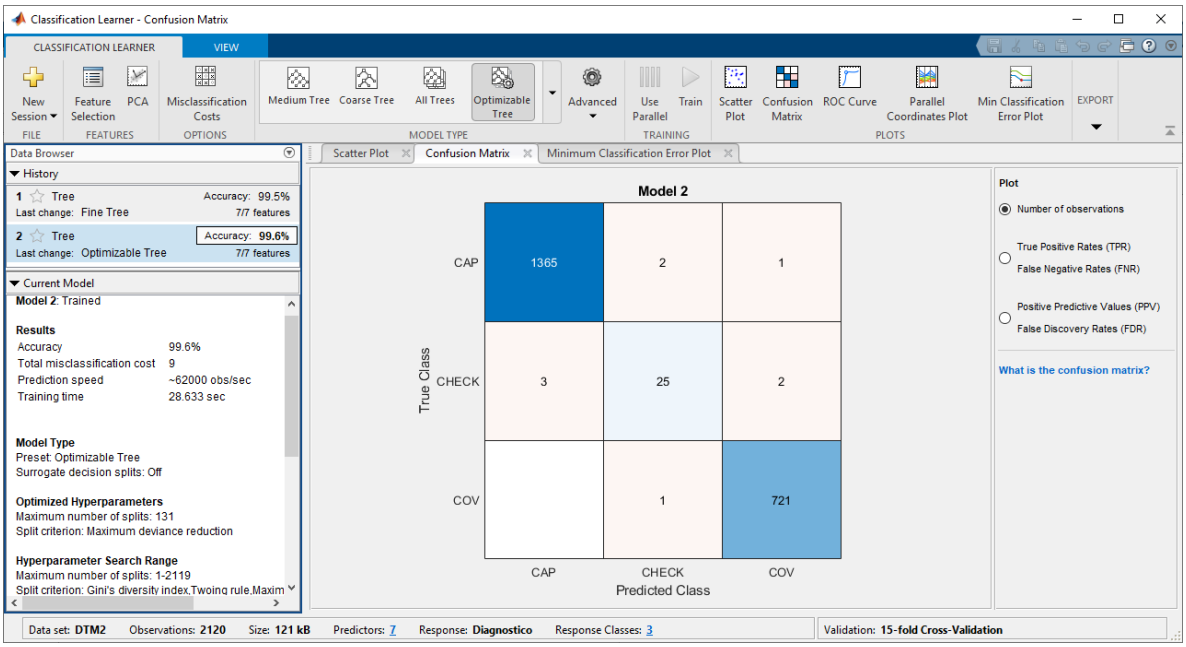


Figura No. 4-25. Matriz de Confusión (Número de observaciones)– Modelo 2. Fuente propia.

A continuación, se realizan los cálculos de todas las métricas que nos permiten evaluar el modelo.

$$\boldsymbol{Exactitud} = \frac{\sum TP_{Todas\ las\ clases}}{Total\ de\ Datos} = \frac{1365 + 25 + 721}{2120} = \frac{2111}{2120} = 0.995754$$

El modelo ofrece una muy buena exactitud, ya que su valor es muy cercano al 100%, con esto se tiene que los resultados de la clasificación de los registros están muy cerca del valor verdadero para cada una de sus clases CAP, COV y CHECK. En otras palabras, se puede decir que prácticamente no existe gran diferencia entre el valor real y el valor esperado.

$$\boldsymbol{Tasa\ de\ error} = \frac{\sum(FP + FN)_{\boldsymbol{Todas\ las\ clases}}}{Total\ de\ Datos} = \frac{3 + 2 + 2 + 1 + 1}{2120} = \frac{9}{2120} = 0.00424583$$

La tasa de error obtenida en el entrenamiento de este modelo también es baja con un valor muy cercano a 0%, por lo tanto, se puede decir que el modelo tiene un muy bajo porcentaje de registros clasificados incorrectamente en cada una de las clases CAP, COV y CHECK.

$$\boldsymbol{Sensibilidad\ CAP} = \frac{TP}{TP + FN} = \frac{1365}{1365 + 2 + 1} = \frac{1365}{1368} = 0.9978$$

$$\boldsymbol{Sensibilidad\ COV} = \frac{VP}{TP + FN} = \frac{721}{721 + 1} = \frac{721}{722} = 0.9986$$

$$\boldsymbol{Sensibilidad\ CHECK} = \frac{VP}{TP + FN} = \frac{25}{25 + 3 + 2} = \frac{25}{30} = 0.8333$$

Para este modelo 2 dado a que se definen tres posibles categorías de clasificación, es necesario realizar el cálculo de la sensibilidad se forma separada para cada una de las clases. Se puede decir que los valores obtenidos son bastante altos para las clases CAP y COV con porcentajes superiores al 99.78%. Por lo anterior se puede concluir que el modelo tiene una alta capacidad para discriminar los casos de celdas UMTS con problemas de capacidad y cobertura que son etiquetados como CAP y COV respectivamente. Para el caso de la clase CHECK la sensibilidad solo alcanza el 83.33%, y aunque es un porcentaje menor comparado con las otras clases, se considera bueno en su propósito de clasificar mediante predicción los registros de esta clase.

En la siguiente figura entregada por la aplicación de Matlab se pueden comprobar los porcentajes calculados para la sensibilidad de cada una de las clases.

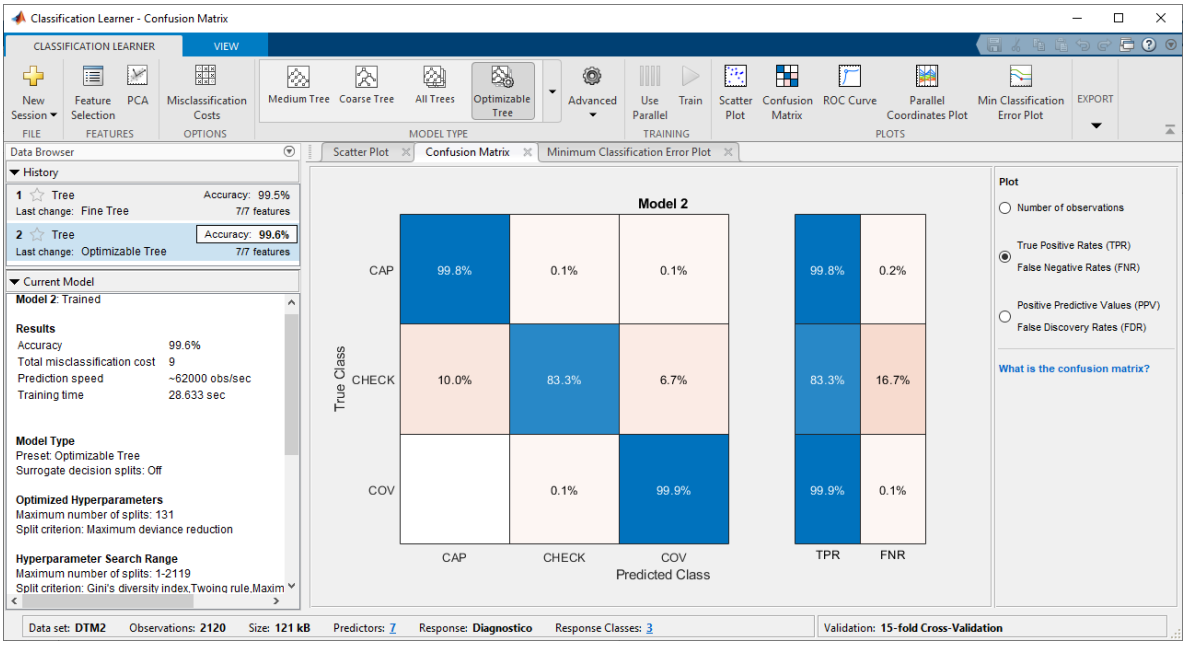


Figura No. 4-26. Matriz de Confusión (TPR/FNR)– Sensibilidad Modelo 2. Fuente propia.

$$\boldsymbol{Especificidad\ CAP} = \frac{TN}{TN + FP} = \frac{25 + 721 + 2 + 1}{25 + 721 + 2 + 1 + 3} = \frac{749}{752} = 0.99601$$

$$\boldsymbol{Especificidad\ COV} = \frac{TN}{TN + FP} = \frac{1365 + 25 + 3 + 2}{1365 + 25 + 3 + 2 + 2 + 1} = \frac{1395}{1398} = 0.99785$$

$$\boldsymbol{Especificidad\ CHECK} = \frac{TN}{TN + FP} = \frac{1365 + 721 + 1}{1365 + 721 + 1 + 2 + 1} = \frac{2087}{2090} = 0.99856$$

Para la métrica de la especificidad también es necesario realizar el cálculo de forma separada para cada una de las clases que se pueden predecir con el modelo. El porcentaje que se logra en las tres clases es muy alto cercano al 100%, con esto se puede decir que el modelo tiene en general una muy alta capacidad para identificar las celdas UMTS que no pertenecen a una determinada clase.

$$\boldsymbol{Precisión\ CAP} = \frac{TP}{TP + FP} = \frac{1365}{1365 + 3} = \frac{1365}{1368} = 0.9978$$

$$\boldsymbol{Precisión\ COV} = \frac{TP}{TP + FP} = \frac{721}{721 + 2 + 1} = \frac{721}{724} = 0.9958$$

$$\boldsymbol{Precisión\ CHECK} = \frac{TP}{TP + FP} = \frac{25}{25 + 2 + 1} = \frac{25}{28} = 0.8928$$

La precisión que se logra para las clases CAP y COP es bastante alta, es decir que se presentaron muy pocas fallas en las que al final se abrían clasificado como CAP o COP respectivamente dentro de los falsos positivos de cada una de estas clases. Por su parte, la precisión para la clase CHECK es un poco menor, y aunque tiene el mismo número de FP que las otras clases, en esta se presenta un alto porcentaje debido a que se tienen muy pocos registros de ejemplos que pertenecen a esta clase CHECK.

En la siguiente figura entregada por la aplicación de Matlab se pueden comprobar los porcentajes calculados para la precisión de cada una de las clases.

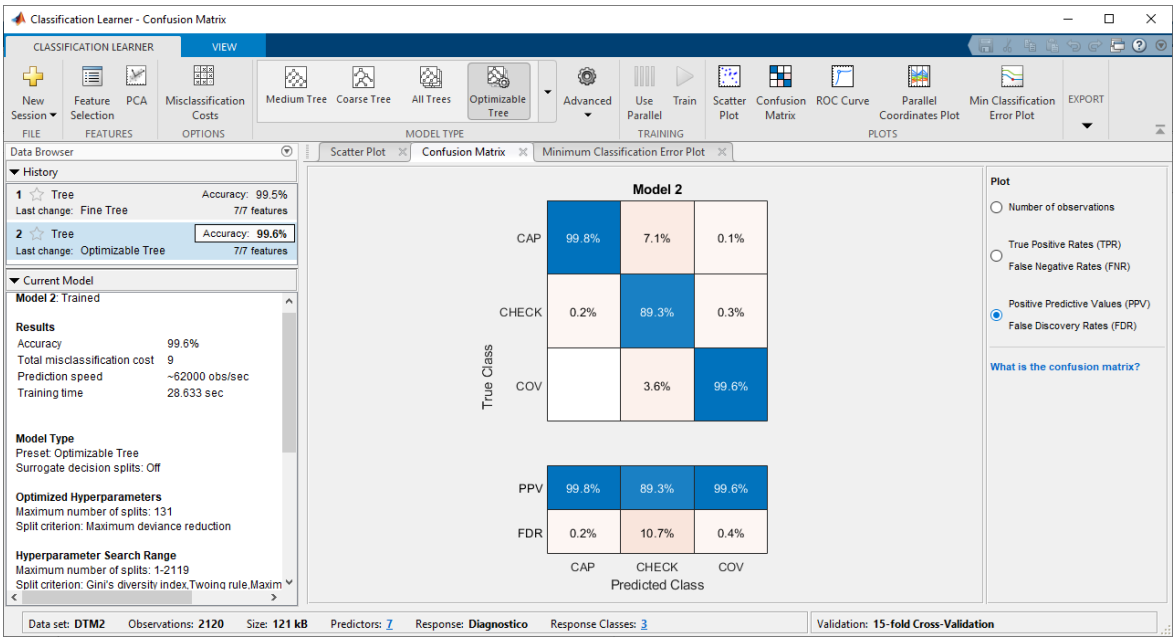


Figura No. 4-27. Matriz de Confusión (PPV/FDR) – Precisión Modelo 2. Fuente propia.

4.2.2.2 Resultado árbol de decisión modelo 2

Al final, cuando el proceso de entrenamiento ha terminado, es posible graficar el árbol de decisión que el algoritmo tendrá en cuenta para clasificar los nuevos registros en las tres posibles categorías del modelo 2 que son CAP, CHECK y COV. Como se puede ver en la siguiente figura, el algoritmo determina que para realizar el proceso de clasificación solo considera cinco de las siete variables de entrada. Se inicia el árbol colocando en el nodo raíz la variable RxQualAchievable, continua en los nodos intermedios con las variables VoiceErlangs, TrafficR99, AvaCodesHSDPA y CellLoad. De esto se puede concluir que en el análisis que realiza el algoritmo con todas las posibles opciones de árboles de decisión, la más efectiva para la tarea de clasificación es esta opción. A continuación, se listan cada una de las variables y sus umbrales de decisión utilizados para crear las divisiones.

- RxQualAchievable = 17.5
- VoiceErlangs = 10.1659, =10.1284, =0.06594, =10.4617, =0.83609
- TrafficR99 = 59.95, =59.8
- CellLoad = 376.085
- AvaCodesHSDPA = 8.0001, =8.1244

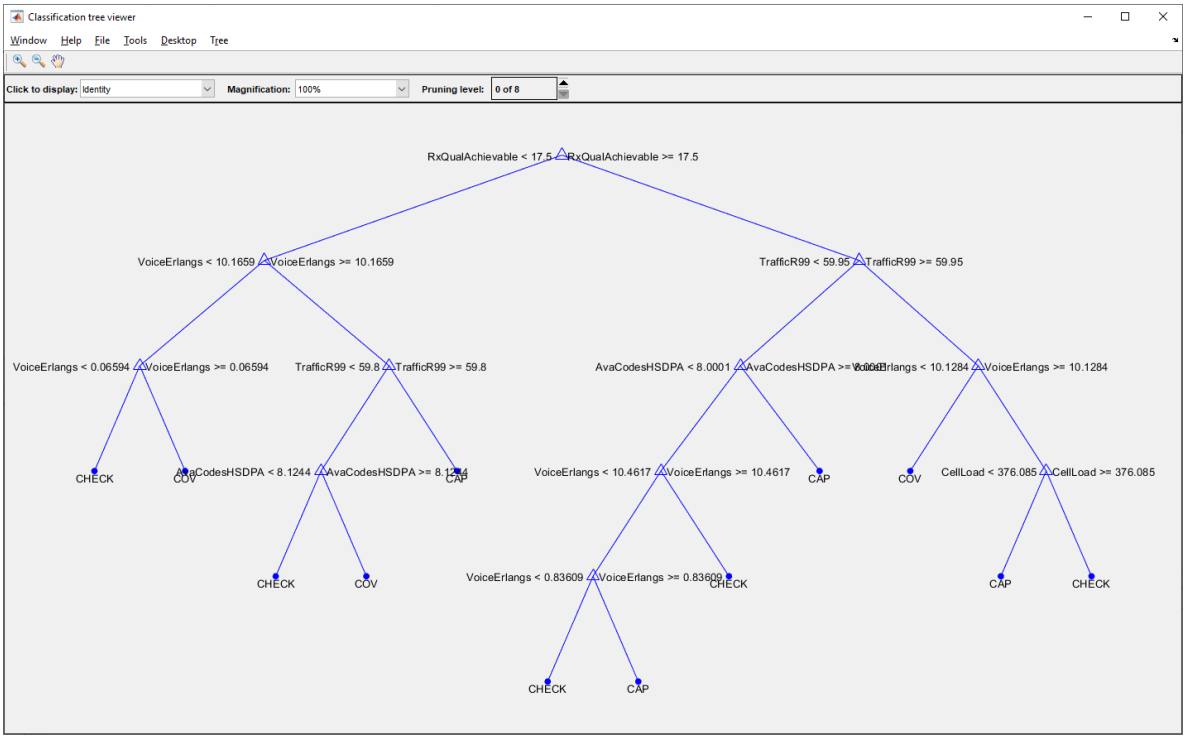


Figura No. 4-28. Árbol de decisión Modelo 2. Fuente propia.

4.3 TÉCNICA RED NEURONAL

4.3.1 Desarrollo Modelo 1 – Red Neuronal

Las redes neuronales son consideradas una alternativa importante para dar solución a problemas de clasificación, pero estas técnicas a diferencia de las de árboles de decisión, demanda que la salida sea binaria, es decir que en los datos de salida no se podrán tener etiquetas como OK, NOK, CAP, CHECK o COV. Para solucionar este requerimiento es necesario codificar las categorías en un formato binario. A continuación, se presentan las convenciones de codificación que se utilizaran para el desarrollo del modelo 1.

Tabla No. 4-4. Codificación salidas Modelo 1 - Red Neuronal. Fuente propia.

Modelo 1

Etiqueta	Código binario
OK	1
NOK	0

Para el modelo 1 que se encarga de clasificar las celdas UMTS en dos categorías, se requerirá una sola variable de salida, el valor 1 para las celdas que están sin problemas y el valor de 0 para las celdas que presentan algún problema y que por lo tanto demanda un plan de acción para optimizar su funcionamiento.

Dentro de las APPs de Matlab existen varias opciones de redes neuronales, pero la que se acomoda a nuestro problema de clasificación por aprendizaje supervisado es la red neuronal para reconocimiento de patrones. La siguiente figura ilustra la ubicación de dicha aplicación que se deberá arrancar para desarrollar cada uno de los modelos.

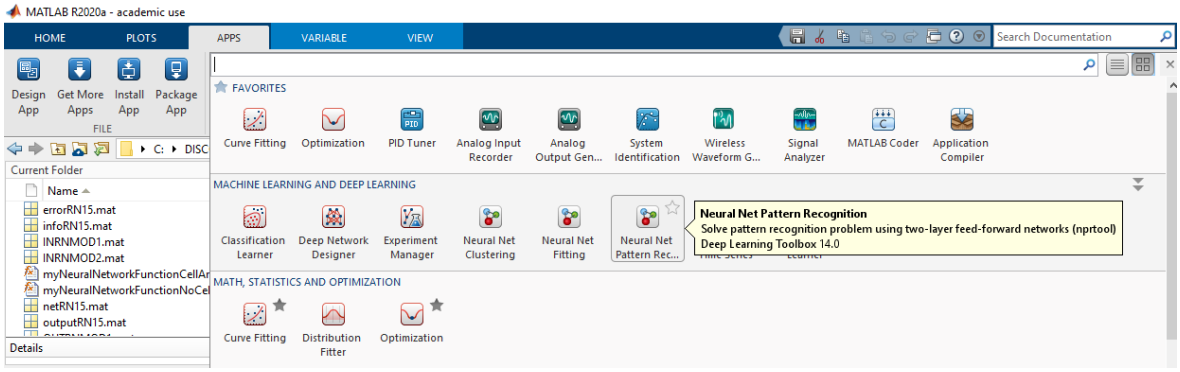


Figura No. 4-29. APP Neuronal Net Pattern Recognition. Fuente propia.

Previo al inicio de la configuración de la APP es necesario importar los data set con su correspondiente ajuste en las variables de salida, es decir la codificación binaria que se mencionó

antes. Las siguientes figuras ilustran las matrices cargadas en Matlab con las cuales se realizará el proceso de entrenamiento para el modelo 1 utilizando redes neuronales.

Para ejecutar el entrenamiento en la APP de redes neuronales seleccionada, es necesario definir por separado los datos de entrada y los datos de salida. Las siente variables de entrada se cargaron en la matriz de orden 28091x7 llamada INRNMOD1 tal como lo muestra la siguiente figura.

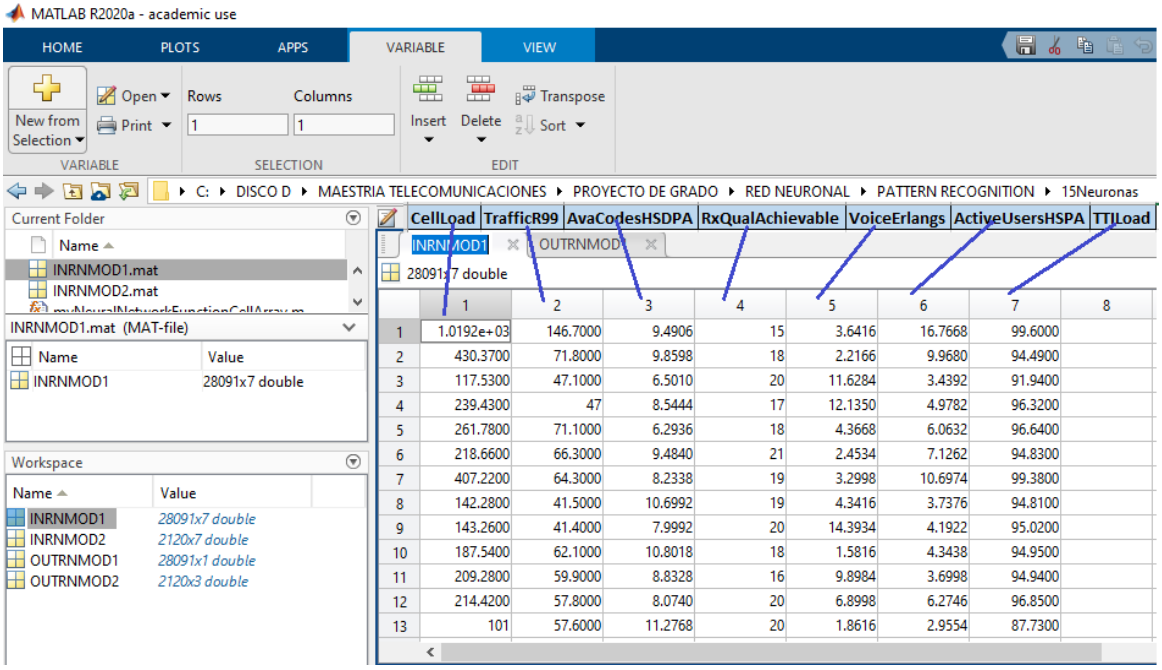


Figura No. 4-30. Variables de entrada - Red Neuronal Modelo 1. Fuente propia.

Por su parte la variable de salida es cargada en la matriz de orden 28091x1 y con el nombre OUTRNMOD1 tal como se muestra en la siguiente figura.

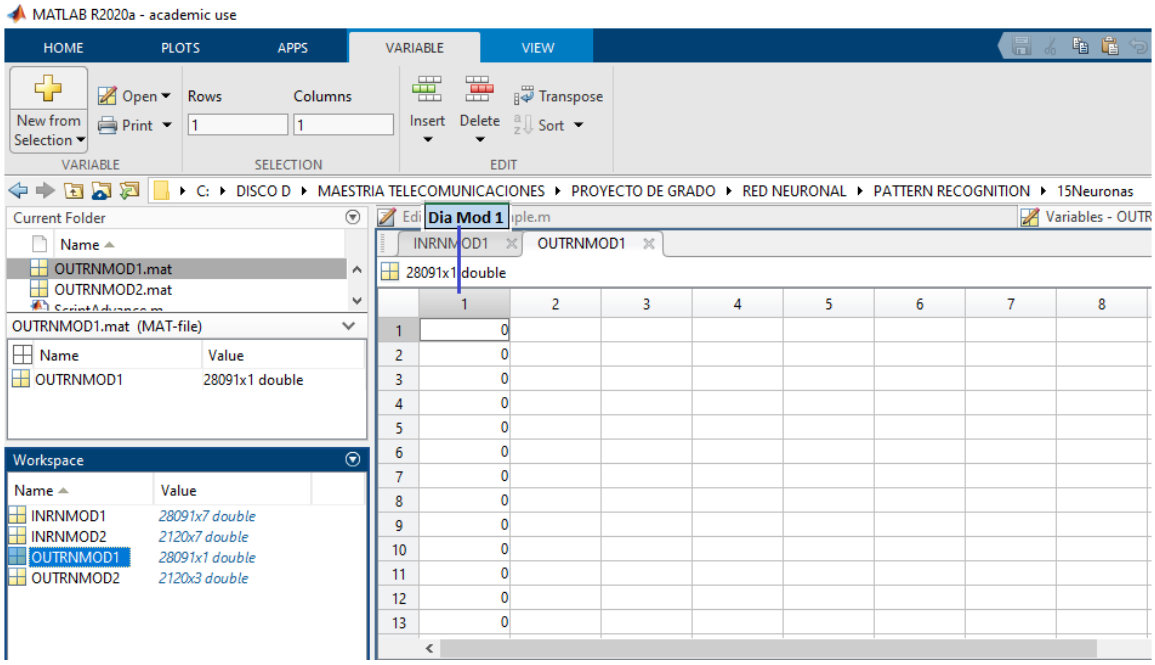


Figura No. 4-31. Variable de salida - Red Neuronal Modelo 1. Fuente propia.

Una vez iniciada la APP se procede a seleccionar los datos de entrada que se cargaron en la matriz INRNMOD1 y los datos de salida cargados en la matriz OUTRNMOD1, adicional es importante indicar como están organizados los ejemplos de entrenamiento, si están en columnas o en filas. Como se puede ver en la siguiente figura, esta configuración de parámetros es sencilla.

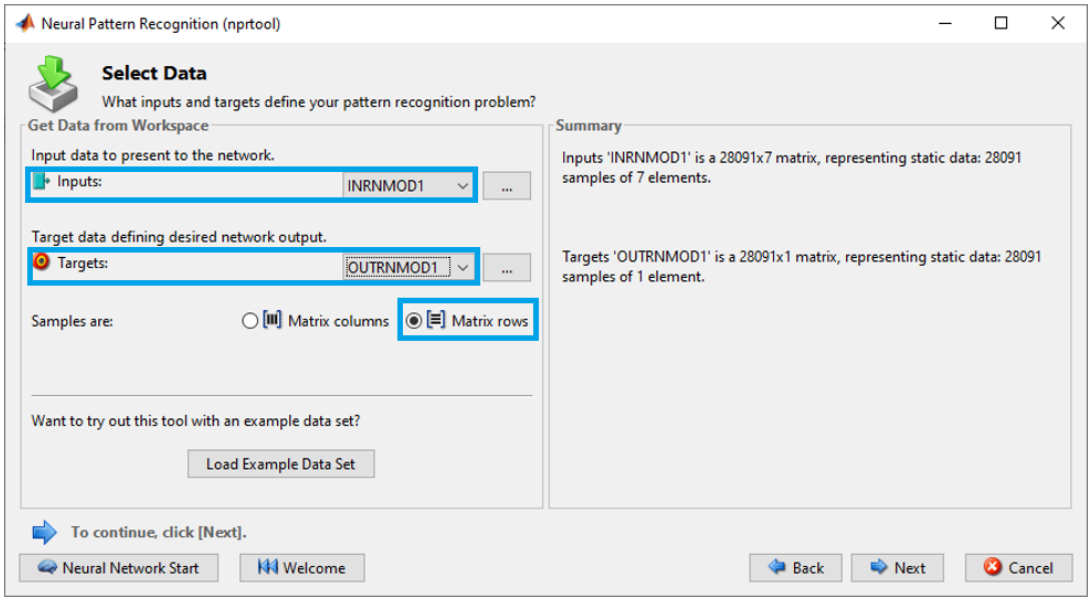


Figura No. 4-32. Configuración APP Neural Pattern Recognition – Modelo 1. Fuente propia.

El siguiente paso consiste en definir los porcentajes de datos que utilizarán para las fases de "Training", "Validation" y "Testing". Dependiendo de la cantidad de muestras de ejemplo de las que se disponga, se pueden ajustar los porcentajes para cada fase. Para el modelo 1 en el que se

clasifican las celdas entre OK y NOK se dispone de un total de 28.091 registros, lo cual es una gran cantidad de muestras. Por lo anterior se dejan los porcentajes por defecto que sugiere la aplicación. La siguiente figura ilustra lo anteriormente indicado.

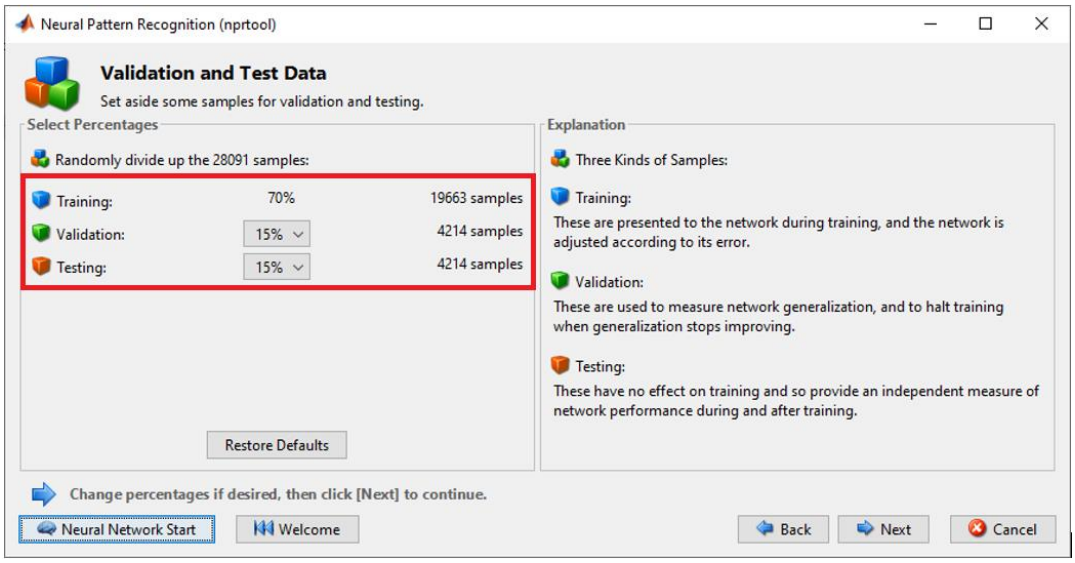


Figura No. 4-33. Asignación datos de entrenamiento, validación y prueba Modelo 1

Para la fase de "Training" se asigna el 70% de los datos de entrenamiento, es decir, 19.663 registros. Las fases de "Validation" y "Testing" tienen asignadas el 15% cada una. Es importante indicar que los registros de estos porcentajes de datos no son asignados de forma consecutiva, sino que la APP los asigna de manera aleatoria. Por otro lado, también es importante indicar que solo las fases de "Training" y "Validation" son las principales responsables en la definición del modelo, en la primera se realiza un primer ajuste del modelo soportado en el error y en la segunda mediante la medida de la generalización de la red neuronal. En ambas fases se hace uso tanto de los datos de entrada como de los datos de salida para dichos ajustes. Por su parte la fase de "Testing" no impacta en el entrenamiento del modelo y solo provee una medida independiente del desempeño de la red neuronal.

Esta aplicación de Matlab basada en redes neuronales solo dispone de una capa oculta tal como se muestra en la siguiente figura, pero en esta capa oculta si es posible definir el número de neuronas. Después de un ejercicio de pruebas del modelo con diferente número de neuronas en la capa oculta, se identifica que una red con 15 neuronas es la opción que mejor desempeño logra para el modelo. La siguiente figura ilustra la arquitectura de red donde se identifican 7 neuronas para la capa de entrada y cuya función es la de recoger la información de todas las variables de entrada, una neurona por variable. La siguiente es la capa oculta y está conformada por 15 neuronas donde se fundamenta principalmente el modelo. Se continua con la capa de salida conformada por una sola neurona en la que se recogen todas las salidas de las 15 neuronas de la capa oculta y entrega una sola salida. Finalmente se tiene la salía donde se entrega la predicción del modelo.

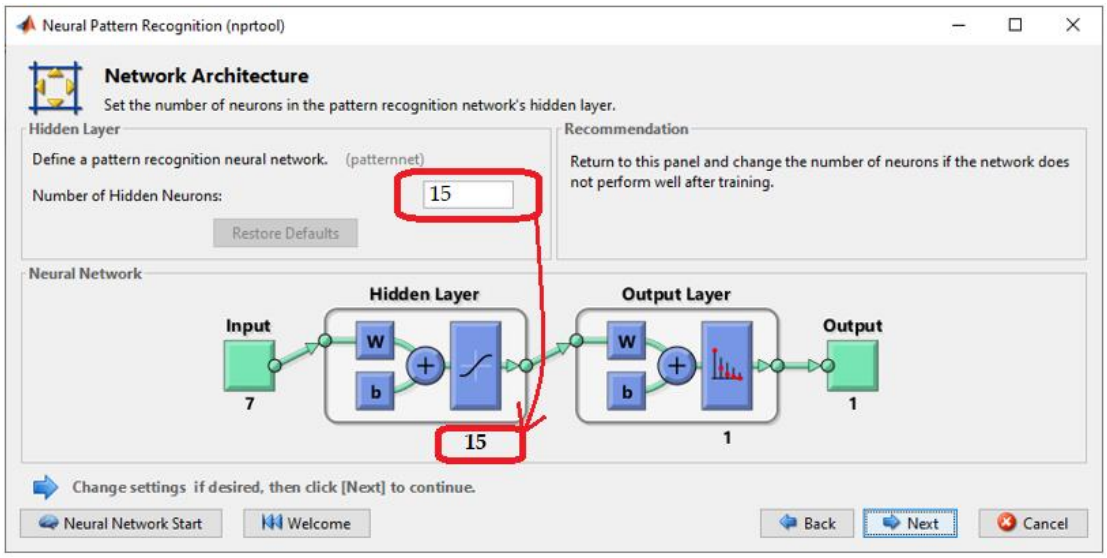


Figura No. 4-34. Arquitectura Red Neuronal – Modelo 1. Fuente propia.

La siguiente figura es otra forma común de representar la anterior arquitectura. Cada capa tiene una dimensión de acuerdo al número de neuronas. A cada neurona se le asocian dos parámetros, un peso W y un bias b tal como se muestra en la anterior figura, estos parámetros son ajustados por el algoritmo a medida que se realiza el entrenamiento del modelo.

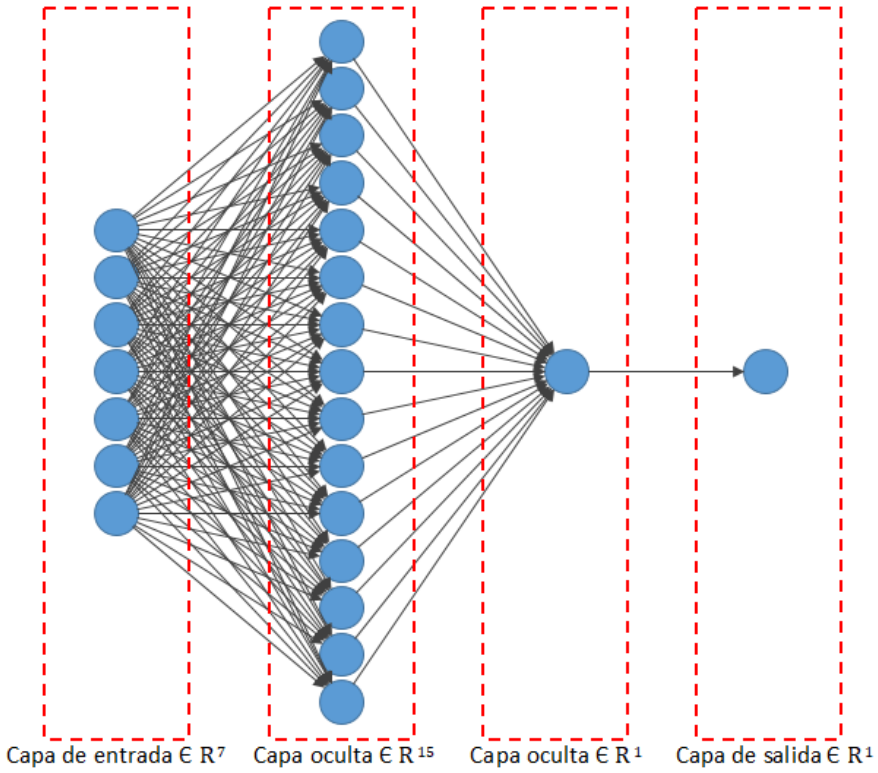


Figura No. 4-35. Arquitectura tradicional Red Neuronal – Modelo 1. Fuente propia.

Después de establecer los parámetros básicos de la red neuronal se procede con el entrenamiento del modelo. Al entrenar la red varias veces, se generará resultados diferentes para diferentes condiciones iniciales y muestreo. Durante el proceso de entrenamiento lo que se busca es minimizar la entropía cruzada lo cual da como resultado una buena clasificación. Los valores más bajos de entropía cruzada son mejores, donde un valor de cero significa que no hay error. Otro indicador que se tiene en cuenta durante el ajuste del modelo, es el porcentaje de error, el cual indica que porción de las muestras de ejemplos son clasificadas erróneamente. Lo ideal es que se den valores de 0% de error. A cada iteración de entrenamiento se le denomina época, y solo se ejecutarán tantas épocas hasta que la generalización determine que ya no se consigue más un mejoramiento del modelo. La mejora del modelo es observada en la fase “Validation” midiendo el error y la entropía cruzada de las muestras asignadas a esta fase. La siguiente figura muestra los resultados obtenidos durante el proceso de entrenamiento para el modelo 1.

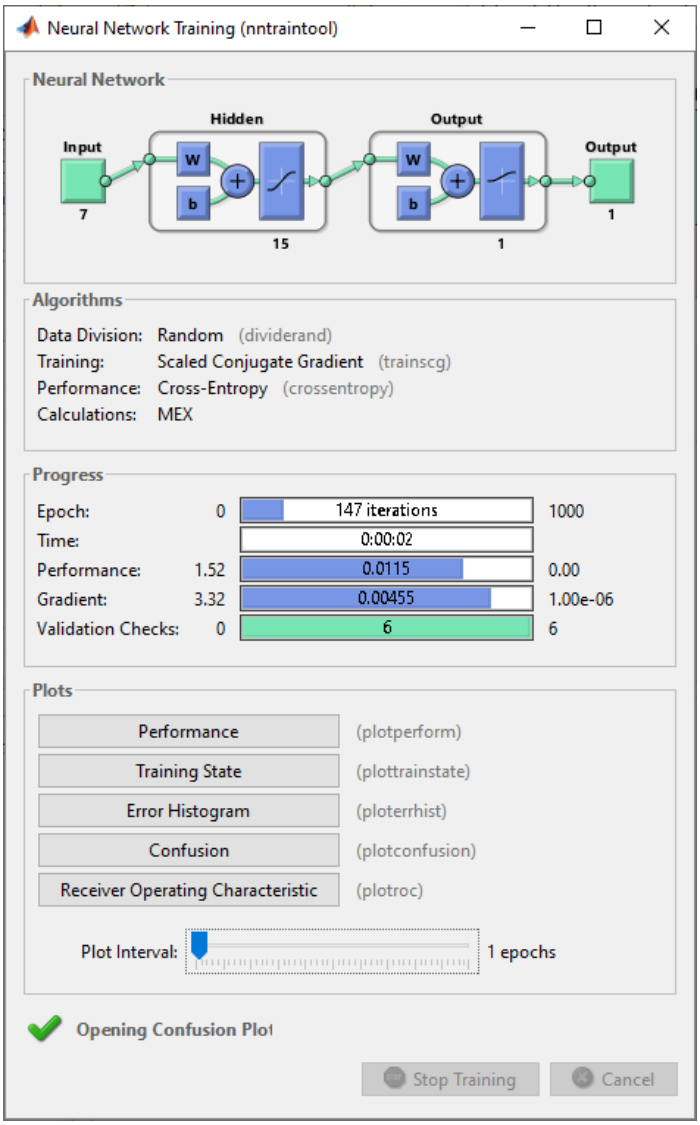


Figura No. 4-36. Resultados proceso de entrenamiento Red Neuronal – Modelo 1. Fuente propia.

Como se puede observar en la anterior figura, para el entrenamiento del modelo solo fueron necesarias 147 épocas o iteraciones, las cuales fueron ejecutadas en aproximadamente 2 segundos.

Para determinar el error en la clasificación de cada una de las muestras de ejemplo, existen varias funciones de error o perdida, pero una de las más utilizadas es la entropía cruzada. No es muy común utilizar el error cuadrático medio puesto que por la dimensión del error que calcula se dificulta la convergencia del gradiente descendente. En otras palabras, con la función de perdida (función de error) dada por la entropía cruzada se hace una mejor dimensión del error para que el gradiente descendente converja mucho más rápido. Es importante recordar que la entropía cruzada se basa en logaritmos, con esto valores pequeños pasan a ser grandes y los valores grandes pasan a estar en una escala relativa. La siguiente figura ilustra los resultados obtenidos para la entropía cruzada en cada una de las fases “Training”, “Validation” y “Testing”. El mejor desempeño que se obtuvo de

la entropía cruzada para la fase "Validation" ocurrió en la época 141 con un valor de 0.011995. Es decir, que el modelo logró su mayor aprendizaje hasta esta época.

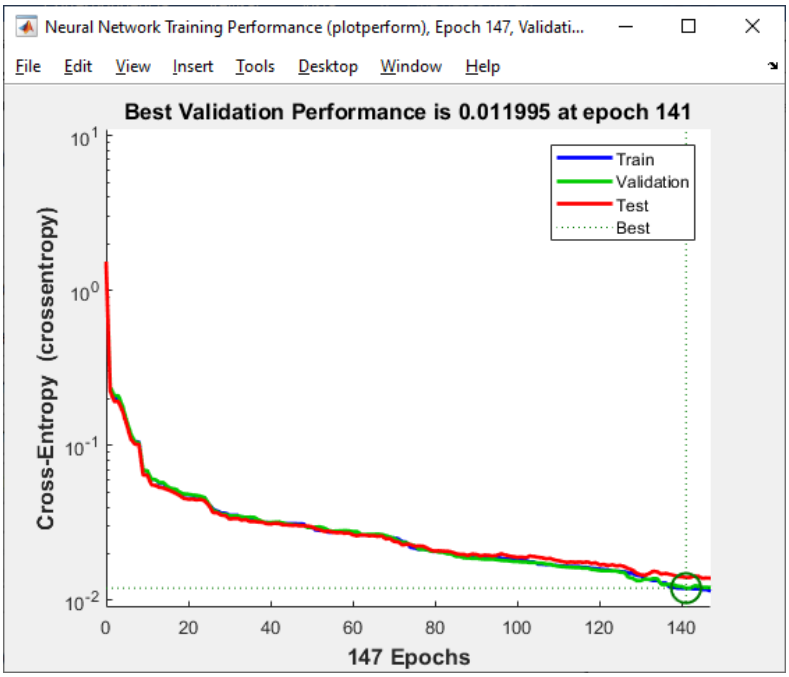


Figura No. 4-37. Entropía cruzada fases de entrenamiento Modelo 1. Fuente propia.

Para el entrenamiento del modelo la APP usa de forma combinada la entropía cruzada y el gradiente descendente. Con la entropía cruzada se crea la función de perdida (función de error) y con el gradiente descendente se busca el mínimo de la función. La siguiente figura ilustra la evolución del gradiente a medida que pasa en cada una de las épocas. El entrenamiento se define con seis controles de validación, es decir que, si el gradiente se incrementa en seis ocasiones consecutivas, el entrenamiento del modelo se detiene, pero se toma el valor más bajo alcanzado, en este caso se detuvo en la época 147, pero se toma el valor de la época 141 como el de mejor desempeño ya que dio el valor más bajo del gradiente.

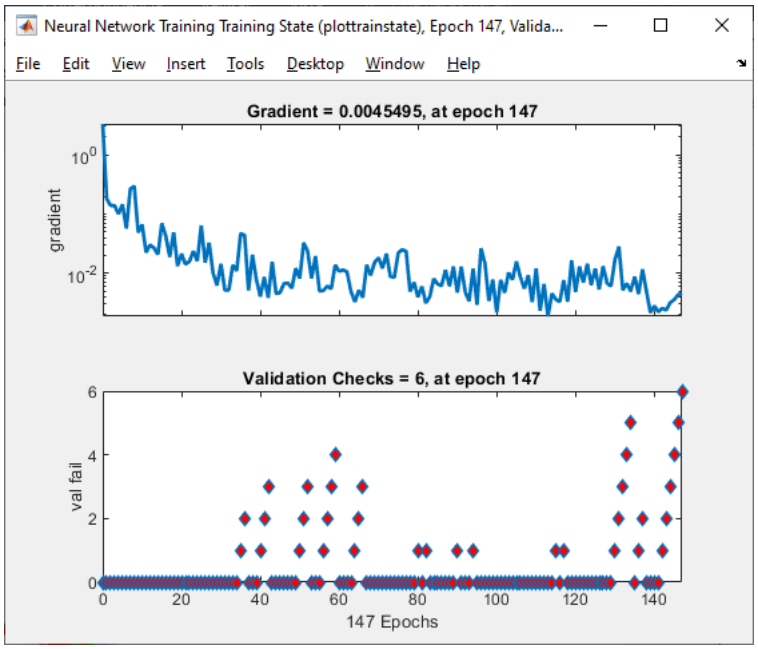


Figura No. 4-38. Evolución gradiente y controles de validación – Red Neuronal Modelo 1. Fuente propia.

En relación a los diferentes valores de error, en la siguiente grafica se presenta el histograma donde se puede apreciar que la gran mayoría de los valores de error se encuentran muy cercanos a cero, esto quiere decir que la diferencia entre el valor predicho por el modelo y el valor real son muy próximos, es un buen indicador de que el modelo funciona bien para realizar las predicciones del estado de las celdas UMTS.

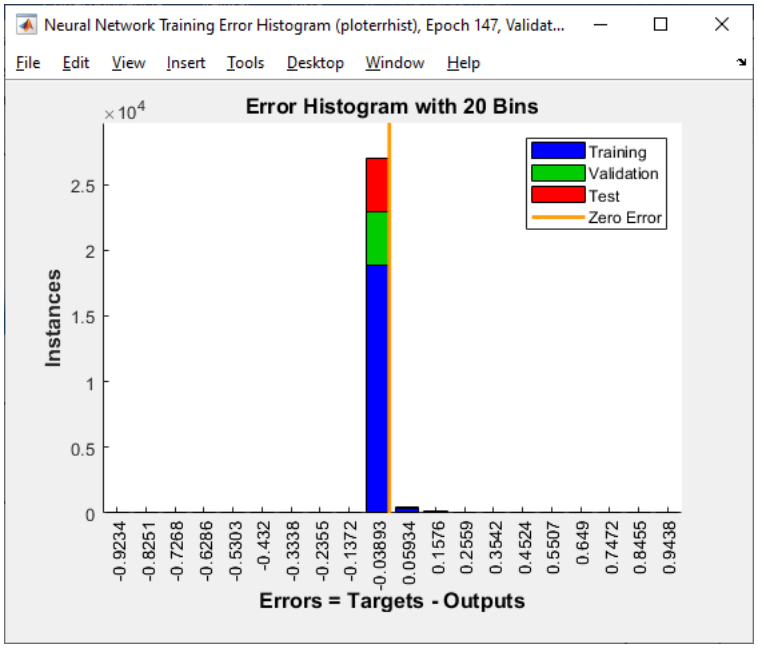


Figura No. 4-39. Histograma – Error en la predicción –Red Neuronal Modelo1. Fuente propia.

Otra forma muy amigable para determinar el desempeño del modelo son las curvas ROC y áreas bajo la curva AUC. La siguiente figura presenta las diferentes curvas ROC y áreas AUC que se obtuvieron en cada una de las fases de entrenamiento. De manera general se puede decir que todas las fases presentaron un buen comportamiento logrando valores muy cercanos al 100% de tasa para verdaderos positivos y 0% para la tasa de falsos positivos. Con estos umbrales de la curva ROC se puede inferir que el AUC de todas las fases está muy cercana a 1, es decir que, la probabilidad de que se dé una adecuada clasificación de las celdas UMTS en las categorías OK y NOK es muy alta, cercano al 100%.

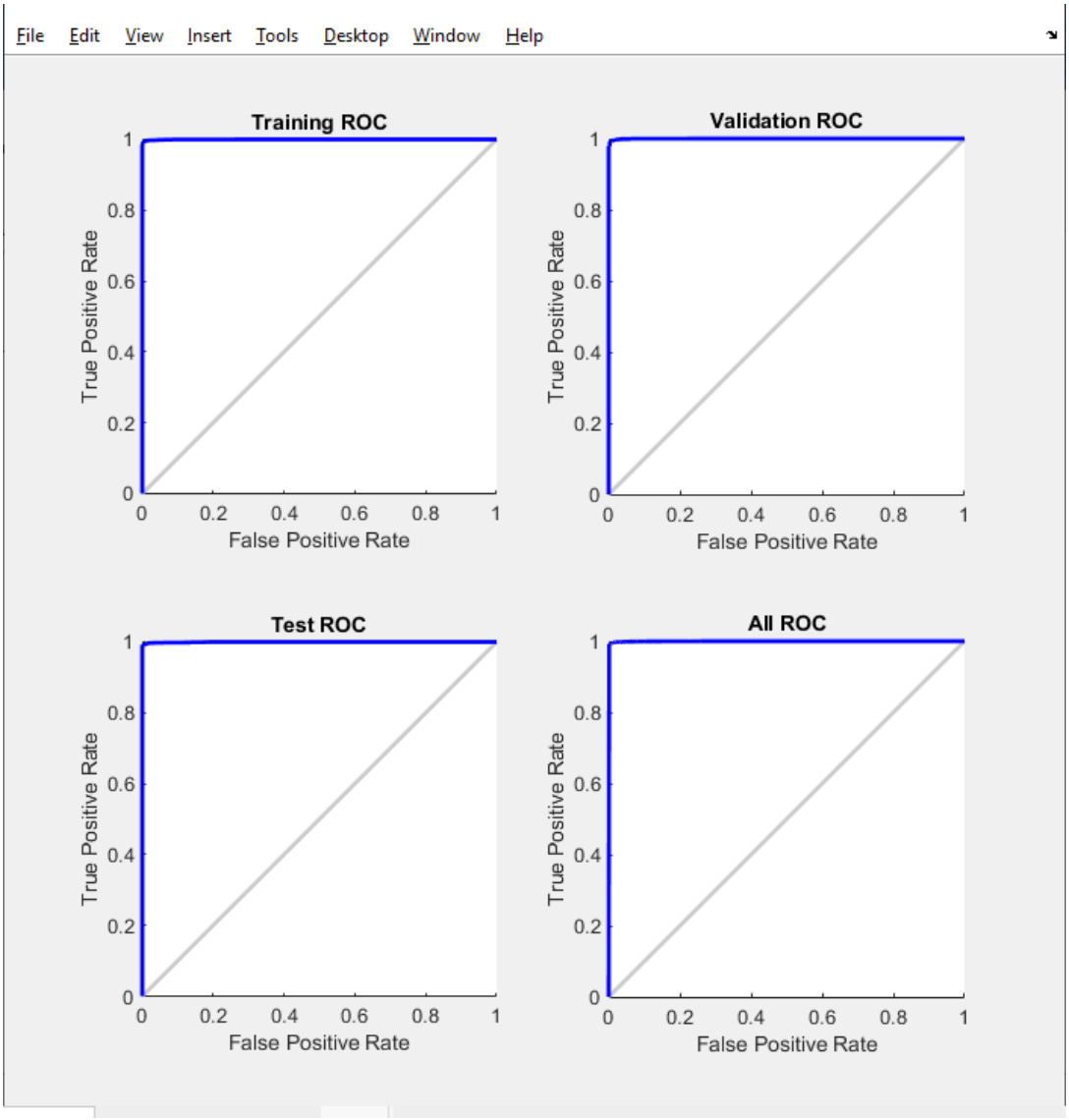


Figura No. 4-40. Curvas ROC y AUC Red Neuronal Modelo 1. Fuente propia.

4.3.1.1 Métricas matrices de confusión – Técnica Red Neuronal Modelo 1

Finalmente se presenta en la siguiente figura, la matriz de confusión para cada una de las fases y el acumulado de todas ellas. Se utilizan los valores de la matriz que acumula todas las fases para el cálculo de las diferentes métricas que permiten evaluar el desempeño del modelo.

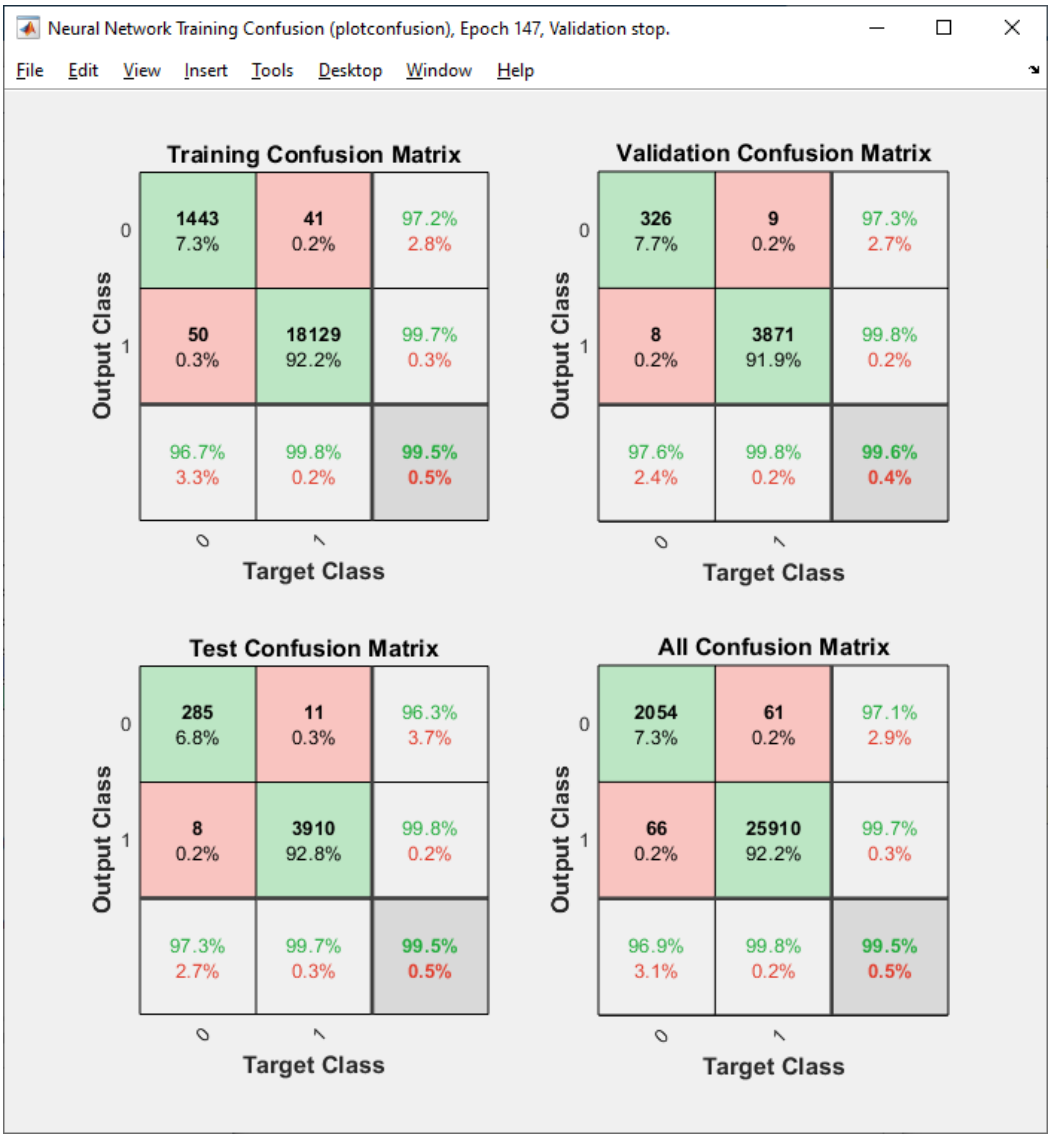


Figura No. 4-41. Matrices de confusión fases Red Neuronal Modelo 1. Fuente propia.

Basados en los valores de la matriz de confusión de la anterior figura, se procede a continuación a realizar los cálculos de cada una de las métricas de desempeño del modelo.

$$\boldsymbol{Exactitud} = \frac{2054 + 25910}{28091} = \frac{28090}{28091} = 0.995479$$

El modelo ofrece una buena exactitud ya que su valor está por encima del 99%, con esto se tiene que los resultados de la clasificación de los registros están muy cerca del valor verdadero para cada

una de sus clases OK y NOK. En otras palabras, se puede decir que es muy pequeña la diferencia entre el valor real y el valor esperado.

$$\boldsymbol{Tasa\ de\ error} = \frac{61 + 66}{28091} = \frac{1}{28091} = 0.004521$$

La tasa de error es demasiado baja con un valor por debajo del 0.5%, por lo tanto, se puede decir que el modelo tiene un bajo porcentaje de registros clasificados incorrectamente, considerando todas las clases OK y NOK.

$$\boldsymbol{Sensibilidad} = \frac{2054}{2054 + 66} = \frac{2054}{2120} = 0.968868$$

En relación a la sensibilidad, se puede decir que el algoritmo es bueno ya que se logra un porcentaje superior al 96%. Por lo tanto, el modelo tiene una buena capacidad para discriminar los casos de celdas UMTS con problemas que se deben clasificar como NOK.

$$\boldsymbol{Especificidad} = \frac{25910}{25910 + 61} = \frac{25910}{25971} = 0.99765$$

Por su parte, la especificidad logra un porcentaje de 99.76%, con esto se puede decir que el modelo tiene una muy alta capacidad para identificar las celdas UMTS que no tienen problema entre todas las celdas que están OK.

$$\boldsymbol{Precisión} = \frac{2054}{2054 + 61} = \frac{2054}{2115} = 0.97116$$

En cuanto a la precisión, también se logra un alto valor, lo que nos indica que existe una baja dispersión de los datos. Dicho de otro modo, se puede decir que existe una buena consistencia de los valores predichos por el modelo.

4.3.2 Desarrollo modelo 2 – Red Neuronal

En este segundo modelo es necesario definir un código binario asociado a cada una de las categorías que indican alguno de los tres problemas de optimización. Para facilitar la identificación se definen tres salidas y cada una de ellas se corresponde con cada una de las categorías. A continuación, se presentan las convenciones de codificación que se utilizaran para el desarrollo del modelo 2.

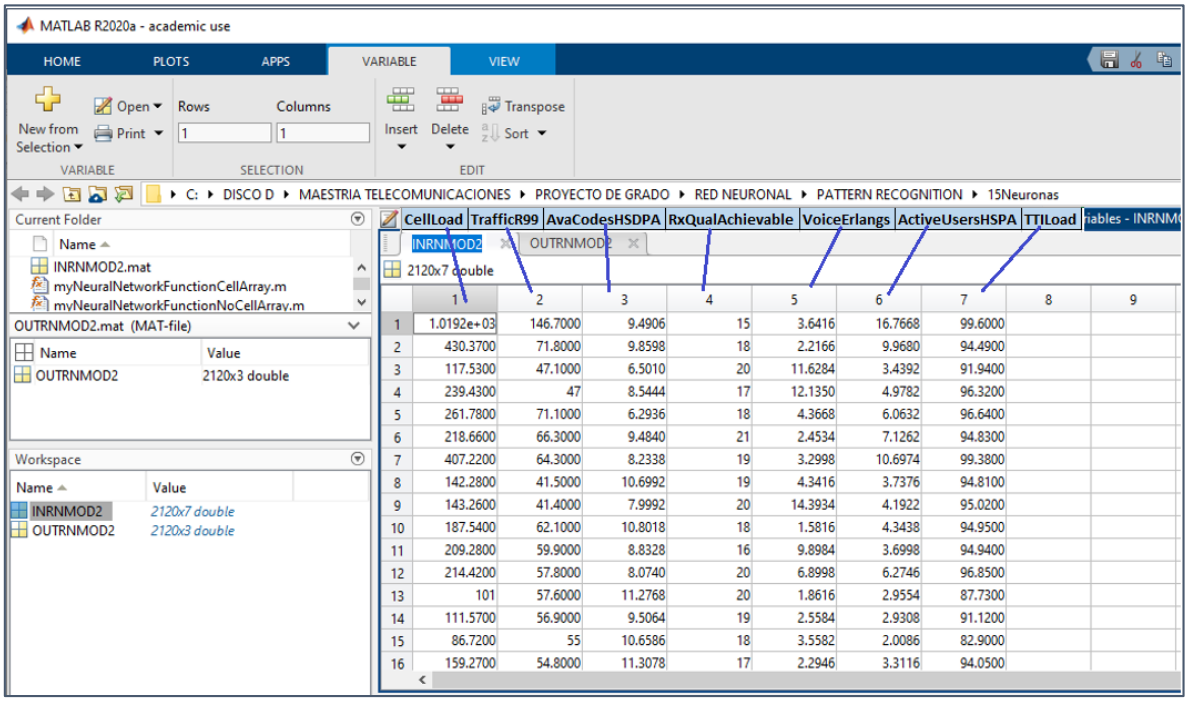
Tabla No. 4-5. Codificación salidas Modelo 2 Redes Neuronales. Fuente propia.

Modelo 2

Etiqueta	Código binario		
COV	0	0	1
CHECK	0	1	0
CAP	1	0	0

El modelo 2 se encargará de clasificar solo las celdas UMTS que se clasificaron como NOK en el primer modelo. Por lo tanto, en el modelo 2 se disminuyen sustancialmente el número de registros. Por otro lado, dado a que para este modelo se tienen tres categorías, es necesario definir tres salidas, una para cada categoría.

Para el entrenamiento de este modelo se toman las mismas variables de entrada que se consideraron en el modelo 1, pero para este modelo con un menor número de registros de ejemplo. La siguiente figura muestra la matriz de orden 2120x7 llamada INRNMOD2 la cual contiene todos los registros de ejemplos con las siete variables de entrada.



	1	2	3	4	5	6	7	8	9
1	1.0192e+03	146.7000	9.4906	15	3.6416	16.7668	99.6000		
2	430.3700	71.8000	9.8598	18	2.2166	9.9680	94.4900		
3	117.5300	47.1000	6.5010	20	11.6284	3.4392	91.9400		
4	239.4300	47	8.5444	17	12.1350	4.9782	96.3200		
5	261.7800	71.1000	6.2936	18	4.3668	6.0632	96.6400		
6	218.6600	66.3000	9.4840	21	2.4534	7.1262	94.8300		
7	407.2200	64.3000	8.2338	19	3.2998	10.6974	99.3800		
8	142.2800	41.5000	10.6992	19	4.3416	3.7376	94.8100		
9	143.2600	41.4000	7.9992	20	14.3934	4.1922	95.0200		
10	187.5400	62.1000	10.8018	18	1.5816	4.3438	94.9500		
11	209.2800	59.9000	8.8328	16	9.8984	3.6998	94.9400		
12	214.4200	57.8000	8.0740	20	6.8998	6.2746	96.8500		
13	101	57.6000	11.2768	20	1.8616	2.9554	87.7300		
14	111.5700	56.9000	9.5064	19	2.5584	2.9308	91.1200		
15	86.7200	55	10.6586	18	3.5582	2.0086	82.9000		
16	159.2700	54.8000	11.3078	17	2.2946	3.3116	94.0500		

Figura No. 4-42. Variables de entrada - Red Neuronal Modelo 2. Fuente propia.

Por otro lado, las variables de las tres salidas son cargadas en la matriz de orden 2120x3 llamada OUTRNMOD2 tal como lo muestra la siguiente imagen.

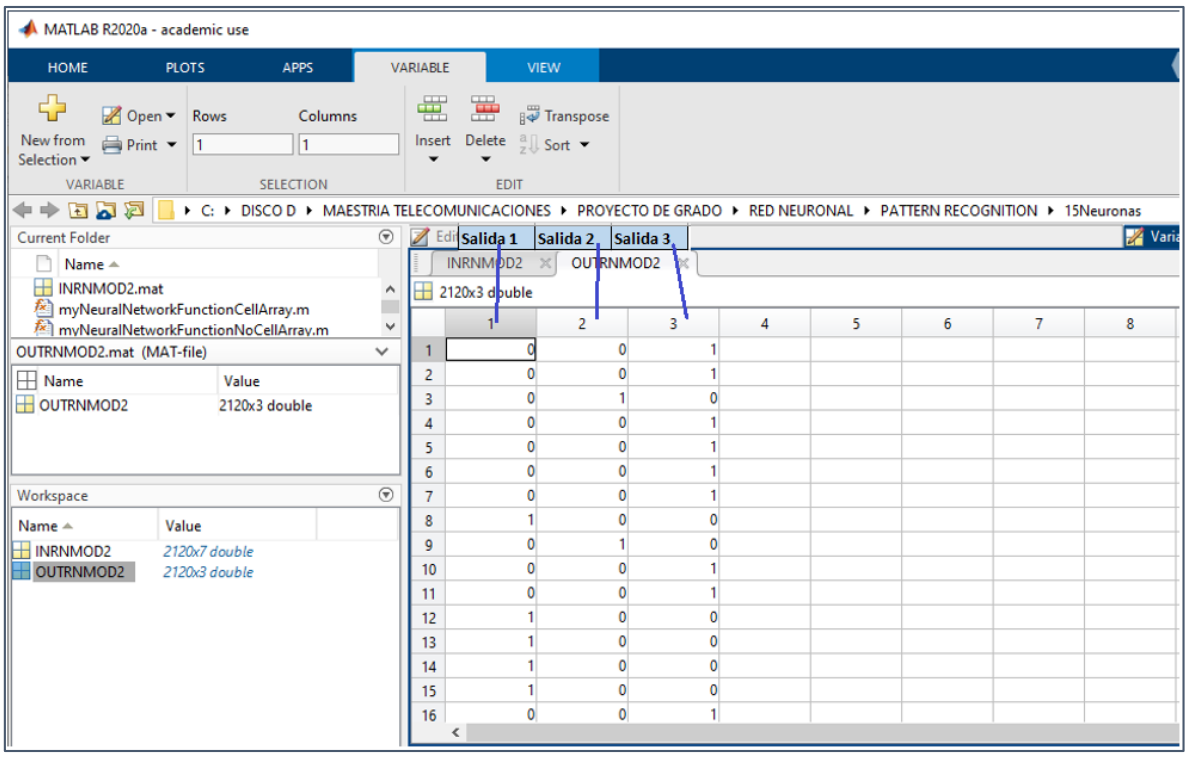


Figura No. 4-43. Variable de salida - Red Neuronal Modelo 2. Fuente propia.

El proceso de entrenamiento es similar al que se realizó para el modelo 1, se selecciona las variables de entrada, las de salida y la forma como están organizados los registros de ejemplo tal como se muestra en la siguiente figura.

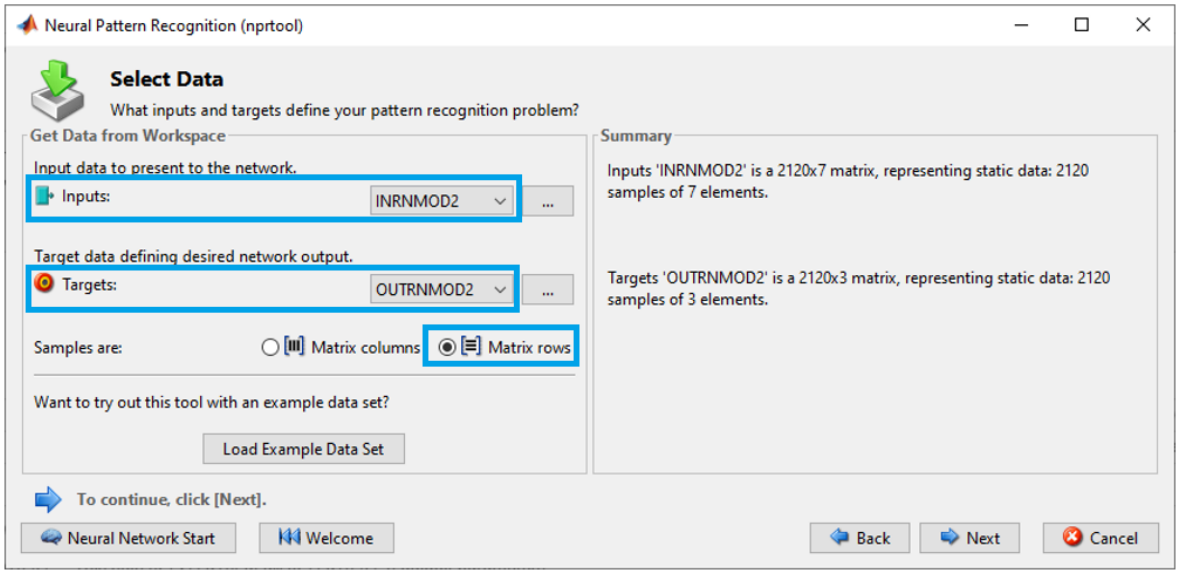


Figura No. 4-44. Configuración APP Neural Pattern Recognition – Modelo 2. Fuente propia.

Para este modelo, dado a que se tienen menos registros de ejemplo para el entrenamiento, se reducen un poco los porcentajes asignados para validación y prueba con el objetivo de aumentar los datos para la fase de Training. De esta manera la fase "Training" queda con un 80% de los datos de ejemplo y las fases "Validation" y "Testing" con un 10% cada una tal como lo muestra la siguiente imagen. Para el 80% de las muestras en la fase de "Training" le corresponden 1696 registros. Por su parte en las fases de "Validation" y "Testing" quedan con 212 registros de ejemplo.

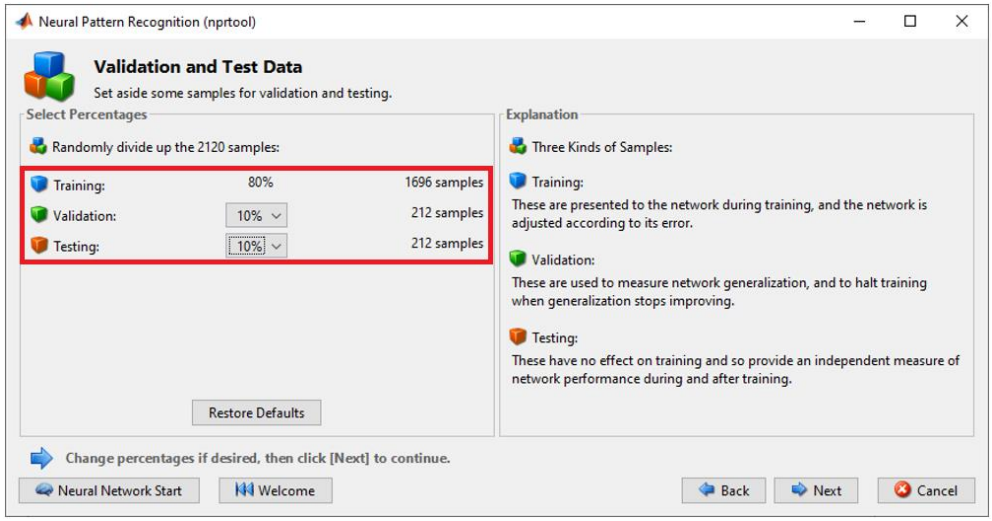


Figura No. 4-45. Asignación datos de entrenamiento, validación y prueba Modelo 2. Fuente propia.

Después de un ejercicio de pruebas del modelo con diferente número de neuronas en la capa oculta, se identifica que, una red con 19 neuronas es la opción con la cual mejor desempeño se logra para el modelo. La siguiente figura ilustra la arquitectura de red donde se identifican 7 neuronas para la capa de entrada y cuya función es la de recoger la información de todas las variables de entrada igual como ocurrió en el modelo 1, una neurona por variable. La siguiente es la capa oculta, la cual está compuesta por 19 neuronas y donde se fundamenta principalmente el modelo. Se continua con la capa de salida conformada por 3 neuronas, una para cada variable de salida. Finalmente se tiene la salía donde se entrega la predicción del modelo.

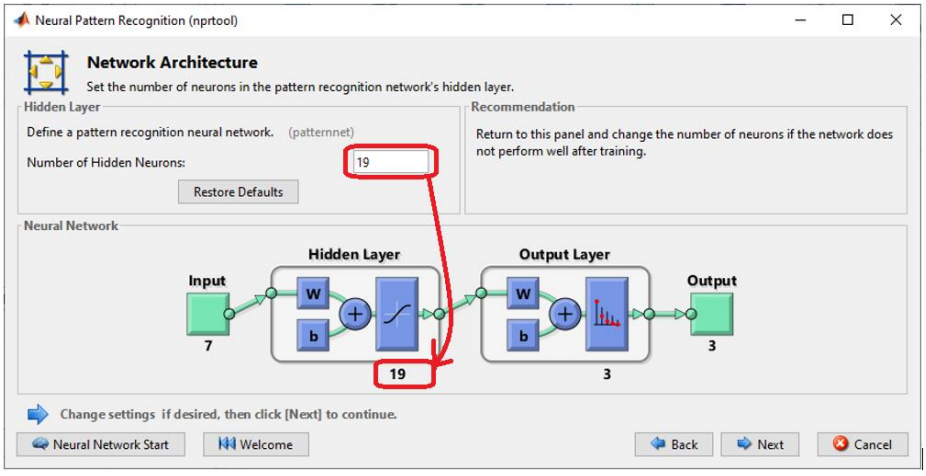


Figura No. 4-46. Arquitectura Red Neuronal – Modelo 2

Una vez definidos los parámetros de la red neuronal y ejecutar el proceso de entrenamiento, se obtienen los resultados de la siguiente figura. En esta ocasión se logra el entrenamiento en un total de 154 épocas. El tiempo que le toma al entrenamiento es tan bajo que no se alcanza a marca, ni siquiera un segundo.

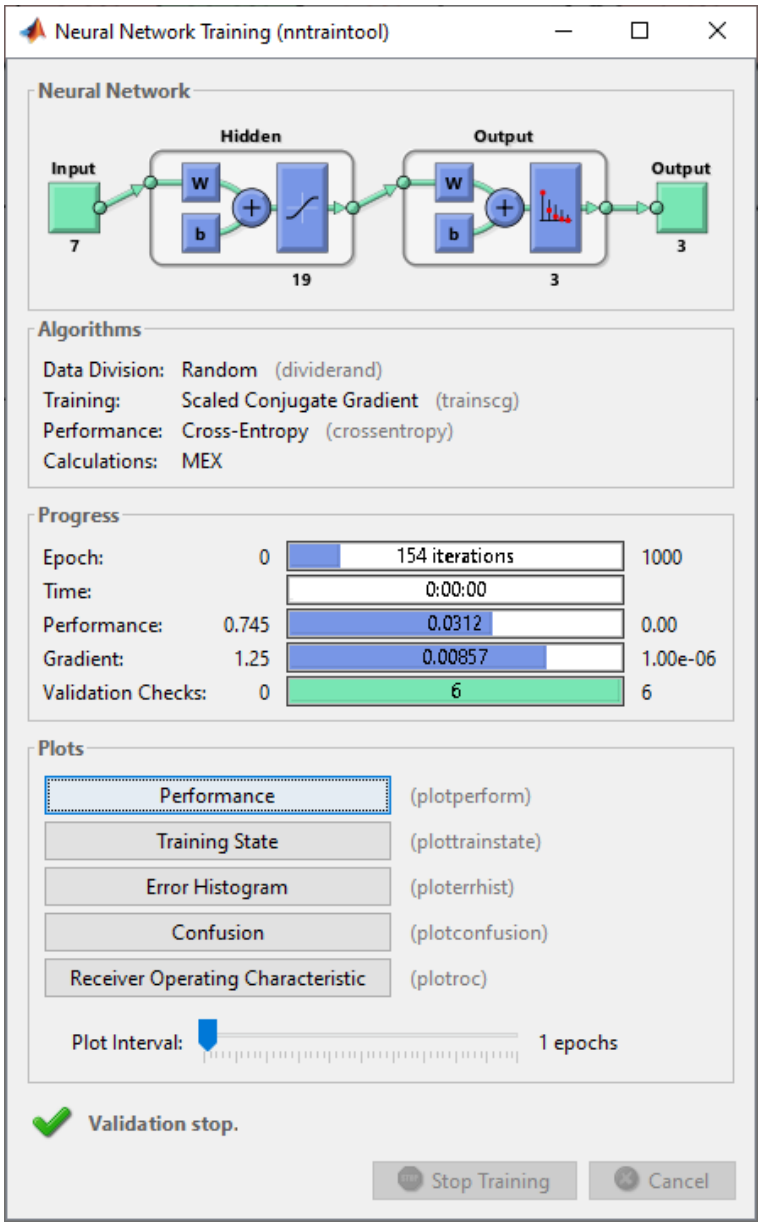


Figura No. 4-47. Resultados proceso de entrenamiento Red Neuronal – Modelo 2. Fuente propia.

La siguiente figura presenta la evolución que se obtuvo en la entropía cruzada para cada una de las fases del entrenamiento y la fase de prueba. El mejor desempeño de la entropía para la fase "Validation" fue de 0.034277 y se consiguió en la época 148. Al compararlo con el valor obtenido en el modelo 1 de redes neuronales, se puede ver que es más alto, por lo que se diría que el modelo 1 presento un mejor desempeño que el modelo 2. Las fases "Training" y "Validation" tiene una

evolución muy parecida, pero la fase "Testing" aunque la tendencia es similar, si existe un desfase sobre la curva que la lleva a un valor superior de entropía cruzada, esto quiere decir que tiene valores de error superiores a las fases de "Training" y "Validation".

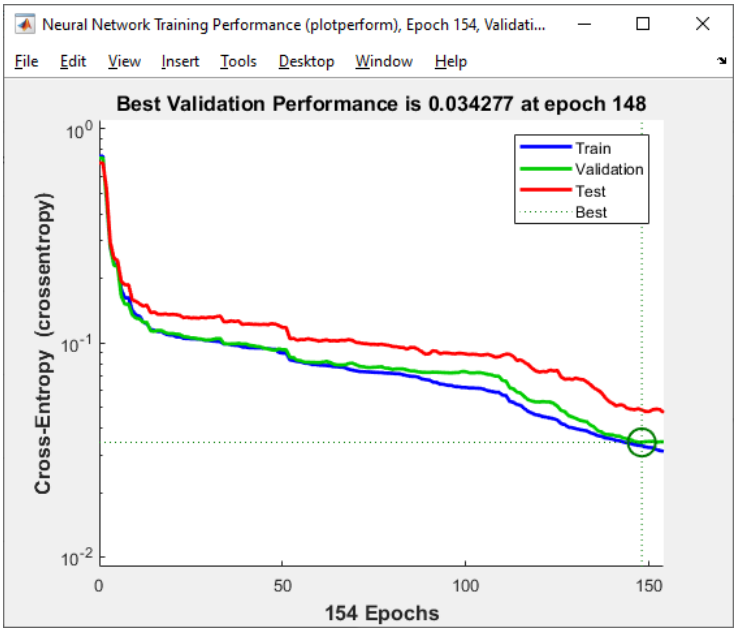


Figura No. 4-48. Entropía cruzada fases de entrenamiento Modelo 2. Fuente propia.

En cuanto al gradiente, se puede ver en la siguiente figura que tardo 154 épocas para converger. En las épocas intermedias se presentaron altibajos pero que solo llegaron hasta 4 incrementos consecutivos. Lo anterior nos indica que la función de pérdida o función de error tiene varios mínimos locales.

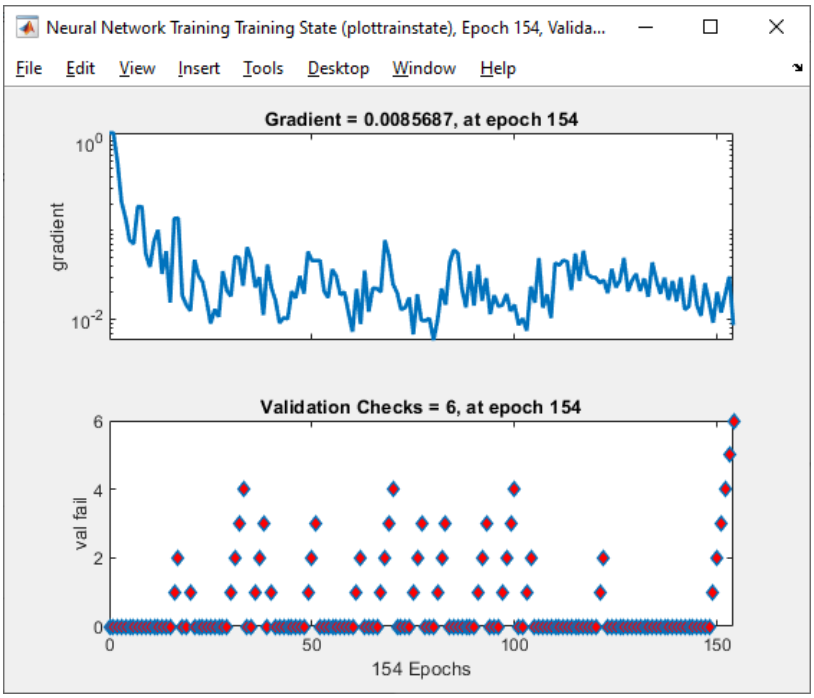


Figura No. 4-49. Evolución gradiente y controles de validación – Red Neuronal Modelo 2. Fuente propia.

Pasando al histograma de error que se entregó para este segundo modelo, se puede ver que la gran mayoría de los valores de error se encuentran muy cercanos al cerro error, la mayoría se concentran con valores de ±0.049. Otros muy pocos valores de error se encuentran entre valores de ±0.1497. En conclusión, se puede decir que en la gran mayoría de las predicciones la diferencia entre los valores predichos y los valores reales es muy pequeña.

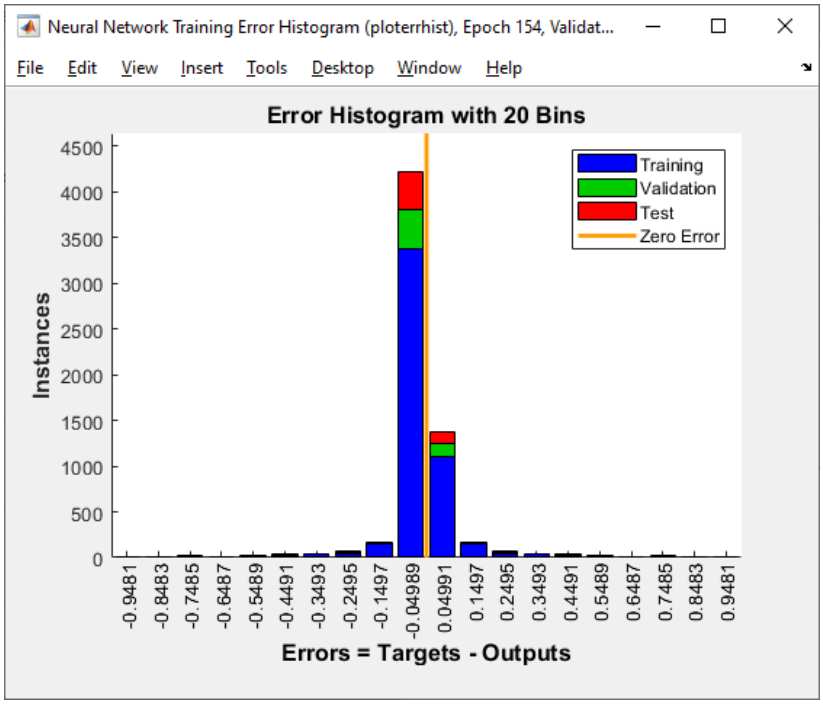


Figura No. 4-50. Histograma – Error en la predicción –Red Neuronal Modelo 2. Fuente propia.

A continuación, en la siguiente figura se presentan las curvas ROC y áreas bajo la curva AUC para cada una de las categorías en cada una de las fases del entrenamiento y fase de prueba. A cada categoría se le ha asignado una clase con su respectivo color tal como se indica en la siguiente tabla de convención.

Tabla No. 4-6. Asignación clases – Red Neuronal Modelo 2. Fuente propia.

Modelo 2

Etiqueta	Código binario			Clase
CAP	1	0	0	Clase 1
CHECK	0	1	0	Clase 2
COV	0	0	1	Clase 3

Las clases 1 y 3 que corresponden a las categorías CAP y COV respectivamente, presentan muy buenas curvas ROC con valores muy altos de tasa de verdaderos positivos y a la vez muy bajas tasas de falsos positivos en todas las fases del modelo. Esto quiere decir que el modelo logró clasificar adecuadamente la gran mayoría de los registros de ejemplos que pertenecen a estas clases. Por su parte las curvas ROC para la clase 2 que corresponde a la categoría CHECK, presenta una pequeña desmejora en las fases “Validation” y “Test”. Esto podría deberse a que esta clase tiene un muy bajo número de registros de ejemplos para entrenamiento del modelo comparado con las demás clases. Sin embargo, este resultado de la curva ROC se sigue considerando bueno.

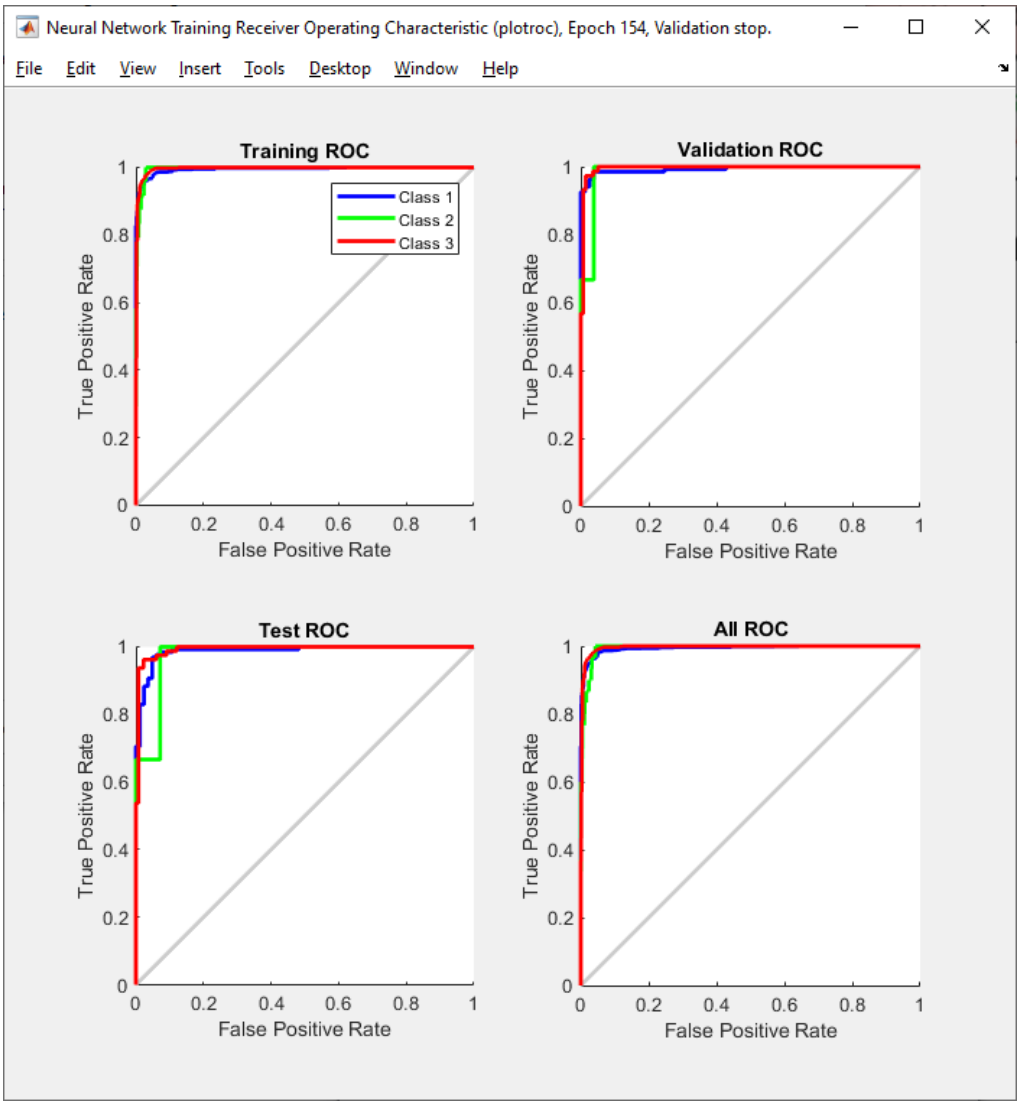


Figura No. 4-51. Curvas ROC y AUC Red Neuronal Modelo 2. Fuente propia.

4.3.2.1 Métricas matrices de confusión – Técnica Red Neuronal Modelo 2

Finalmente, en la siguiente figura se presentan las matrices de confusión para cada una de las fases del modelo. Igual como se realizó con el modelo 1 de red neuronal, se utilizarán los valores de la matriz de confusión acumulada ubicada en la parte inferior derecha para realizar los cálculos de cada una de las métricas que servirán para la evaluación del modelo.

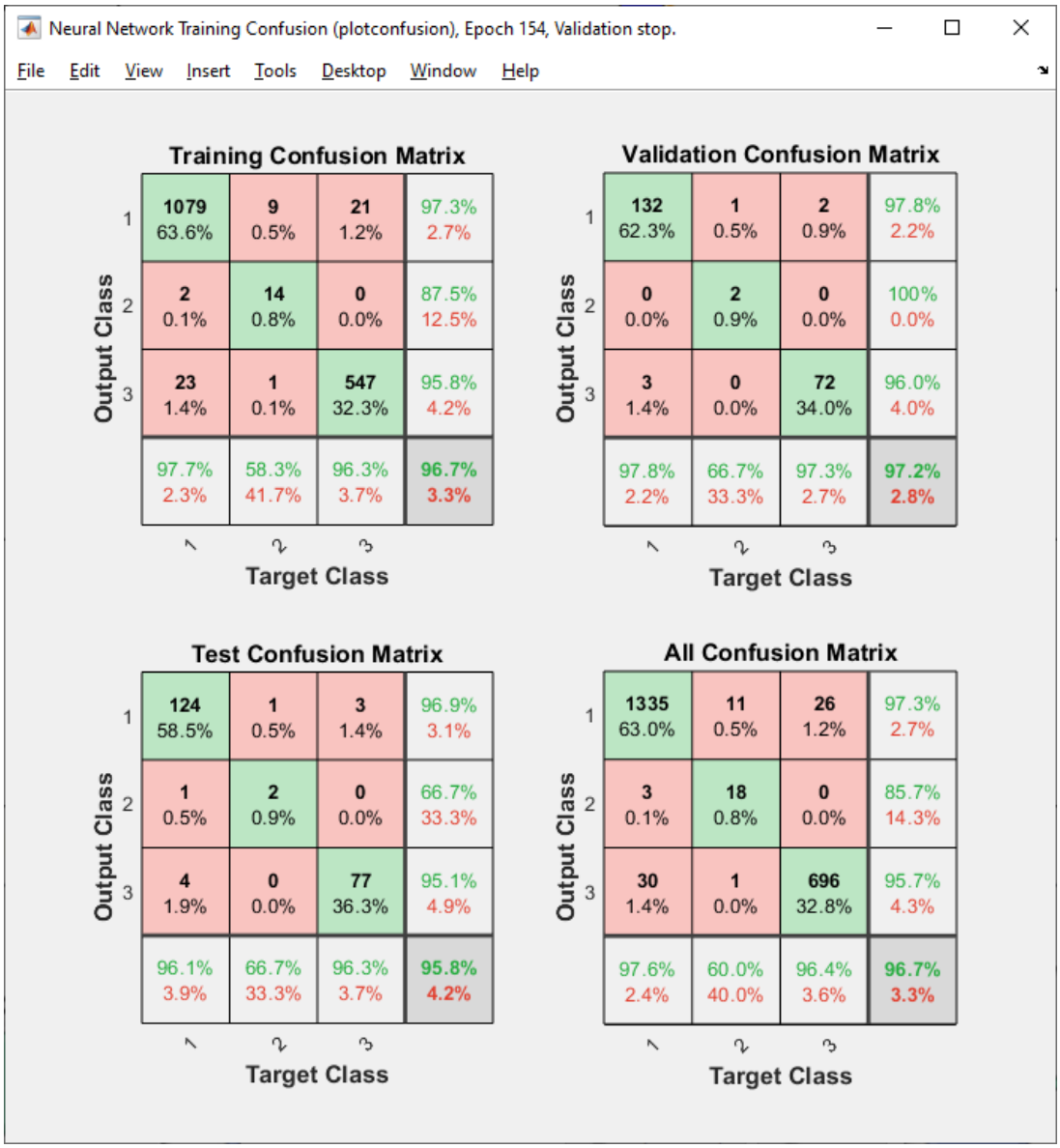


Figura No. 4-52. Matrices de confusión fases Red Neuronal Modelo 2. Fuente propia.

Pero antes de realizar los respectivos cálculos para cada una de las métricas, es necesario que se definan las ubicaciones de cada uno de los componentes de la matriz, lo anterior dado a que este modelo realiza una clasificación multiclase la cual es diferente a la matriz de confusión de una clasificación binaria. La siguiente figura muestra la distribución de los TP, TN, FP y FN de acuerdo a la organización dada por la APP de redes neuronales.

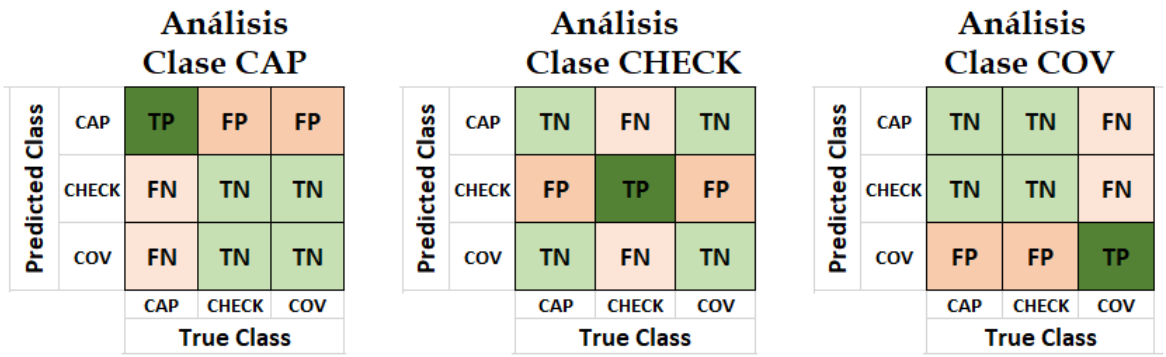


Figura No. 4-53. Matrices Multiclase CAP, CHECK y COV. Red Neuronal Modelo 2. Fuente propia.

A continuación, se realizan los cálculos de todas las métricas que nos permiten evaluar el modelo.

Para las métricas exactitud y tasa de error, el cálculo se realiza de la misma forma como se realiza para una clasificación binaria, es decir, considerando todos los aciertos para la exactitud y todos los fallos para la tasa de error. Esto dado a que se pueden considerar como métricas generales del modelo que involucra todas las clases.

$$\boldsymbol{Exactitud} = \frac{\sum TP_{Todas\ las\ clases}}{Total\ de\ Datos} = \frac{1335 + 18 + 696}{2120} = \frac{2049}{2120} = 0.96651$$

El modelo ofrece una buena exactitud ya que su valor es superior al 95%, con esto se tiene que los resultados de la clasificación de los registros están muy cerca del valor verdadero para cada una de sus clases CAP, COV y CHECK. En otras palabras, se puede decir que el modelo funciona adecuadamente para la gran mayoría de los registros.

$$\boldsymbol{Tasa\ de\ error} = \frac{\sum(FP + FN)_{\boldsymbol{Todas\ las\ clases}}}{Total\ de\ Datos} = \frac{3 + 30 + 11 + 1 + 26}{2120} = \frac{60}{2120} = 0.0283$$

La tasa de error obtenida en el entrenamiento de este modelo también es baja, con un valor muy cercano a 0%. Por lo tanto, se puede decir que el modelo tiene un muy bajo porcentaje de registros clasificados incorrectamente en todo el conjunto de clases CAP, COV y CHECK.

$$\boldsymbol{Sensibilidad\ CAP} = \frac{TP}{TP + FN} = \frac{1335}{1335 + 3 + 30} = \frac{1335}{1368} = 0.97588$$

$$\boldsymbol{Sensibilidad\ COV} = \frac{VP}{TP + FN} = \frac{696}{696 + 26} = \frac{696}{722} = 0.96399$$

$$\boldsymbol{Sensibilidad\ CHECK} = \frac{VP}{TP + FN} = \frac{18}{18 + 11 + 1} = \frac{18}{30} = 0.6$$

Para este modelo 2 dado a que se definen tres posibles categorías de clasificación, es necesario realizar el cálculo de la sensibilidad de forma separada para cada una de las clases. Se puede decir que los valores obtenidos son bastante altos para las clases CAP y COV con porcentajes superiores al 96%. Por lo anterior se puede concluir que el modelo tiene una aceptable capacidad para discriminar los casos de celdas UMTS con problemas de capacidad y cobertura que son etiquetados

como CAP y COV respectivamente. Para el caso de la clase CHECK la sensibilidad solo alcanza el 60%, esta clase es la que peor desempeño tiene de las tres, su desventaja es la cantidad muy reducida de registros de ejemplo que disponen para el entrenamiento. Explicado de otra manera, se puede decir que por el bajo número de registros que tiene la categoría CHECK, cualquier fallo que se de en las fases de entrenamiento representa un alto porcentaje que se descuenta en la sensibilidad.

$$\boldsymbol{Especificidad\ CAP} = \frac{TN}{TN + FP} = \frac{18 + 1 + 696}{18 + 1 + 696 + 11 + 26} = \frac{715}{752} = 0.950798$$

$$\boldsymbol{Especificidad\ COV} = \frac{TN}{TN + FP} = \frac{1335 + 3 + 11 + 18}{1335 + 3 + 11 + 18 + 26} = \frac{1367}{1411} = 0.968816$$

$$\boldsymbol{Especificidad\ CHECK} = \frac{TN}{TN + FP} = \frac{1335 + 30 + 26 + 696}{1335 + 30 + 26 + 696 + 11 + 1} = \frac{2087}{2099} = 0.994283$$

Para la métrica de la especificidad también es necesario realizar el cálculo de forma separada para cada una de las clases que se pueden predecir con el modelo. El porcentaje que se logra en las tres clases es alto, superior al 99.6%. Con esto se puede decir que el modelo tiene en general una muy alta capacidad para identificar las celdas UMTS que no pertenecen a una determinada clase.

$$\boldsymbol{Precisión\ CAP} = \frac{TP}{TP + FP} = \frac{1335}{1335 + 11 + 26} = \frac{1335}{1372} = 0.97303$$

$$\boldsymbol{Precisión\ COV} = \frac{TP}{TP + FP} = \frac{696}{696 + 30 + 1} = \frac{696}{724} = 0.95736$$

$$\boldsymbol{Precisión\ CHECK} = \frac{TP}{TP + FP} = \frac{18}{18 + 3} = \frac{18}{21} = 0.85714$$

La precisión que se logra para las clases CAP y COP es alta, es decir que se presentaron muy pocas fallas en las que al final se abrían clasificado como CAP o COP sin pertenecer realmente a estas categorías. Por otro lado, la precisión lograda para la clase CHECK es menor, pero aun aceptable para el modelo. Como se puede ver, el número de FP para esta categoría es de solo 3, y aunque es un numero bajo, si representa un alto porcentaje por el bajo número de registros que pertenecen a la clase. Al tener pocos registros en esta clase, cualquier dispersión en los datos obtenidos genera un impacto en el cálculo de la precisión.

5. ANÁLIS RESULTADOS MODELOS DE MACHINE LEARNING

5.1 COMPARACIÓN DE LOS MODELOS DE CLASIFICACIÓN

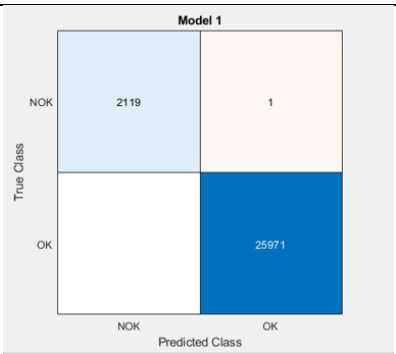
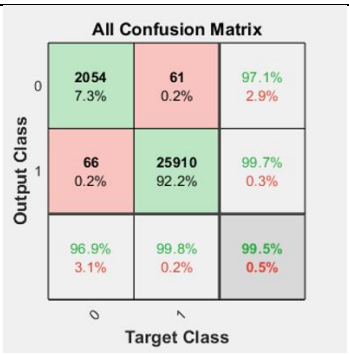
Una vez desarrollados todos los modelos para la clasificación de las celdas UMTS en cada una de las categorías de acuerdo al estatus dado por sus variables de entrada, se procede en este capítulo a cotejar los resultados obtenidos con cada técnica y de esta manera seleccionar el mejor modelo con el cual se puedan diagnosticar las celdas para que ayude en el proceso de optimización de red UMTS.

Para el propósito de comparar los modelos, se utilizarán los resultados obtenidos en los cálculos de cada una de las métricas del capítulo anterior, las cuales son: precisión, tasa de error, sensibilidad, especificidad y precisión.

5.1.1 Comparación modelo 1

La siguiente tabla presenta de forma consolidada todos los resultados de cada una de las métricas tanto para la técnica árbol de decisión en el lado izquierdo como para la técnica de red neuronal en el lado derecho. Este primer modelo se encargará de clasificar las celdas UMTS en las categorías OK y NOK.

Tabla No. 5-1. Comparación de métricas - Técnicas de clasificación modelo 1. Fuente propia.

Árbol de Decisión	Red Neuronal
	
$\boldsymbol{Exactitud} = \frac{2119 + 25971}{28091} = 0.999964$ $\boldsymbol{Tasa\ de\ error} = \frac{1 + 0}{28091} = 0.000035598$ $\boldsymbol{Sensibilidad} = \frac{2119}{2119 + 1} = 0.999528$ $\boldsymbol{Especificidad} = \frac{25971}{25971 + 0} = 1$	$\boldsymbol{Exactitud} = \frac{2054 + 25910}{28091} = 0.995479$ $\boldsymbol{Tasa\ de\ error} = \frac{61 + 66}{28091} = 0.004521$ $\boldsymbol{Sensibilidad} = \frac{2054}{2054 + 66} = 0.968868$ $\boldsymbol{Especificidad} = \frac{25910}{25910 + 61} = 0.99765$

$\boldsymbol{Precisión} = \frac{2119}{2119 + 0} = 1$	$\boldsymbol{Precisión} = \frac{2054}{2054 + 61} = 0.97116$

Como se explicó en el capítulo anterior, la métrica de la exactitud se refiere a lo cerca que esta una predicción del valor real. En esta métrica se entrega la porción de datos que son clasificados correctamente respecto del total de datos. No se consideran las clases de manera independiente, sino que de manera global toma todas las clasificaciones correctas que se lograron predecir. Para esta métrica la técnica de árbol de decisión presenta mejor desempeño con un valor de 0.999964 comparado con 0.995479 de la técnica red neuronal.

Tabla No. 5-2. Ranking métrica Exactitud Modelo 1. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Exactitud	0.999964	0.995479

Pasando a la métrica tasa de error, en la cual de manera global se calcula la porción de predicciones fallidas del total de predicciones. Al igual que en la métrica anterior, en el cálculo no se consideran los valores independientes de cada clase, sino que se agrupan todos los fallos para sacar la porción general de errores de predicción. En esta métrica se busca el valor más bajo posible cercano a cero. Al validar los datos obtenidos con las dos técnicas, se observa que la técnica árbol de decisión presenta mejor desempeño que el modelo obtenido con la red neuronal. El valor de 0.000035598 obtenido con el modelo de la técnica árbol de decisión es muy bueno y está bastante cercano al cero ideal. Por su parte el valor de 0.004521 obtenido con el modelo de la red neuronal, aunque es pequeño no es tan bajo como el del otro modelo.

Tabla No. 5-3. Ranking métrica Tasa de Error Modelo 1. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Tasa de Error	0.000035598	0.004521

La siguiente métrica es la sensibilidad que ofrece el modelo o también llamada “Recall”, se enfoca en determinar la tasa de verdaderos positivos (TPR). Los valores de esta métrica van de cero a uno y se toma el valor de 1 como el ideal. De los cálculos obtenidos con las dos técnicas se puede ver que el modelo del árbol de decisión tiene una mejor capacidad para discriminar los casos positivos de los casos negativos en comparación con el modelo de la red neuronal. Mientras que el árbol de decisión logra un 0.999528, la red neuronal llega a un valor de 0.968868. La diferencia entre los dos valores es de aproximadamente 3 puntos porcentuales.

Tabla No. 5-4. Ranking métrica Sensibilidad Modelo 1. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Sensibilidad	0.999528	0.968868

La métrica especificidad por su parte se encarga de medir la tasa de los verdaderos negativos (TNR) entre todos los que se clasificaron como negativos. Su valor va de cero a uno donde 1 es el valor deseado. Los valores obtenidos por las dos técnicas fueron muy buenos, pero se destaca la técnica árbol de decisión por lograr un valor perfecto de 1, es decir que su modelo tiene una muy alta capacidad para clasificar correctamente los casos negativos.

Tabla No. 5-5. Ranking métrica Especificidad Modelo 1. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Especificidad	1	0.99765

Finalmente se tiene la métrica precisión. Es importante no confundir la precisión con la exactitud. Pues mientras que la exactitud se refiere a la cercanía que están los valores predichos de los valores reales, la precisión se refiere a la dispersión de los valores obtenidos en la predicción. Para entender mejor el concepto de estas dos métricas, a continuación, en la siguiente figura se presentan los cuatro posibles escenarios en los que se combinan la exactitud y la precisión. En el primer ejemplo los datos están dispersos y alejados del centro, por lo tanto, no se tiene ni precisión ni exactitud. En el segundo caso, los datos están concentrados y por lo tanto existe precisión o consistencia de los datos, sin embargo, como están alejados del centro que sería el valor real, entonces no se tiene exactitud. En el siguiente caso los datos están dispersos, pero alrededor del centro o valor real, con esto se puede decir que se logra exactitud, pero no precisión. Finalmente, en el cuarto caso los datos están concentrados y alrededor del centro lográndose tanto la exactitud como la precisión.

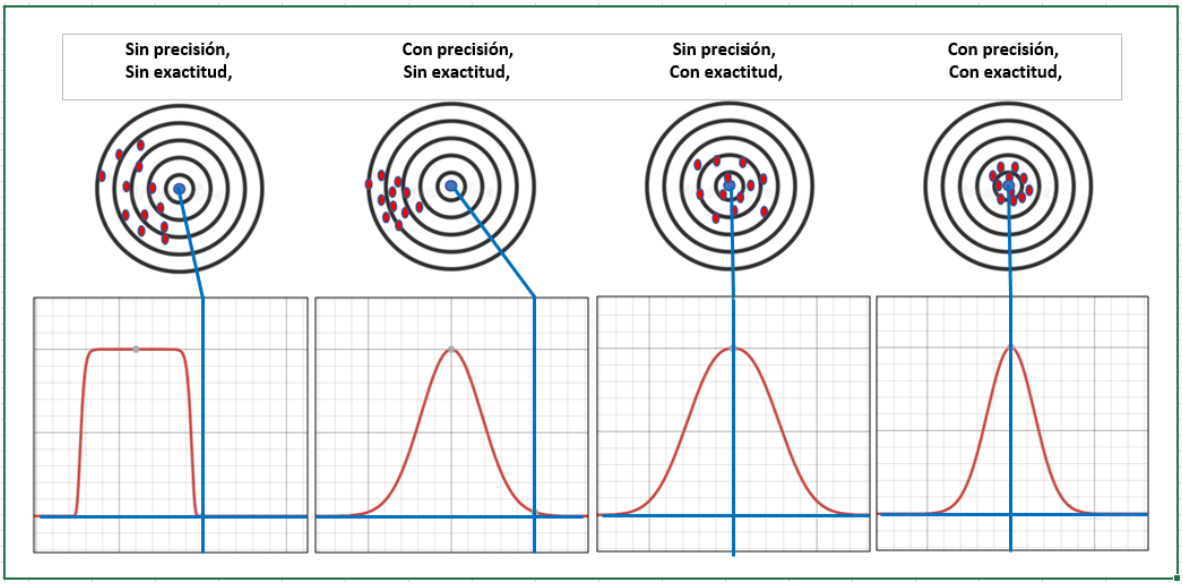


Figura No. 5-1. Escenarios de Exactitud y Precisión. Fuente propia.

En relación a la métrica precisión se puede ver que la técnica árbol de decisión presenta un desempeño perfecto, ya que su cálculo da un valor de 1. Con esto se puede decir que los datos se encuentran concentrados sin ninguna dispersión. Por su parte, el valor de la precisión conseguido

por la red neuronal el cual es de 0.97116 se considera bueno, pero es inferior al obtenido con la otra técnica.

Tabla No. 5-6. Ranking métrica Precisión Modelo 1. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Precisión	1	0.97116

Una métrica que evalúa el modelo de una forma más integral es la llamada "Medida F1", esta se aplica solo a los modelos de clasificación binaria, esta métrica se considera una media armónica en la que se combinan las métricas sensibilidad y precisión. Cuando los valores de la Medida F1 son cercanos a 1, se puede decir que el modelo cumple con el propósito de clasificación. A continuación, la expresión que se utiliza para el cálculo.

$$\boldsymbol{Medida\ F1} = 2 * \frac{Sensitividad * Precisión}{Senitividad + Precisión}$$

Aplicando esta expresión con los datos de cada una de las técnicas, se obtienen los siguientes resultados.

$$\boldsymbol{Medida\ F1\ Árbol\ de\ Decision} = 2 * \frac{0.9999528 * 1}{0.999528 + 1} = \frac{1.999056}{1.999528} = 0.9997639$$

$$\boldsymbol{Medida\ F1\ Red\ Neuronal} = 2 * \frac{0.968868 * 0.97116}{0.968868 + 0.97116} = \frac{1.8818516}{1.940028} = 0.970125$$

De esta manera se puede decir que de acuerdo a la métrica Medida F1, el modelo perteneciente a la técnica árbol de decisión cumple mucho mejor el propósito de clasificación que el otro modelo de la red neuronal.

Tabla No. 5-7. Ranking métrica Score F1 Modelo 1. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Valor F1	0.999763944	0.97001265

Definitivamente el modelo generado con la técnica de árbol de decisión es la que presenta mejor comportamiento y rendimiento para la clasificación de las celdas UMTS en las categorías OK y NOK. El modelo generado por el árbol de decisión supero en todas las métricas al modelo generado por la red neuronal. La siguiente tabla consolida todos los resultados y nos permite concluir que el modelo escogido para la primera clasificación de las celdas en OK y NOK se realizará con el modelo de la técnica árbol de decisión.

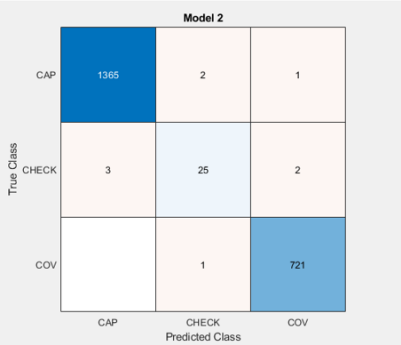
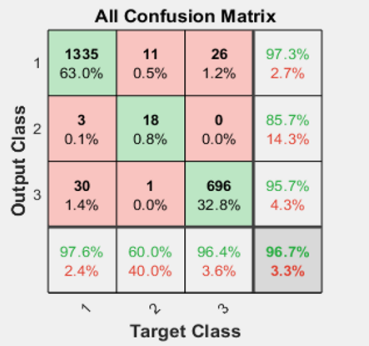
Tabla No. 5-8. Consolidado ranking métricas – Técnicas de Clasificación Modelo 1. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Exactitud	0.999964	0.995479
Tasa de Error	0.000035598	0.004521
Sensibilidad	0.999528	0.968868
Especificidad	1	0.99765
Precisión	1	0.97116
Valor F1	0.999763944	0.97001265

5.1.2 Comparación modelo 2

La siguiente tabla presenta de forma consolidada todos los resultados de cada una de las métricas tanto para la técnica árbol de decisión en el lado izquierdo como para la técnica de red neuronal en el lado derecho. Este segundo modelo se encargará de clasificar las celdas UMTS que resultaron del primer modelo en NOK en las tres categorías COP, CAP y CHECK.

Tabla No. 5-9. Comparación de métricas - Técnicas de clasificación modelo 2. Fuente propia.

Árbol de Decisión	Red Neuronal
	
$\boldsymbol{Exactitud} = \frac{1365 + 25 + 721}{2120} = 0.995754$	$\boldsymbol{Exactitud} = \frac{1335 + 18 + 696}{2120} = 0.96651$
$\boldsymbol{Tasa\ de\ error} = \frac{3 + 2 + 2 + 1 + 1}{2120} = 0.00424583$	$\boldsymbol{Tasa\ de\ error} = \frac{3 + 30 + 11 + 1 + 26}{2120} = 0.0283$
$\boldsymbol{Sensibilidad\ CAP} = \frac{1365}{1365 + 2 + 1} = 0.9978$	$\boldsymbol{Sensibilidad\ CAP} = \frac{1335}{1335 + 3 + 30} = 0.97588$
$\boldsymbol{Sensibilidad\ COV} = \frac{721}{721 + 1} = 0.9986$	$\boldsymbol{Sensibilidad\ COV} = \frac{696}{696 + 26} = 0.96399$
$\boldsymbol{Sensibilidad\ CHECK} = \frac{25}{25 + 3 + 2} = 0.8333$	$\boldsymbol{Sensibilidad\ CHECK} = \frac{18}{18 + 11 + 1} = 0.6$
$\boldsymbol{Especificidad\ CAP} = \frac{25 + 721 + 2 + 1}{25 + 721 + 2 + 1 + 3} = 0.99601$	$\boldsymbol{Especificidad\ CAP} = \frac{18 + 1 + 696}{18 + 1 + 696 + 11 + 26} = 0.950798$
	$\boldsymbol{Especificidad\ COV} = \frac{1335 + 3 + 11 + 18}{1335 + 3 + 11 + 18 + 26} = 0.968816$

$$\textbf{\textit{Especificidad COV}} = \frac{1365+25+3+2}{1365+25+3+2+2+1} = 0.99785$$	$$\textbf{\textit{Especificidad CHECK}} = \frac{1335+30+26+696}{1335+30+26+696+11+1} = 0.994283$$
$$\textbf{\textit{Especificidad CHECK}} = \frac{1365+721+1}{1365+721+1+2+1} = 0.99856$$	$$\textbf{\textit{Precisión CAP}} = \frac{1335}{1335+11+26} = 0.97303$$
$$\textbf{\textit{Precisión CAP}} = \frac{1365}{1365+3} = 0.9978$$	$$\textbf{\textit{Precisión COV}} = \frac{696}{696+30+1} = 0.95736$$
$$\textbf{\textit{Precisión COV}} = \frac{721}{721+2+1} = 0.9958$$	$$\textbf{\textit{Precisión CHECK}} = \frac{18}{18+3} = 0.85714$$
$$\textbf{\textit{Precisión CHECK}} = \frac{25}{25+2+1} = 0.8928$$	

En este segundo modelo la clasificación que se realiza ya no es binaria, sino que es multiclase, al existir más de dos categorías es preciso calcular el desempeño de los modelos generados con las técnicas para cada una de las categorías. Las dos primeras métricas exactitud y tasa de error realizan la evaluación del modelo de manera general, pero para el resto de métricas se realiza el cálculo de forma independiente para cada categoría.

Para la métrica exactitud se logra un mejor desempeño con el modelo de la técnica árbol de decisión en comparación con el modelo de la red neuronal. Pues, aunque los valores obtenidos con los dos modelos son buenos, el valor del modelo de árbol de decisión lo supera al otro en más de 2.9 puntos porcentuales.

Tabla No. 5-10. Ranking métrica Exactitud Modelo 2. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Exactitud	0.995754	0.96651

Igual que la anterior métrica, la tasa de error es una evaluación general de los modelos sin importar si la clasificación es binaria o multiclase. El modelo generado con la técnica árbol de decisión continúa siendo el de mejor desempeño, su valor es inferior al de la red neuronal en más de 2 puntos porcentuales con un valor de 0.00424583. Se puede decir que los errores de clasificación que presenta el modelo de la técnica árbol de decisión son muy bajos.

Tabla No. 5-11. Ranking métrica Tasa de Error Modelo 2. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Tasa de Error	0.00424583	0.0283

La siguiente métrica en analizar es la sensibilidad que tienen los modelos para detectar los verdaderos positivos en cada una de las categorías. El modelo que mejor desempeño tiene en relación a la sensibilidad es el modelo de la técnica árbol de decisión. Pues en las tres categorías se superan los valores obtenidos comparados con los modelos de la red neuronal. En las categorías

CAP y COV las diferencias son pequeñas, pero en la categoría CHECK la diferencia entre una técnica y otra es de más de 23 puntos porcentuales, lo cual es muy considerable.

Tabla No. 5-12. Ranking métrica Sensibilidad Modelo 2. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Sensibilidad CAP	0.9978	0.97588
Sensibilidad COV	0.9986	0.96399
Sensibilidad CHECK	0.8333	0.6

Pasando a la métrica especificidad, se puede ver en la siguiente tabla que los valores obtenidos en las categorías no se presentan grandes diferencias, con lo cual se podría decir que los modelos de las dos técnicas ofrecen rendimientos similares. Sin embargo, el modelo de la técnica árbol de decisión es superior al modelo de la red neuronal para las tres categorías CAP, COV y CHECK en 4.5%, 2.9% y 0.09% respectivamente.

Tabla No. 5-13. Ranking métrica Especificidad Modelo 2. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Especificidad CAP	0.99601	0.950798
Especificidad COV	0.99785	0.968816
Especificidad CHECK	0.99856	0.99765

Continuando con la métrica de precisión, la cual nos indica como es la dispersión de los datos en cada una de las categorías, se puede apreciar que el rendimiento del modelo árbol de decisión sigue siendo superior al modelo de red neuronal. Las diferencias no son grandes, pero si sustanciales para inclinarse por el primer modelo.

Tabla No. 5-14. Ranking métrica Precisión Modelo 2. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Precisión CAP	0.9978	0.97303
Precisión COV	0.9958	0.95736
Precisión CHECK	0.8928	0.85714

Finalmente, se evalúa la métrica combinada Medida F1 en cada una de las categorías. De manera consistente el modelo derivado de la técnica árbol de decisión presenta un mejor desempeño en todas las categorías. En especial para la clase CHECK el primer modelo es superior en 15.61 puntos porcentuales.

Tabla No. 5-15. Ranking métrica Valor F1 Modelo 2. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Valof F1 CAP	0.9978	0.97445292
Valof F1 COV	0.997198034	0.96066356
Valof F1 CHECK	0.862024495	0.70588138

De forma contundente el modelo generado por la técnica árbol de decisión supera al modelo de la red neuronal, lo anterior puesto que, en todas las métricas de evaluación, este modelo presento un mejor desempeño, en algunas métricas la diferencia entre los valores de las métricas fue pequeño, pero en otras fue considerable. Y aunque en una solución se podrían combinar modelos de varias técnicas, en este caso la técnica de árbol de decisión es la apropiada tanto para el modelo 1 que clasifica las celdas UMTS en OK y NOK como para el modelo 2 que clasifica las celdas en las categorías CAP, COV y CHECK. La siguiente tabla muestra de manera consolidada todos los valores obtenidos en las diversas métricas de evaluación.

Tabla No. 5-16. Consolidado Métricas – Técnicas de Clasificación Modelo 2. Fuente propia.

Métrica	Árbol de Decisión	Red Neuronal
Exactitud	0.995754	0.966510
Tasa de Error	0.004246	0.028300
Sensibilidad CAP	0.997800	0.975880
Sensibilidad COV	0.998600	0.963990
Sensibilidad CHECK	0.833300	0.600000
Especificidad CAP	0.996010	0.950798
Especificidad COV	0.997850	0.968816
Especificidad CHECK	0.998560	0.997650
Precisión CAP	0.997800	0.973030
Precisión COV	0.995800	0.957360
Precisión CHECK	0.892800	0.857140
Valof F1 CAP	0.997800	0.974453
Valof F1 COV	0.997198	0.960664
Valof F1 CHECK	0.862024	0.705881

5.2 ESQUEMA FINAL DIAGNÓSTICO CELDAS UMTS

La siguiente figura muestra como funcionaria todo el proceso de diagnóstico de las celdas UMTS utilizando los dos modelos de árbol de decisión. Se inicia con el data set que contiene las siete variables de entrada y que son utilizadas por el primer modelo para clasificar las celdas en las categorías OK y NOK. Las celdas que se etiquetan como “NOK” son las que tienen algún problema y por lo tanto requerirán un trabajo de optimización. En el siguiente paso y tomando únicamente los registros que fueron clasificados como NOK, se ingresan al segundo modelo para clasificar a las celdas en uno de los tres posibles problemas de optimización que son CAP, COV y CHECK. Es importante indicar que tanto el primer modelo como el segundo modelo están basados en la técnica de árbol de decisión. De esta manera se realiza un completo diagnóstico de las celdas UMTS, primero identificando las celdas que están bien y las que están mal, y posterior sobre las celdas que están mal asociándoles uno de los tres posibles problemas de optimización.

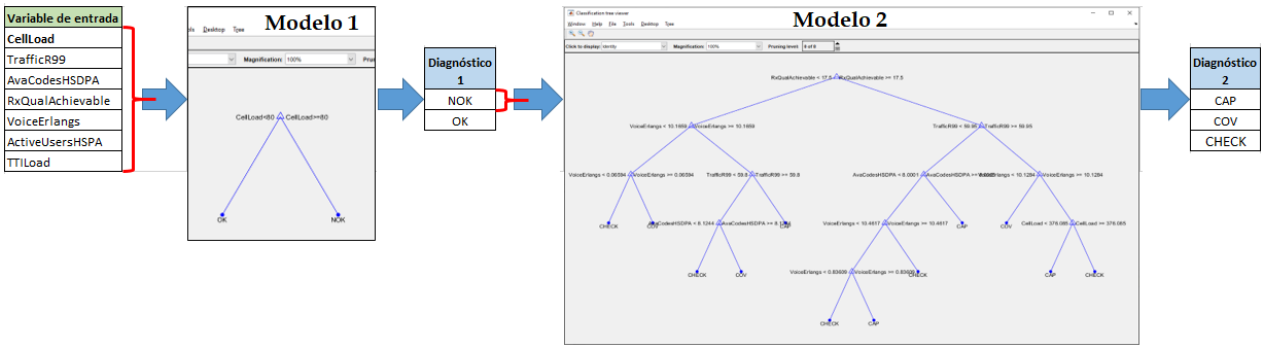


Figura No. 5-2. Proceso diagnóstico de celdas UMTS. Fuente propia.

5.3 TIEMPOS DE PROCESAMIENTO

Una parte importante en los procesos de Machine Learning tiene que ver con los tiempos de entrenamiento que le toma a la aplicación para generar el modelo en cualesquiera de las técnicas. El objetivo de este punto no es realizar un análisis profundo de todas las condiciones que afectan los tiempos de entrenamiento de cada uno de los modelos, sino poder validar si las condiciones y características con las que se aborda el proceso de aprendizaje demandan recursos de computo especiales. Existen varios factores que afectan directamente los tiempos invertidos para el entrenamiento entre los que se pueden destacar los siguientes.

- La capacidad de computo donde se ejecuta el proceso de entrenamiento.

Dentro de estos factores se destacan el tipo de procesador y la cantidad de memoria. Es evidente que entre más poderoso sea el procesador del equipo menor será el tiempo invertido para el entrenamiento del modelo. Para el desarrollo de estos modelos se utilizó un equipo de cómputo con las siguientes características.

Sistema Operativo: Windows 10 Home

Procesador: Intel(R) Core(TM) i5-1035G7 CPU @ 1.20GHz 1.50 GHz

Memoria RAM: 8,00 GB (7,77 GB utilizable)

Como se verá más adelante, un equipo con estas características es suficiente para el entrenamiento y desarrollo de los modelos.

- Cantidad de registros de ejemplo considerados para el entrenamiento.

El tamaño del data set tanto en variables de los objetos como en la cantidad de registros de ejemplo influyen de alguna manera en los tiempos que se invierten para el proceso de entrenamiento de los modelos. Para este caso, el problema se dividió en dos modelos, el primero utiliza un data set de 28091 registros con 8 variables, y en el segundo modelo un data set de 2120 registros también con 8 variables. Aunque el tamaño de los data set son grandes del orden de los miles, el tiempo de procesamiento fue muy bajo en cada uno de los modelos.

- Configuración de los parámetros de entrenamiento en cada una de las técnicas.

Dependiendo de la técnica utilizada para generar el modelo, se establecen ciertos parámetros que determinan como se produce el entrenamiento y con ello el tiempo que le tomará ejecutarlo. Por ejemplo, en la técnica árbol de decisión se tienen varias opciones en función del tipo de árbol. Para este proyecto se escogió la opción de "Fine Tree" con la cual se consigue mayor número de ramificaciones y hojas terminales, obviamente esta opción es la que más tiempo en el proceso de entrenamiento demanda al tener una mayor cantidad de combinaciones que probar.

Por el lado de la red neuronal, la cantidad de neuronas en la capa oculta es una característica modificable que de acuerdo a su configuración demandará más o menos tiempo durante el proceso de entrenamiento. Cada neurona que se adicione implicaran nuevas conexiones en la red y por lo tanto mayor capacidad para el procesamiento de la información.

5.3.1 Tiempo de entrenamiento – Técnica Árbol de Decisión

5.3.1.1 Tiempos de procesamiento Modelo 1

De acuerdo a los resultados del entrenamiento que se muestran en la siguiente figura, se tiene que el tiempo total de entrenamiento del modelo fue de tan solo 5.0103 segundos. Este tiempo es muy bajo considerando la gran cantidad de registros de ejemplo que se analizaron en total 28091. La velocidad de entrenamiento del modelo fue de 400.000 observaciones por segundo, o sea que se ejecutaron en total 2.004.120 observaciones.

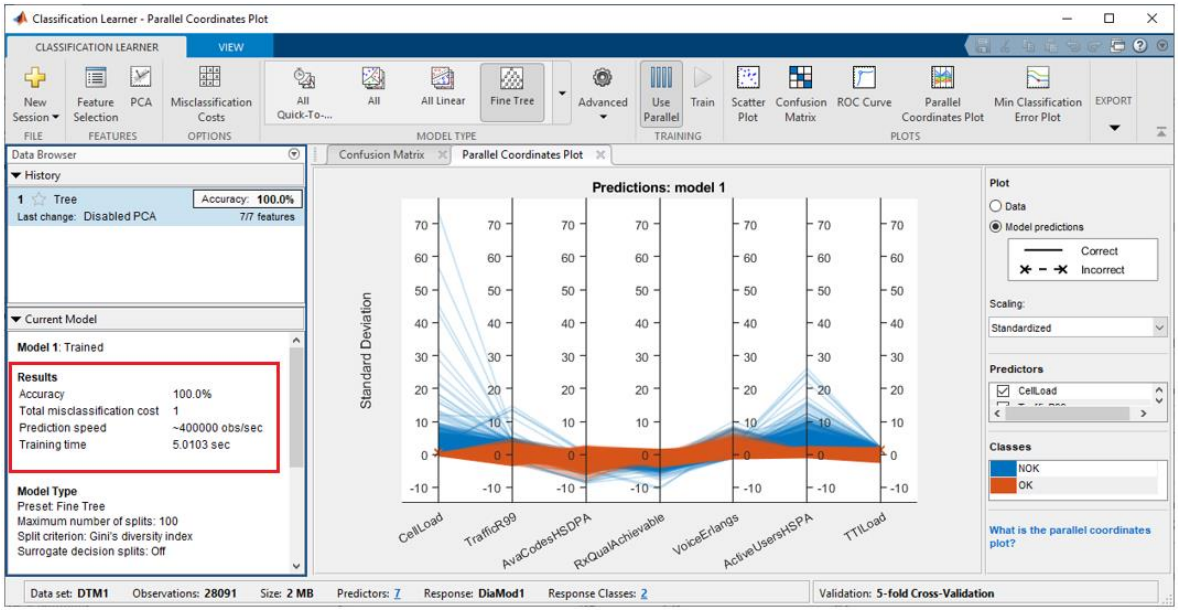


Figura No. 5-3. Tiempo de entrenamiento Modelo 1- Técnica Árbol de Decisión. Fuente propia.

5.3.1.2 Tiempos de procesamiento Modelo 2

De igual manera en la siguiente figura se muestran los resultados para el modelo 2, en este segundo modelo los registros de ejemplo se reducen a 2.120, y aunque son mucho menos registros, se emplean mayor tiempo que el modelo 1 con un total de 28.633 segundos. Esto puede ser explicado por el hecho de que en la variable de salida ya no son solo dos categorías como en el modelo 1 sino que se incrementan a tres categorías. De este análisis se puede concluir que, se tiene un mayor impacto en el tiempo de procesamiento el hecho de tener más clases por clasificar que el número de registros. En relación a la velocidad de entrenamiento, se puede ver que en este modelo se reduce a 62.000 observaciones por segundo para tener un total de 1.775.246 observaciones en todo el proceso de entrenamiento.

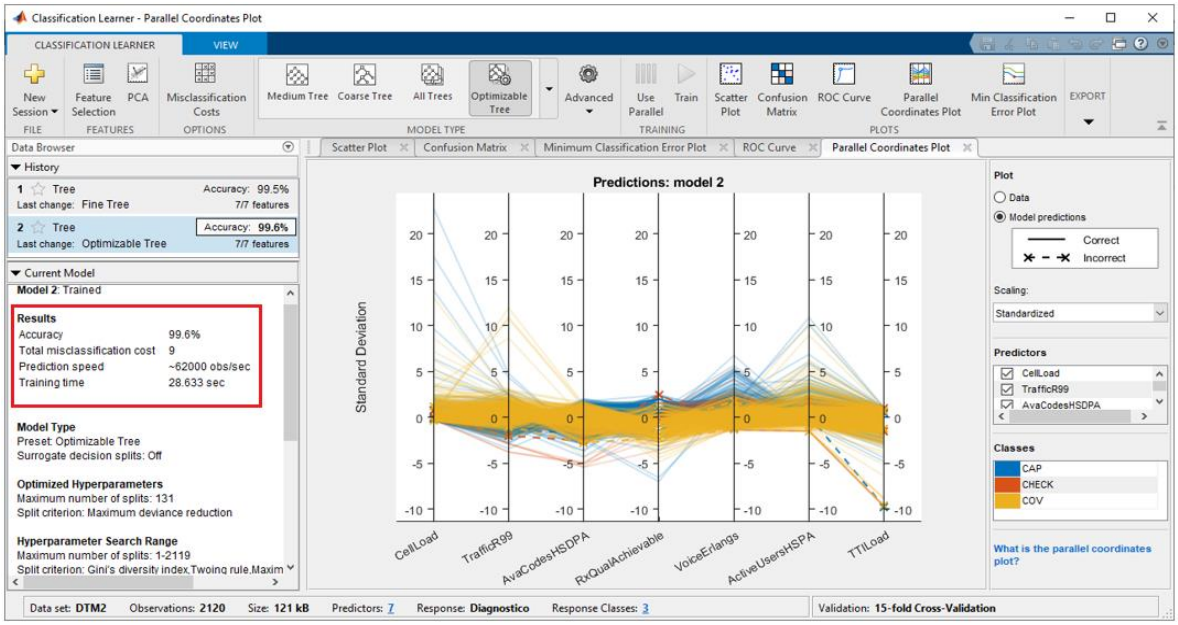


Figura No. 5-4. Tiempo de entrenamiento Modelo 2- Técnica Árbol de Decisión. Fuente propia.

5.3.2 Tiempo de entrenamiento – Técnica Red Neuronal

5.3.2.1 Tiempos de procesamiento Modelo 1

Aunque ninguno de los modelos derivados de la técnica de red neuronal fue seleccionado para el esquema final, si se considera importante validar los tiempos que consumen en los procesos de entrenamiento. En la siguiente figura se presentan los resultados del modelo 1, en donde se puede observar que a la red neuronal le tomo solo 2 segundos en realizar el proceso de entrenamiento. Durante este tiempo se ejecutaron 147 iteraciones, es decir que se ejecutaron 63 iteraciones por segundo aproximadamente.

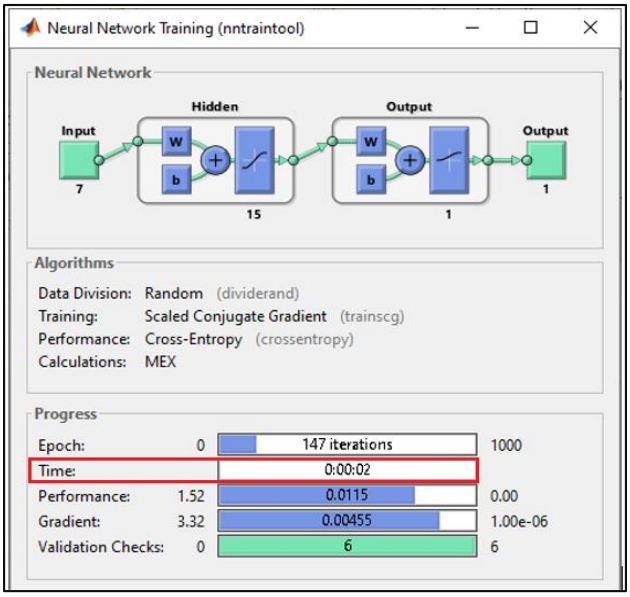


Figura No. 5-5. Tiempo de entrenamiento Modelo 1 – Técnica Red Neuronal. Fuente propia.

5.3.2.2 Tiempos de procesamiento Modelo 2

Para el modelo 2, se observa en los resultados de la siguiente figura que el tiempo de entrenamiento es muy pequeño y ni siquiera se marca 1 segundo. En cuanto al número de iteraciones, este modelo ejecuta más que el modelo 1 pero invierte un menor tiempo, esto se podría explicar por el hecho de que tiene menos registros que analizar.

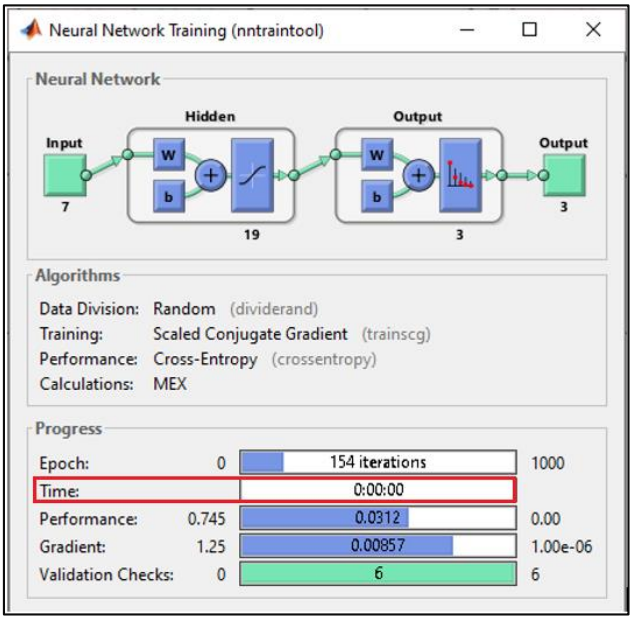


Figura No. 5-6. Tiempo de entrenamiento Modelo 2 – Técnica Red Neuronal. Fuente propia.

Al final lo que se observa en todos los modelos de las dos técnicas, es que el tiempo que se invierte en el proceso de entrenamiento es muy poco y no se requieren computadoras con características especiales de computo.

6. IMPLEMENTANCIÓN MODELO FINAL DE MACHINE LEARNING

Una vez terminado todo el proceso de generación y evaluación de los modelos, se procede a continuación con la implementación de los modelos seleccionados y que mejor desempeño tuvieron para realizar el proceso de diagnóstico de las celdas UMTS. El propósito de este objetivo es el de montar los algoritmos de los modelos en alguna de las aplicaciones disponibles, de tal manera que se puedan analizar nuevos datos de celdas. Para lograr este objetivo se deben exportar los algoritmos de estos modelos a un código de un lenguaje de programación que pueda aplicar todas las reglas de clasificación que se definieron y optimizaron para cada modelo.

Se toma como opción para la implementación de los algoritmos el uso de la aplicación "Guide" de Matlab, la cual funciona principalmente mediante interfaces graficas de usuario. La siguiente figura muestra el inicio de esta APP.

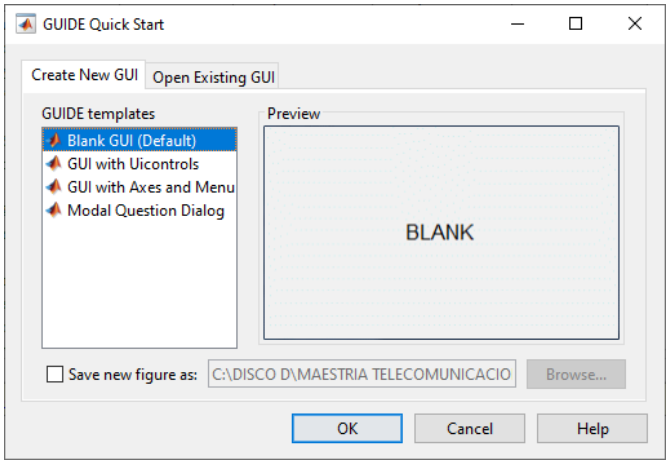


Figura No. 6-1. Aplicación Guide Matlab. Fuente propia.

Mediante esta aplicación es posible diseñar una interfaz de usuario que permita importar la información de las celdas UMTS que se desean diagnosticar, y posterior dentro de su código implementar las reglas de clasificación para finalmente entregar un reporte de las celdas con su correspondiente diagnóstico. Dado a que son dos modelos de clasificación los que se han desarrollado, se realiza el diseño de dos interfaces, una para cada modelo, pero conectadas entre ellas para el paso de información.

6.1 IMPLEMENTACIÓN ALGORITMO MODELO 1

La siguiente imagen muestra el diseño de la interfaz, en el que se incluyen una tabla para mostrar la información de las celdas, una imagen para mostrar los resultados consolidados y dos botones para ejecutar tanto el proceso de clasificación como el enlace al siguiente modelo.

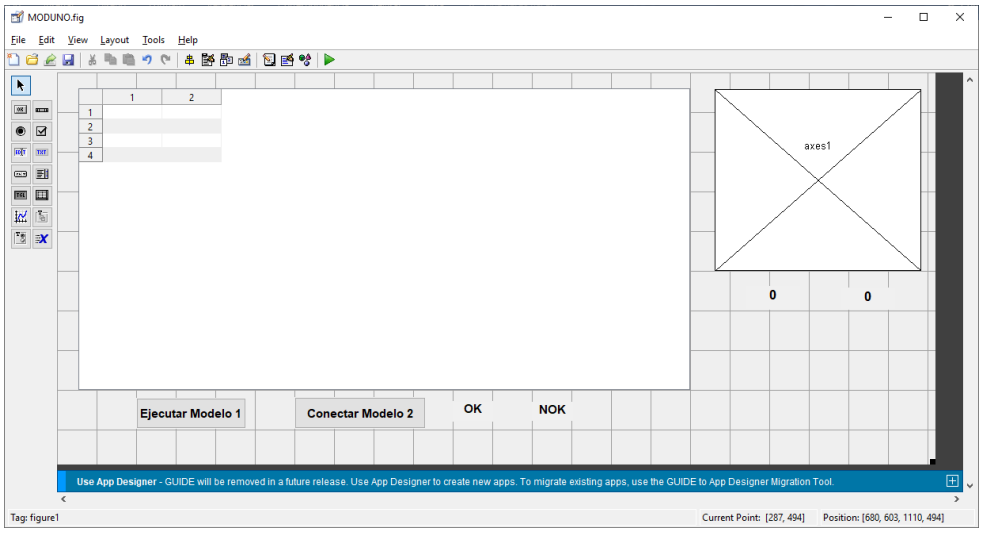


Figura No. 6-2. Diseño interfaz de usuario modelo 1. Fuente propia.

La siguiente imagen muestra el diseño final de la GUI para el modelo 1. Este modelo se encarga de clasificar las celdas en NOK y OK categorizando las celdas UMTS que tiene y no tienen problemas respectivamente. En la parte superior izquierda enmarcado en color rojo se puede ver que se ha agregado un menú con opciones tanto para importar la información que se va a procesar como para exportar la información ya procesada, en la parte central se ha definido una tabla en la cual se muestran toda la información de las celdas, al lado derecho se ha destinado un espacio para mostrar los resultados finales del proceso de clasificación mediante un diagrama de barras, y finalmente en la parte inferior la interfaz contiene dos botones con los cuales se pueden disparar los procesos de clasificación con el botón "Ejecutar modelo 1" al lado izquierdo y el segundo botón etiquetado como "Conectar Modelo 2" para el paso de los registros que se clasificaron como "NOK' al siguiente modelo.

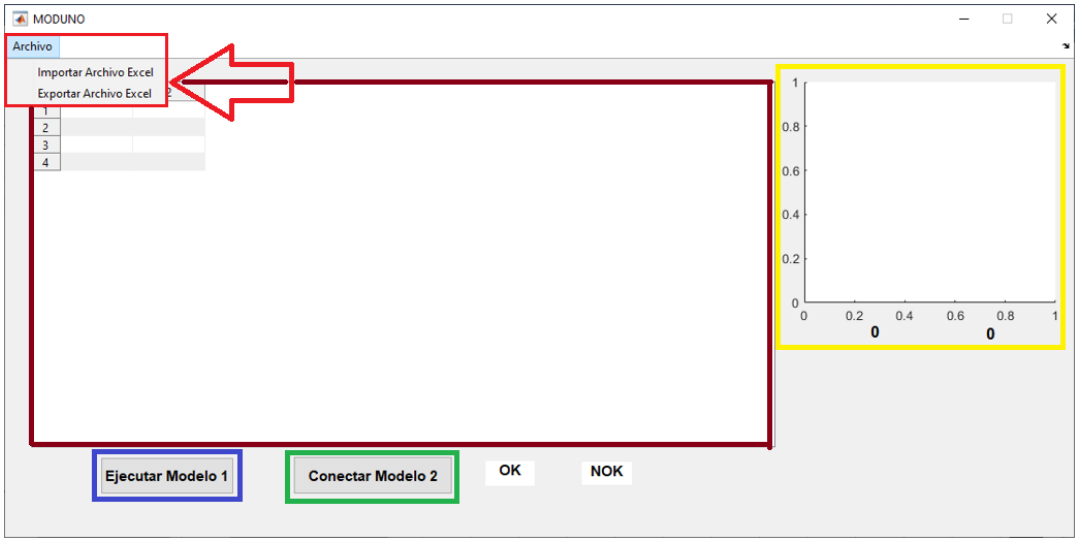
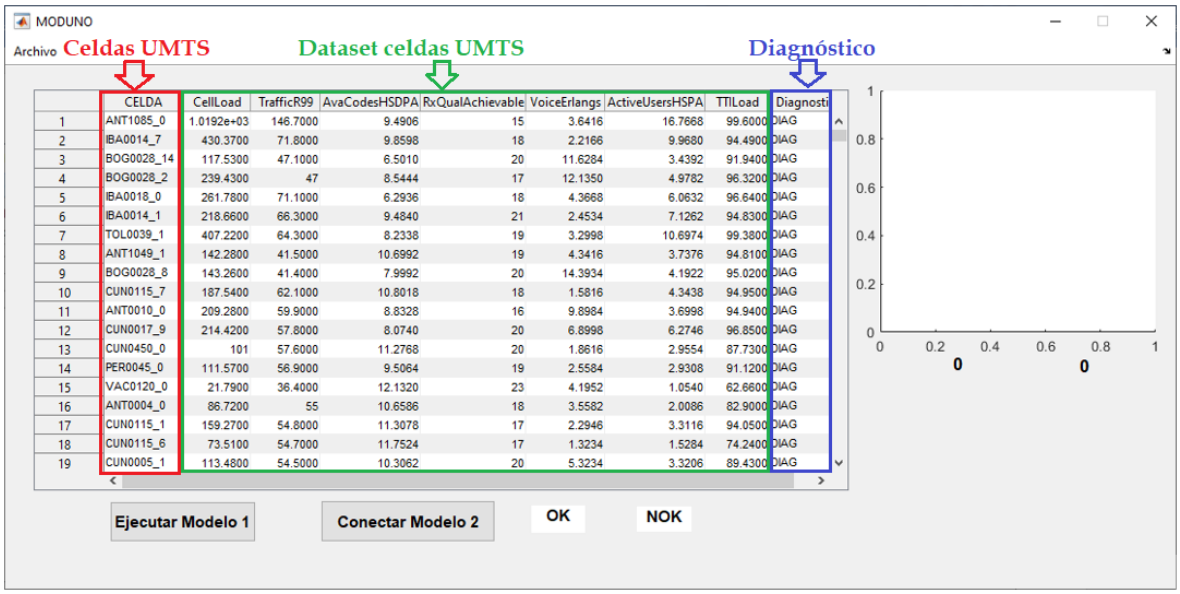


Figura No. 6-3. GUI – Modelo 1. Fuente propia.

La siguiente imagen muestra los datos importados para cada una de las celdas UMTS que se desean diagnosticar. En la parte izquierda enmarcada en color rojo se tiene la columna que indica el nombre de las celdas UMTS, en la parte central enmarcadas en color verde, se tienen siete columnas que corresponden a la información de las siete variables de entrada para cada celda y que serán utilizadas por el código del algoritmo para realizar el proceso de clasificación, finalmente en la parte derecha enmarcada en color azul se tiene la columna donde se entregará el resultado del diagnóstico.



	CELDA	CellLoad	TrafficR99	AvaCodesHSDPA	RxQualAchievable	VoiceErlangs	ActiveUsersHSPA	TTILoad	Diagnosti
1	ANT1085_0	1.0192e+03	146.7000	9.4906	15	3.6416	16.7668	99.6000	DIAG
2	IBA0014_7	430.3700	71.8000	9.8598	18	2.2166	9.9680	94.4900	DIAG
3	BOG0028_14	117.5300	47.1000	6.5010	20	11.6284	3.4392	91.9400	DIAG
4	BOG0028_2	239.4300	47	8.5444	17	12.1350	4.9782	96.3200	DIAG
5	IBA0018_0	261.7800	71.1000	6.2936	18	4.3668	6.0632	96.6400	DIAG
6	IBA0014_1	218.6600	66.3000	9.4840	21	2.4534	7.1262	94.8300	DIAG
7	TOL0039_1	407.2200	64.3000	8.2338	19	3.2998	10.6974	99.3800	DIAG
8	ANT1049_1	142.2800	41.5000	10.6992	19	4.3416	3.7376	94.8100	DIAG
9	BOG0028_8	143.2600	41.4000	7.9992	20	14.3934	4.1922	95.0200	DIAG
10	CUN0115_7	187.5400	62.1000	10.8018	18	1.5816	4.3438	94.9500	DIAG
11	ANT0010_0	209.2800	59.9000	8.8328	16	9.8984	3.6998	94.9400	DIAG
12	CUN0017_9	214.4200	57.8000	8.0740	20	6.8998	6.2746	96.8500	DIAG
13	CUN0450_0	101	57.6000	11.2768	20	1.8616	2.9554	87.7300	DIAG
14	PER0045_0	111.5700	56.9000	9.5064	19	2.5584	2.9308	91.1200	DIAG
15	VAC0120_0	21.7900	36.4000	12.1320	23	4.1952	1.0540	62.6600	DIAG
16	ANT0004_0	86.7200	55	10.6586	18	3.5582	2.0086	82.9000	DIAG
17	CUN0115_1	159.2700	54.8000	11.3078	17	2.2946	3.3116	94.0500	DIAG
18	CUN0115_6	73.5100	54.7000	11.7524	17	1.3234	1.5284	74.2400	DIAG
19	CUN0005_1	113.4800	54.5000	10.3062	20	5.3234	3.3206	89.4300	DIAG

Figura No. 6-4. Tabla de información celdas UMTS. Fuente propia.

Después de cargar el archivo con toda la información de las celdas se puede procesar la información mediante el botón con la etiqueta “Ejecutar Modelo 1” que se encuentra en la parte inferior izquierda de la GUI. Como se vio en el anterior capitulo donde se entrenaron y desarrollaron los modelos, cada modelo de clasificación entregó un árbol de decisión con unos umbrales de decisión. Para el primer modelo solo se evalúan la variable CellLoad con un valor de umbral de 80 para categorizar las celdas UMTS en OK y NOK. La siguiente imagen muestra el código implementado para el árbol de decisión del Modelo 1. Consiste en una sola sentencia de if – else que evalúa el umbral de 80. Las muestras que son menores a 80 se clasifican como OK y las mayores o iguales a 80 se clasifican como NOK.

```
for i=1:NF
    if C(i,1)<80
        T(i,9) = cellstr("OK");
        j(1,1)=j(1,1)+1;
    else
        T(i,9) = cellstr("NOK");
        j(1,2)=j(1,2)+1;
        handles.m = handles.m + 1;
    end
end
```

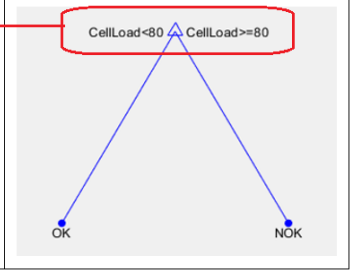


Figura No. 6-5. Asociación código y árbol de decisión – Modelo 1. Fuente propia.

La siguiente imagen muestra los resultados después de ejecutar el algoritmo del modelo 1. En la gráfica de barras de la parte derecha de la GUI se muestran los resultados consolidados del total de celdas en cada categoría. De igual manera en la última columna de la tabla se actualiza la categoría en la que se clasifica cada celda UMTS. Un total de 2120 celdas fueron diagnosticadas como OK y 25971 como NOK. Todos los códigos desarrollados para cada uno de los modelos se encuentran en los anexos.

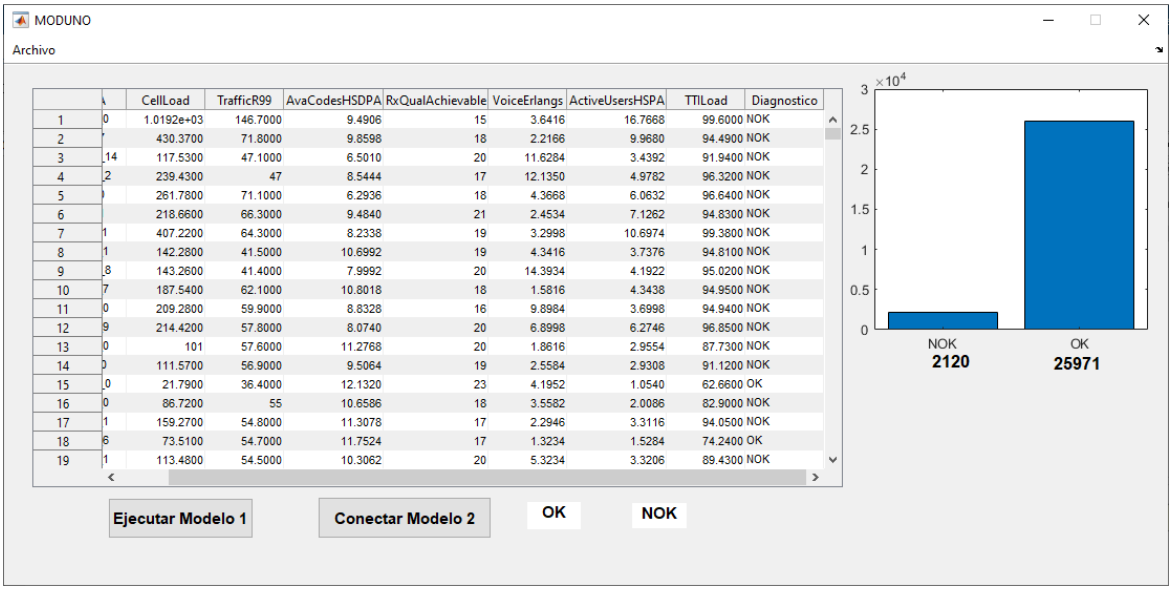
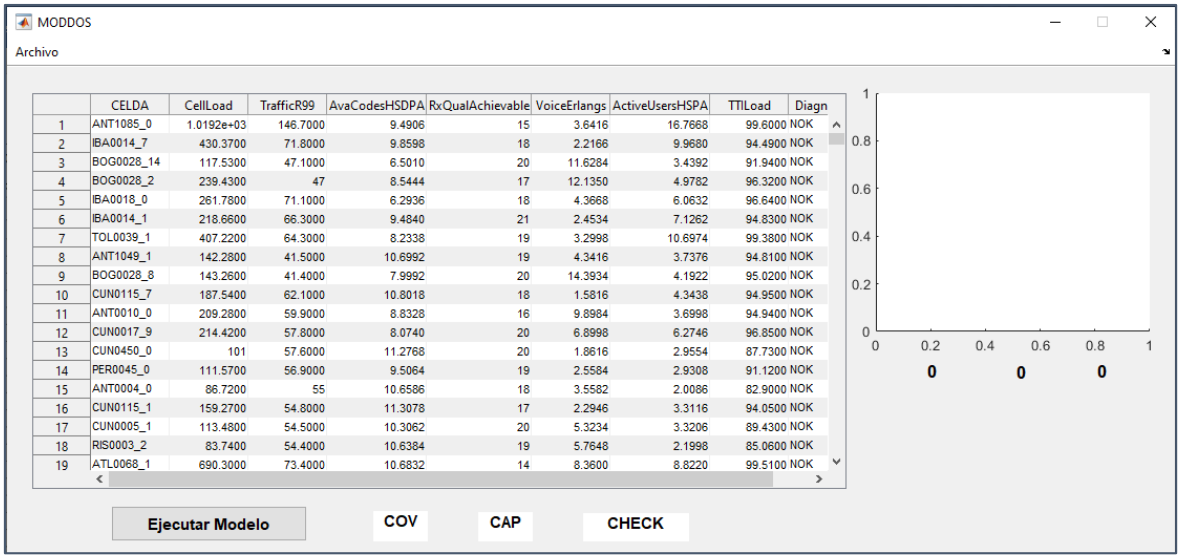


Figura No. 6-6. Resultados clasificación Modelo 1. Fuente propia.

Pero este es el primer paso del diagnóstico de las celdas, para pasar al siguiente paso se ha dispuesto en la GUI de un botón etiquetado como “Conectar Modelo 2” para poderse enlazar con el modelo 2. Es importe indicar que la información que se transfiere desde el modelo 1 al modelo 2 solo corresponde a las celdas que fueron clasificadas como NOK.

6.2 IMPLEMENTACIÓN ALGORITMO MODELO 2

La siguiente imagen muestra la interfaz diseñada para el segundo modelo, en realidad es muy similar a la del primer modelo, solo que en esta ocasión se diseña para presentar tres posibles categorías que son: COV, CAP y CHECK. Por otro lado, en esta interfaz solo se dispone de un boto etiquetado como “Ejecutar Modelo” y con el cual se realiza la ejecución del código del algoritmo asociado al árbol de decisión del modelo 2. En la parte superior de la GUI se dispone también de un menú con las opciones para importar información o exportar los resultados tal como aparecía en la primera interfaz de usuario.



	CELDA	CellLoad	TrafficR99	AvaCodesHSDPA	RxQualAchievable	VoiceErlangs	ActiveUsersHSPA	TTILoad	Diagn
1	ANT1085_0	1.0192e+03	146.7000	9.4906	15	3.6416	16.7668	99.6000	NOK
2	IBA0014_7	430.3700	71.8000	9.8598	18	2.2166	9.9680	94.4900	NOK
3	BOG0028_14	117.5300	47.1000	6.5010	20	11.6284	3.4392	91.9400	NOK
4	BOG0028_2	239.4300	47	8.5444	17	12.1350	4.9782	96.3200	NOK
5	IBA0018_0	261.7800	71.1000	6.2936	18	4.3668	6.0632	96.6400	NOK
6	IBA0014_1	218.6600	66.3000	9.4840	21	2.4534	7.1262	94.8300	NOK
7	TOL0039_1	407.2200	64.3000	8.2338	19	3.2998	10.6974	99.3800	NOK
8	ANT1049_1	142.2800	41.5000	10.6992	19	4.3416	3.7376	94.8100	NOK
9	BOG0028_8	143.2600	41.4000	7.9992	20	14.3934	4.1922	95.0200	NOK
10	CUN0115_7	187.5400	62.1000	10.8018	18	1.5816	4.3438	94.9500	NOK
11	ANT0010_0	209.2800	59.9000	8.8328	16	9.8984	3.6998	94.9400	NOK
12	CUN0017_9	214.4200	57.8000	8.0740	20	6.8998	6.2746	96.8500	NOK
13	CUN0450_0	101	57.6000	11.2768	20	1.8616	2.9554	87.7300	NOK
14	PER0045_0	111.5700	56.9000	9.5064	19	2.5584	2.9308	91.1200	NOK
15	ANT0004_0	86.7200	55	10.6586	18	3.5582	2.0086	82.9000	NOK
16	CUN0115_1	159.2700	54.8000	11.3078	17	2.2946	3.3116	94.0500	NOK
17	CUN0005_1	113.4800	54.5000	10.3062	20	5.3234	3.3206	89.4300	NOK
18	RIS0003_2	83.7400	54.4000	10.6384	19	5.7648	2.1998	85.0600	NOK
19	ATL0068_1	690.3000	73.4000	10.6832	14	8.3600	8.8220	99.5100	NOK

Figura No. 6-7. GUI – Clasificación Modelo 2. Fuente propia.

Para este segundo modelo se tiene un árbol de decisión mucho más amplio, lo que implica que el código equivalente con el cual se implementa el algoritmo del modelo es mucho más amplio. Esta técnica de árbol de decisión está compuesta por varios nodos y ramificaciones. Cada nodo del árbol de decisión es equivalente en el código a una sentencia de if-else, pero como existen varios nodos, lo que se tiene en el código es una serie de if anidados en ramificaciones como en el árbol. Cada sentencia de if se evalúa con el umbral definido en el árbol durante el proceso de entrenamiento. En las terminaciones de las ramificaciones se encuentran las tres posibles categorías. En la siguiente imagen se muestra la equivalencia que existe entre el árbol de decisión y el código desarrollado para este segundo modelo. Este código al igual que todos los códigos se encuentran de manera completa en los anexos del documento.

Para facilitar las equivalencias entre los nodos del árbol de decisión y las sentencias if del código, se han creado líneas de conexión en una de las ramificaciones del árbol de decisión iniciando desde el nodo raíz.

Cada umbral definido en los nodos de decisión para la variable dada se transfiere al código para ser evaluada mediante las sentencias if.

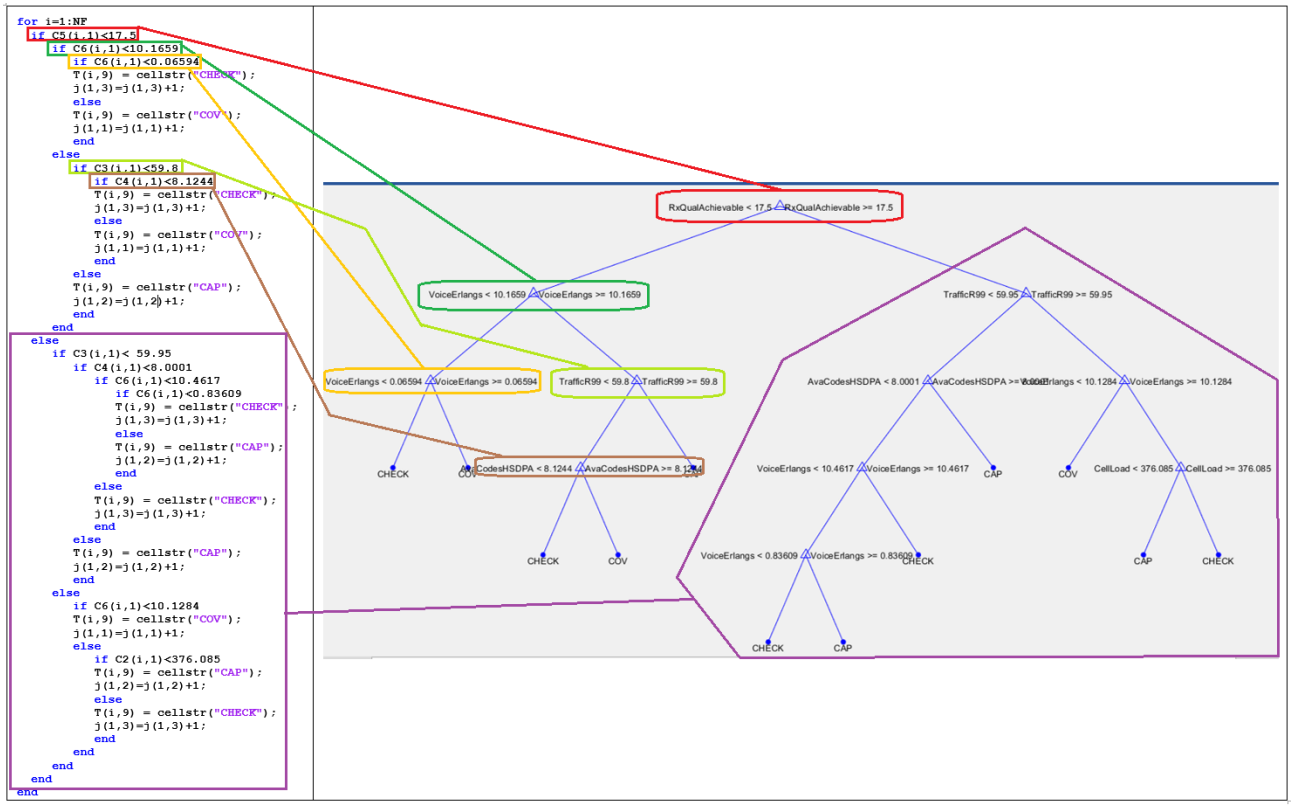


Figura No. 6-8. Asociación código y árbol de decisión – Modelo 2. Fuente propia.

La siguiente imagen muestra los resultados después de ejecutado el segundo modelo, y al igual que en el primer modelo, en la gráfica ubicada en la parte derecha de la GUI, se presentan en diagrama de barras los resultados acumulados de cada una de las categorías. En la última columna de la tabla se actualiza la categoría de cada una de las celdas UMTS según el diagnóstico realizado por el algoritmo. En la parte superior izquierda de la GUI, dentro del menú "Archivo", se da la opción para exportar en un archivo de Excel todos los resultados finales del modelo.

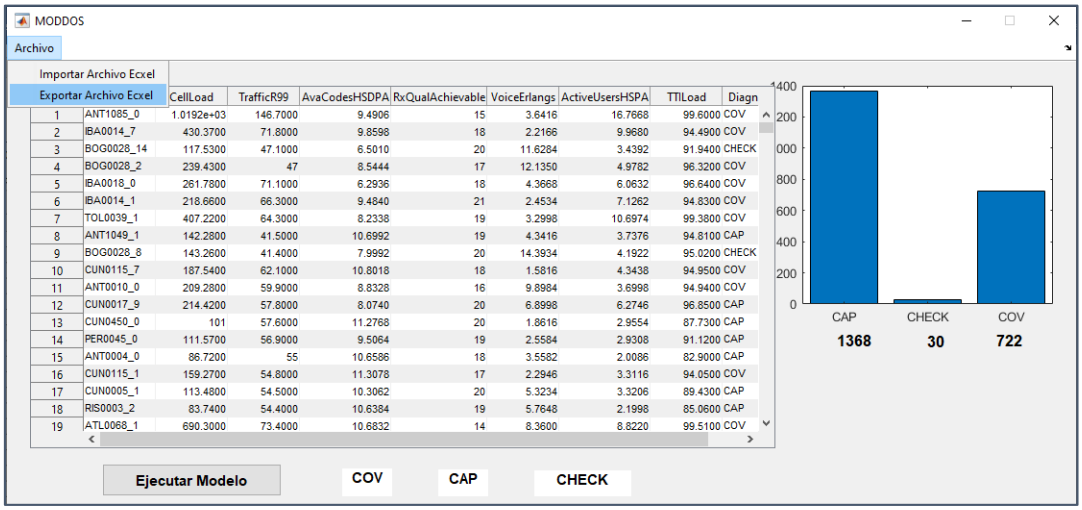


Figura No. 6-9. Resultados clasificación Modelo 2.

Para este ejercicio en particular, se pudo diagnosticar un total de 28.091 celdas UMTS, de las cuales 25.971 celdas no presentan ningún tipo de problema. Es decir que, aproximadamente un 92.45% de las celdas de la red se encuentran operando bien y no necesitan un proceso de optimización, estas celdas fueron identificadas en el modelo 1. Para las 2120 celdas que presentan algún problema, se utiliza el modelo 2, el cual las clasifica con mayor detalle de la siguiente manera, 1368 celdas con problemas de capacidad (CAP), 722 celdas con problemas de cobertura (COV) y 30 celdas con problemas de hardware, software o configuración. De esta manera se concluye todo el proceso de diagnóstico de las celdas UMTS, todo esto haciendo uso de los dos algoritmos entregados por la técnica de árbol de decisión.

De esta manera se concluye con todos los objetivos propuestos en este proyecto de investigación. Al final se logra crear una aplicación capaz de realizar el diagnostico de las celdas UMTS con solo un data set de variables que se pueden obtener de los gestores de desempeño. Ahora la aplicación que se obtuvo como resultado de los modelos podrá ser utilizada por los ingenieros de optimización en sus tareas para el mejoramiento del desempeño de la red. El beneficio es la agilidad que se logra en el diagnóstico de las celdas, esto brinda mayor dinamismo en la ejecución de los planes de acción que se deben ejecutar sobre la red.

7. CONCLUSIONES

- Durante las pruebas realizadas se pudo evidenciar que un gran problema de clasificación que involucra varias clases, se puede dividir en varios modelos, para ello se deben unificar las clases similares en otras que las contengan.

- Cuando se crean varios modelos, la velocidad de entrenamiento se disminuye al tener menos clases que clasificar comparado con un solo modelo que tenga que clasificar varias clases. Pues como se pudo observar en los tiempos de entrenamiento de los modelos de la técnica árbol de decisión, impacta mucho más el número de categorías o clases a clasificar que el número de registros de ejemplo.

- Cuando una clase tiene muy pocos elementos, cualquier error ya sea de tipo 1 o tipo 2 pueden impactar significativamente la sensibilidad y precisión de la clase. Es por esto que recomienda obtener data sets con bastantes muestras de ejemplos para cada una de las categorías o clases que se desean clasificar.

- Fue importante abordar el problema de clasificación con dos técnicas de Machine Learning, y aunque en primera instancia se pensaría que la técnica de redes neuronales podría tener mejor desempeño, se pudo comprobar mediante las métricas de desempeño que la técnica de árbol de decisión es una excelente alternativa para los problemas de clasificación.

- Respecto del ejercicio planteado, se puede concluir que la clasificación mediante los diferentes diagnósticos para celdas UMTS funcionó en gran medida y podría ser considerado como una alternativa para ayudar en los procesos de diagnósticos de las celdas. Con estos modelos se agilizaría el trabajo de validación, en especial en el tiempo de procesamiento.

8. TRABJOS FUTUROS

En la actualidad se encuentran en funcionamiento muchas redes celulares de segunda, tercera, cuarta y próximamente las de quinta generación. Cada una de ellas con sus características propias en la parte de acceso en la interfaz aire, o con sus capacidades en el Backhaul hacia el Core. Y para todos estos elementos e interfaces en las redes celulares es indispensable el desarrollo de las actividades de optimización que busquen garantizar al máximo su apropiado funcionamiento.

Como cada generación de red celular tiene sus propios indicadores que miden el desempeño de sus interfaces, se definen problemas propios en cada red que deben ser abordados y optimizados de acuerdo a la información disponible con la cual se realizan los diagnósticos de los elementos e interfaces de red. Pero sin importar el tipo de red que se desee optimizar, ya sea GSM, UMTS o LTE, siempre se tendrán a disposición los llamados contadores de desempeño estadísticos y sus correspondientes KPIs que son la información vital de entrada a cualquier proceso de optimización.

La clave está en saber seleccionar y analizar la información disponible en un problema, que permita un adecuado diagnóstico asociado a cada uno de los elementos de red o de sus interfaces, y en este sentido cobra especial importancia el trabajo realizado por los expertos de redes celulares, quienes basados en su expertis saben que información se debe analizar y como interpretarla. En muchos casos la información útil para abordar los problemas de cada una de las redes celulares puede ser propia de cada tecnología o similar en otros casos como por ejemplo la cantidad de tráfico o la potencia consumida en una celda. Pero siempre podrá ser posible armar un data set valido para el entrenamiento de un modelo de Machine Learning.

Al final el mensaje que se quiere entregar, es que sin importar el tipo de red siempre será viable el uso de técnicas de Machine Learning en la resolución de problemas del mundo de las redes celulares en cualquiera de sus generaciones. En este proyecto solo se abordó un problema de optimización en la parte de acceso de una red 3G, pero también se podrían plantear los siguientes proyectos de investigación como complemento o continuación a este proyecto.

- En redes 3G para el diagnóstico de la interfaz Iub donde se consideren factores como el consumo del recurso de transmisión, jitter, delay, latencia, retransmisiones y paquetes errados entre otros. Al final se podrían predecir posibles bloqueos o fallas antes de que ocurran.
- En redes 4G problemas similares en la parte de acceso, donde factores claves podrían ser el porcentaje de uso de PRB (Physical Radio Beares), cantidad de usuarios, trafico cursado, potencia consumida entre otros. El objetivo final sería similar al de este proyecto, diagnosticar problemas de cobertura, capacidad o simplemente fallas de hardware o software.
- En redes 4G para problemas de diagnóstico de la interfaz S1, similar a la primera propuesta de 3G en la interfaz Iub.

9. REFERENCIAS

[1] Comisión de Regulación de Comunicaciones. (23 de diciembre de 2019). *Modernización-redes-móviles CRC* [Archivo pdf]. https://www.crcom.gov.co/uploads/images/files/Documento-Modernizacion-redes-moviles-1.pdf

[2] Unión Internacional de Telecomunicaciones. Proyecto IMT-2000. (2002). *What is IMT-2000* [Archivo pdf]. http://www.itu.int/osg/imt-project/docs/What_is_IMT2000-2.pdf.

[3] 3GPP Technichal Specification Group. 25.401. (abril 1999). *UTRAN Overall Description (Release 99)*. https://www.3gpp.org/ftp/Specs/archive/25_series/25.401/.

[4] Villareal L., (2015). *Procedimientos de Optimización en Redes de Acceso 3G WCDMA/HSPA y su efectividad en Casos de Estudio en Nicaragua*. [Tesis de pregrado]. Universidad Nacional de Ingeniería. Nicaragua-Managua.

[5] Gonzáles J., (2015). *Monitorización y Gestión de la Capacidad de la Interfaz Radio de una Red UMTS*. [Tesis de pregrado]. Universidad de Sevilla. España-Sevilla.

[6] 3GPP Technichal Specification Group. 25.214. (junio de 2003). *Physical layer procedures (FDD) (Release 5)*. https://www.3gpp.org/ftp/Specs/archive/25_series/25.214/.

[7] Haider, F., Hepsaydir, E., y Binucci, N. (2011). *Performance Analysis of a Live Mobile Broadband-HSDPA Network*. [Archivo jpg]. https://www.researchgate.net/figure/Cell-and-HS-user-throughput-with-no-backhaul-limitations_fig1_221640131.

[8] Unión Internacional de Telecomunicaciones. CCITT E.800. (noviembre de 1988). *Vocabulario de Calidad de Servicio y Seguridad de Funcionamiento.* [Archivo pdf]. https://www.itu.int/rec/T-REC-E.800/es.

[9] Constain D., (febrero de 2019). *Cobertura y capacidad en redes 2G, 3G y 4G*. [Archivo pdf]. https:r//epository.udistrital.edu.co. Colombia – Bogotá.

[10] Lozano I., (2011). *Optimización del Dimensionado de una red móvil (3G).* [Tesis de grado]. Universidad Politécnica de Cataluña. España-Barcelona.

[11] Rouhiainen L., (2018). Inteligencia Artificial. *Editorial Alienta*. España-Barcelona.

[12] LPSIngenieria, Noticias sobre tecnología y telecomunicaciones. (18 de enero de 2018). *El uso de la Inteligencia Artificial para optimizar las redes móviles*. https://lpsingenieria.com/uso-inteligencia-artificial-redes-moviles.

[13] The MathWorks, Inc. (1994-2021). MathWorks: *Machine Learning_ Tres cosas que es necesario saber*. Recuperado de https://la.mathworks.com/discovery/deep-learning.html

[14] Pimentel G., (2020). *La Inteligencia Artificial y su influencia en la eficiencia del Comercio Internacional*. [Tesis de pregrado]. Universidad San Ignacio de Loyola. Perú-Lima.

[15] The MathWorks, Inc. (1994-2021). MathWorks: *Aprendizaje automático en MATLAB*. Recuperado de https://la.mathworks.com/help/stats/machine-learning-in-matlab.html?searchHighlight=MACHINE%20LEARNING&s_tid=srchtitle#bvni0go.

[16] Maganto, Á., (junio de 2017). *Estudio de Técnicas Supervisadas de Reducción de Dimensionalidad para Problemas de Clasificación*. [Tesis de pregrado]. Universidad Carlos Tercero de Madrid. España-Madrid.

[17] Reyes, C., Morales, E., Escalante, H., y Torres, A. (2018). *Aprendizaje e Inteligencia Computacional*. Academia Mexicana de Computación.

[18] Cisneros, S., Medina, L., (agosto de 2017). *Reconocimiento Automático de Cartas de Baraja Ubicadas sobre una mesa*. [Tesis de pregrado]. Universidad Politécnica Salesiana. Ecuador-Quito.

[19] Caparrini, F., (14 de diciembre 2019). *Redes Neuronales: una visión superficial*. http://www.cs.us.es/~fsancho/?e=72.

[20] Artola, Á., (2019). *Clasificación de imágenes usando redes neuronales convolucionales en Python*. [Tesis de pregrado]. Universidad de Sevilla. España-Sevilla.