UNIVERSIDAD NACIONAL SAN CRISTÓBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

“Estrés estudiantil universitario basado en redes neuronales profundas, en pandemia covid-19, 2021”

Tesis presentada por	**:**	Bach. Kelly Patricia De la Cruz Oriundo
Para optar el título profesional de	**:**	Ingeniero de Sistemas
Tipo de investigación	**:**	Aplicada
Área de Investigación	**:**	Ciencia de Datos
Asesor	**:**	Mg. Ing. Hubner Janampa Patilla

Ayacucho - 2022

DEDICATORIA

Al Señor Todopoderoso y a la virgen Auxiliadora, quienes me han cuidado, bendecido y acompañado en cada uno de los pasos que he dado en mi vida.

A Hugo y Digna, mis padres que, con su ejemplo de integridad y esmero han guiado mi camino y me han formado con valores que han sido pilares en mi vida, por su soporte que me ha ayudado a afrontar los momentos difíciles.

A mis hermanos que, con cada uno de sus logros profesionales, su ética laboral, su continua asesoría, fueron un ejemplo y guía para cumplir mis metas.

A las diferentes personas que, a lo largo de mi vida, me ayudaron y me impulsaron a ser mejor.

AGRADECIMIENTO

A la Universidad Nacional de San Cristóbal de Huamanga, por ayudarme a culminar mi formación universitaria, gracias por todo el apoyo en esta etapa y en esta investigación.

A la plana docente de la Escuela de Ingeniería de Sistemas de la Universidad Nacional de San Cristóbal de Huamanga, por todos los conocimientos impartidos, que me hicieron crecer día a día como profesional, de manera especial, al Mg. Ing. Hubner Jananpa Patilla asesor de este proyecto de investigación quien durante todo este proceso me ha guiado y orientado para culminar este trabajo.

A los alumnos de pregrado de diferentes universidades, en especial a los de la Universidad Nacional de San Cristóbal de Huamanga, que me sirvieron de inspiración, y que me permitieron recabar la información con cada encuesta realizada, por su valioso aporte para esta investigación.

RESUMEN

En marzo del 2020 en el Perú, el gobierno declaro una cuarentena para evitar el incremento de contagios de COVID-19, esto generó diversas medidas, una de estas fue la interrupción indefinida de clases presenciales a todos los niveles, por lo que se implementaron en las diferentes instituciones educativas las clases virtuales; esta situación ocasionó cambios en las rutinas de las personas, así como en los estudiantes universitarios que al adaptarse a la enseñanza virtual en un contexto de aislamiento social se vieron afectados y algunos podrían haber desarrollado estrés, que si no son detectados y tratados a tiempo, en un futuro se verían reflejados en diferentes trastornos crónicos.

El principal objetivo de esta investigación es utilizar redes neuronales profundas y con ayuda de estas, determinar el estrés estudiantil universitario, en pandemia covid-19, en el Perú mediante técnicas e instrumentos, usando la metodología ASUM-DM, métodos interpretativos, algoritmos de aprendizaje, arquitectura de redes neuronales y análisis exploratorio de datos, para así generar nueva información y automatizar este proceso.

Los datos de la investigación están basados en la encuesta realizada a diversos alumnos de pregrado, durante el último semestre del año 2021.

En la presente investigación se utilizará métodos interpretativos que con el modelado estadístico nos permitirá reconocer patrones de clasificación y de predicción, que ayudara a describir como la cuarentena y los cambios de habito en los estudiantes universitarios generan diversos niveles de estrés, esto según la información obtenida.

PALABRAS CLAVE

Estudiante universitario, Educación virtual, Estrés, COVID – 19, Redes Neuronales Profundas.

ABSTRACT

In March 2020 in Peru, the government declared a quarantine to prevent the increase in COVID-19 infections, this reinforced various measures, one of these was the indefinite interruption of face-to-face classes at all levels, for which they were implemented in the different educational institutions virtual classes; This situation caused changes in people's routines, as well as in university students who, adapting to virtual teaching in a context of social isolation, were affected and some could have developed stress, which if not detected and treated in time, in the future they would be reflected in different chronic disorders.

The main objective of this research is to use deep neural networks and with the help of these, determine university student stress, in the covid-19 pandemic, in Peru through techniques and instruments, using the ASUM-DM methodology, interpretive methods, learning algorithms, neural network architecture and exploratory data analysis, in order to generate new information and automate this process.

The research data is based on the survey carried out on various undergraduate students, during the last semester of the year 2021.

In the present investigation, interpretive methods will be used that, with statistical modeling, will allow us to recognize classification and prediction patterns, which will help to describe how quarantine and habit changes in university students generate different levels of stress, according to the information obtained.

KEYWORDS

University student, Virtual education, Stress, COVID – 19, Deep Neural Networks.

INDICE

PAGINA DE APROBACIÓN O CONFORMIDAD.....**¡Error! Marcador no definido.**
DEDICATORIA ii
AGRADECIMIENTO iii
RESUMEN.....iv
ABSTRACT v
INDICE vi
LISTA DE TABLAS ix
LISTA DE FIGURAS x
INTRODUCCIÓN.....13

CAPÍTULO I

PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1. DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA.....14
1.2. DEFINICIÓN DEL PROBLEMA DE INVESTIGACIÓN 16
1.3. OBJETIVO GENERAL.....16
1.4. OBJETIVOS ESPECÍFICOS.....17
1.5 JUSTIFICACIÓN.....17
1.6 DELIMITACIÓN 17

CAPÍTULO II

MARCO TEÓRICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN 18
2.2. MARCO TEÓRICO.....18
2.2.1. COVID - 19 18
2.2.2. ESTUDIANTES UNIVERSITARIOS.....19
2.2.3. ESTRÉS.....21
2.2.4. EDUCACION A DISTANCIA 23
2.2.5. INVENTARIO SISCO SV - 21 24
2.2.6. INTELIGENCIA ARTIFICIAL 28
2.2.7. DATA SCIENCE.....30
2.2.8. MACHINE LEARNING.....31
2.2.9. DATA MINING.....31
2.2.10.REDES NEURONALES ARTIFICIALES 33

2.2.11. DEEP LEARNING 36
2.2.12. FUNCIONES DE ACTIVACIÓN 37
2.2.12.1. Partially differentiable activation functions 37
2.2.12.2. Fully differentiable activation functions 39
2.2.13. ALGORITMO DE BACKPROPAGATION 44
2.2.14. ARQUITECTURA DE MULTIPLES CAPAS 48
2.2.15. THE LOSS FUNCTION 50
2.2.16. OPTIMIZADORES 52
2.2.17. PROCESOS DE ENTRENAMIENTO Y PROPIEDADES DEL APRENDIZAJE 53
2.2.17.1. Aprendizaje supervisado 54
2.2.17.2. Aprendizaje sin supervision 56
2.2.17.3. Aprendizaje reforzado 56
2.2.17.4. Offline learning 57
2.2.17.5. Online learning 57
2.2.18. CAPAS DE SALIDA SOFTMAX 57
2.2.19. GRADIENTE DESCENDIENTE 58
2.2.20. LEARNING RATE O RANGO DE APRENDIZAJE 60
2.2.21. MÉTODOS DE VALIDACIÓN CRUZADA 61
2.2.22. MATRIZ DE CONFUSIÓN 64
2.2.23. REGRESIÓN LOGÍSTICA 65
2.2.23.1. Regresión logística para datos binarios no estratificados 65
2.2.23.2. Regresión logística para datos binarios estratificados 67
2.2.24. GOOGLE COLABORATORY 69
2.2.25. PYTHON 71
2.2.26. TENSORFLOW 72
2.2.27. KERAS 72

CAPÍTULO III

MATERIAL Y MÉTODOS

3.1. TIPO DE INVESTIGACIÓN 73
3.2. DISEÑO DE INVESTIGACIÓN 73
3.3. HIPÓTESIS DE LA INVESTIGACIÓN 74
3.4. POBLACIÓN Y MUESTRA 74

3.5. DEFINICIÓN CONCEPTUAL DE LAS VARIABLES 75
3.6. DEFINICIÓN OPERACIONAL DE LAS VARIABLES 77
3.7. ASUM – DM 77
3.8. TÉCNICAS E INSTRUMENTOS 81

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1. RECOLECCIÓN DE DATOS 82
4.2. LIMPIEZA Y TRANSFORMACIÓN DE DATOS 86
4.2.1. Limpieza de datos 86
4.2.2. Feature engineering 87
4.2.3. Transformación de datos 88
4.3. VALIDACIÓN DE DATAFRAME 89
4.4. CALIBRACIONES 95
4.4.1. Calibración del ratio de aprendizaje o learning rate 95
4.4.2. Calibración del tamaño de batch 101
4.4.3. Calibración del optimizador 105
4.5. EVALUACION DEL MODELO DE REDES NEURONALES 110
4.5.1. Modelo con 5 capas 110
4.5.2. Modelo con 5 capas II 113
4.5.3. Modelo con 6 capas 116
4.5.4. Modelo con 5 capas III 119
4.6. ARQUITECTURA DE RED NEURONAL CON CLASIFICACION MULTICLASE 122
4.7. VALIDACION DEL MODELO DE RED NEURONAL 124

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES 130
5.2. RECOMENDACIONES 131
REFERENCIAS BIBLIOGRAFICAS 132
ANEXO A 136
ANEXO B 139
ANEXO C 141

LISTA DE TABLAS

Tabla 1 Porcentaje de hogares con internet y computadoras.....15
Tabla 2 Escala de valores.....25
Tabla 3 Listado de estresores.....26
Tabla 4 Listado de síntomas.....26
Tabla 5 Listado de estrategias.....27
Tabla 6 Elección de variables para determinar estrés estudiantil.....87
Tabla 7 Elección de variables entorno de covid -19.....88
Tabla 8 Niveles de estrés estudiantil. 88
Tabla 9 Tipos de respuestas y transformación por variable. 89
Tabla 10 Valores predicción y error de los modelos según iteración. 141

LISTA DE FIGURAS

Figura 1 Estructura clásica de un Sistema de conocimientos. 29
Figura 2 Proceso de descubrimiento del conocimiento 33
Figura 3 Diagrama de Neurona simple. 35
Figura 4 Grafica de función de paso. 37
Figura 5 Grafica de la función de paso bipolar. 38
Figura 6 Grafica de la función de rampa simétrica. 39
Figura 7 Grafica de función de activación logística. 40
Figura 8 Grafica de función de activación logística cuando el parámetro β cambia. 40
Figura 9 Grafica de función de activación tangente hiperbólica. 41
Figura 10 Grafica de la función de activación tangente hiperbólica cuando el parámetro β cambia. 42
Figura 11 Grafica de la función de activación gaussiana. 43
Figura 12 Grafica de la función de activación lineal. 44
Figura 13 Una red de tres capas de retro propagación. 46
Figura 14 Designación de las neuronas y pesos para la aplicación de la regla de retro propagación. 47
Figura 15 Algoritmo de backpropagation. 48
Figura 16 Ejemplo de red neuronal con arquitectura multicapas. 49
Figura 17 La superficie de error cuadrático para una neurona lineal. 59
Figura 18 Visualización de la superficie de error como un conjunto de contornos. 60
Figura 19 Método de validación cruzada de submuestreo aleatorio. 62
Figura 20 Método de validación cruzada de k-fold. 63
Figura 21 Método de validación cruzada uno fuera. 64
Figura 22 Matriz de confusión. 64
Figura 23 Ciclo de extracción de datos de ASUM – DM. 78
Figura 24 Metodología ASUM – DM de IBM. 79
Figura 25 Gráfico de universidades a las que pertenecen los encuestados. 82
Figura 26 Gráfico de ciclos a los que pertenecen los encuestados. 83
Figura 27 Gráfico de las semanas del semestre en que están los encuestados. 83
Figura 28 Gráfico de las respuestas sobre capacitaciones respecto a salud mental. 84
Figura 29 Gráfico de las respuestas sobre impacto de covid-19 en rendimiento académico. 84
Figura 30 Gráfico de las respuestas sobre si el impacto de covid-19 fue positivo o negativo. 85
Figura 31 Gráfico de las respuestas sobre si tuvo contagio de covid-19. 85
Figura 32 Gráfico de las respuestas sobre si pudo realizar sus labores de forma cotidiana. 86
Figura 33 Conexión a archivo de datos 90
Figura 34 Definición de features. 90
Figura 35 Código de inserción de variables dummy. 91
Figura 36 Features insertadas a matriz. 92
Figura 37 Definicion de Label 92
Figura 38 Representación de Labels a variables numéricas. 92
Figura 39 Representación en matriz de los labels. 93

Figura 40 Código inicial del modelo. 93
Figura 41 Llamada a la libreria KerasClassifier. 94
Figura 42 Ejecución de validaciones. 94
Figura 43 Resultado de la validación. 94
Figura 44 Resultado de la validación estándar. 95
Figura 45 Valores en matriz de la feature. 95
Figura 46 Valores categóricos en matriz. 96
Figura 47 Validación cruzada. 96
Figura 48 Código de la división de la data. 97
Figura 49 Calibración de rango de aprendizaje. 97
Figura 50 Validación de entrenamiento del modelo. 97
Figura 51 Valores de LR. 98
Figura 52 Código de calibración de LR. 98
Figura 53 Rango de aprendizaje de LR 99
Figura 54 Rango de error de LR. 99
Figura 55 Gráfico de la evolución del loss. 100
Figura 56 Valores de accuracy de LR. 100
Figura 57 Gráfico de la evolución del accuracy. 101
Figura 58 Código de separación de data. 101
Figura 59 Valores para el batch. 102
Figura 60 Código de calibración del batch. 102
Figura 61 Código de resultados en array. 103
Figura 62 Valores de loss del batch. 103
Figura 63 Gráfico de la evolución del loss del batch. 104
Figura 64 Valores de accuracy del batch. 104
Figura 65 Gráfico de la evolución del accuracy del batch. 105
Figura 66 Código de separación de data. 105
Figura 67 Importación de librerías de optimizadores. 106
Figura 68 Definición de valores de optimizadores. 106
Figura 69 Calibración de optimizadores. 107
Figura 70 Código para dataframe de resultados de optimizadores. 107
Figura 71 Valores de error de optimizador. 108
Figura 72 Gráfico de la evolución del loss del optimizador. 108
Figura 73 Valores de accuracy de optimizador. 109
Figura 74 Gráfico de la evolución del accuracy del optimizador. 109
Figura 75 *Código de modelo I.* 110
Figura 76 Descripción del modelo I. 110
Figura 77 Valores de entrenamiento del modelo I. 111
Figura 78 Gráfico de nivel de predicción del modelo 1. 112
Figura 79 Gráfico de nivel de error del modelo 1. 112
Figura 80 Código de modelo II. 113
Figura 81 Descripción del modelo II. 113
Figura 82 Valores de entrenamiento del modelo II. 114
Figura 83 Gráfico de nivel de predicción del modelo 2. 115
Figura 84 Gráfico de nivel de error del modelo 2. 115
Figura 85 Código de modelo III. 116
Figura 86 Descripción del modelo III. 116

Figura 87 Valores de entrenamiento del modelo III. 117
Figura 88 Gráfico de nivel de predicción del modelo 3. 118
Figura 89 Gráfico de nivel de error del modelo 3. 118
Figura 90 Código de modelo IV. 119
Figura 91 Descripción del modelo IV. 119
Figura 92 Valores de entrenamiento del modelo IV. 120
Figura 93 Gráfico de nivel de predicción del modelo 4. 121
Figura 94 Gráfico de nivel de error del modelo 4. 121
Figura 95 Arquitectura del modelo final de redes neuronales profundas. 123
Figura 96 Código de separación de data. 124
Figura 97 Código del modelo final. 124
Figura 98 Código de optimizador calibrado. 124
Figura 99 Iteraciones de entrenamiento del modelo final. 125
Figura 100 Array de predicción. 126
Figura 101 Array de valor de predicción. 126
Figura 102 Valores de Y reales. 127
Figura 103 Valores de Y de prueba. 127
Figura 104 Valores de matriz de confusión. 128
Figura 105 Página Web donde esta implementado el modelo de la red neuronal profunda. 129

INTRODUCCIÓN

La Organización Mundial de la Salud, el 11 de marzo del 2020, clasifico el brote de la COVID - 19 como una pandemia al extenderse en más de cien países. En este contexto el Perú desde el 16 de marzo del 2020 se declaró el estado de emergencia nacional, obligando a la población a un aislamiento social. Al incrementarse los casos de COVID - 19 en el Perú, hizo que la cuarentena inicialmente de quince días, se fuera prorrogando, esto ocasionó que se declarara la suspensión indefinida de clases presenciales.

El Ministerio de Educación (MINEDU) como consecuencia de la cuarentena, saco diversas disposiciones para la prevención y monitoreo de las clases en las universidades, donde se tiene como objetivo evitar la expansión del coronavirus; además de esto el ministerio desea que las clases de educación superior sigan de manera remota en diversas materias, a diferencia de las clases escolares, ya que el 98% de las universidades ya están operando de manera remota desde el 2020 , lo cual indica que aún se continuará con la educación virtual.

Mi motivación para realizar esta investigación sobre el Estrés estudiantil universitario basado en redes neuronales profundas, en pandemia covid-19, es aplicar los diversos conocimientos obtenidos en la carrera y en mi experiencia profesional, en la realidad social de nuestro país, y ver cómo se puede prevenir los futuros problemas aplicando diversas tecnologías.

Los objetivos específicos son: a) Emplear las redes neuronales profundas basado en sistemas conexionistas para determinar el estrés estudiantil universitario como afecciones asociadas a la depresión y abatimiento, en pandemia covid-19, 2021. b) Emplear las redes neuronales profundas basado en funciones de activación para determinar el estrés estudiantil universitario como afecciones asociadas a la ansiedad y angustia, en pandemia covid-19, 2021. c)Emplear las redes neuronales profundas basado en la propagación hacia atrás para determinar el estrés estudiantil universitario como afecciones asociadas a trastornos digestivos y de sueño, en pandemia covid-19, 2021.

CAPÍTULO I

PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1. DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

Según el Diario El Comercio (2020), el covid - 19 ha tenido una gran repercusión en el ámbito educativo, cambiando la forma de trabajo del sistema educativo universitario. Para volver a su ámbito de trabajo, en el 2020, las 144 universidades del Perú tuvieron que implementar diversas plataformas para lograran llevar a cabo las clases virtuales, además de capacitar a la población estudiantil en el uso de estos recursos tecnológicos, para así cumplir con las disposiciones dadas por el Ministerio de educación

La Superintendencia Nacional de Educación Superior Universitaria (SUNEDU) ha establecido programas en diversas universidades, con modalidades semipresencial y a distancia, cumpliendo con los lineamientos de educación superior universitaria, en el estado de emergencia sanitaria de Covid-19. Esto está orientado a reglamentar los aspectos formativos que garanticen un adecuado desarrollo de los programas, así como la interacción entre docentes y estudiantes. Así, se estableció que las universidades deben contar con docentes calificados para realizar dichas actividades y utilizar los mecanismos TIC (Tecnologías de la Información y la Comunicación). Las universidades deben contar con infraestructura tecnológica que garantizara el funcionamiento de las plataformas virtuales para el desempeño óptimo de las clases.

Según el portal web SomosPeriodismo (2021), las universidades del Perú, no estaban preparadas para la realización de clases virtuales. La psicóloga Mónica Cassaretto, presidente del Comité de Promoción y Cuidado de la Salud Mental en la Pontificia Universidad Católica del Perú, explica que las clases online deben ajustarse a las nuevas circunstancias y tener en cuenta las limitaciones de los estudiantes. Según datos de la Oficina de Servicio de Apoyo al Estudiante (OSOE) de la PUCP, de abril a noviembre de 2020, 2062 estudiantes universitarios fueron atendidos a través del programa virtual de Acompañamiento Psicosocial. Los diagnósticos más comunes entre los alumnos fueron ansiedad (49%), estrés académico (36%) y estado depresivo (32%).

Para Cassereto el apoyo psicológico en estos tiempos de pandemia es algo indispensable, pues según estadísticas oficiales, 174 mil estudiantes de educación universitaria abandonaron sus estudios. Los diferentes padecimientos que se empezaron a manifestar en los estudiantes a nivel psicológico varían desde malestares leves hasta deserción universitaria. La falta de el seguimiento psicológico a los estudiantes en esta situación podría llevarlos a un colapso emocional que detonarían en intentos de suicidio.

En el año 2019 el Ministerio de Educación público una serie de lineamientos que contempla que en las universidades del país se debe asegurar la salud mental de los estudiantes universitarios, esto con la finalidad de la adecuada integración de la sociedad y los miembros de la comunidad universitaria. Así mismo el 2020 se aprobó la actualización de estos lineamientos debido a los acontecimientos de cuarentena, donde se recomienda a las universidades contar con un Centro de Salud Mental Comunitaria Universitario ("CSMCU"), el cual debería brindar servicios de telemedicina que contribuyan a la salud mental.

Según el INEI (2021) el 63,3% de los hogares del área Lima Metropolitana disponen del servicio de internet, del Resto urbano, el 52,5% y en los hogares del Área rural el 13,2%; además viendo los datos del área de residencia, en los hogares de Lima Metropolitana el 50,2% disponen por lo menos de una computadora, en el Resto urbano el 38,1% y solo el 7,2%, de los hogares del área rural.

Tabla 1

Porcentaje de hogares con internet y computadoras.

	Enero a marzo 2020		Enero a marzo 2021	
Area de residencia	Computadora	Internet	Computadora	Internet
Lima metropolitana	52.9	62.5	50.2	63.3
Resto urbano	38.3	40.5	38.1	52.5
Área rural	7.5	5.9	7.2	13.2

Nota: Esta tabla muestra como varia los recursos tecnológicos en el Perú – INEI, 2021.

También se supo que en el trimestre enero a marzo del 2021, de la población que usa internet el 88.5 % lo hace a través del celular; el 16.7 % lo hace a través de una laptop; el 14 % usa computadoras de escritorio, el 2 % usa Tablet y el 7.9 % usan otros dispositivos, como televisores Smart.

La pandemia tomo desprevenidos a muchos sectores y uno de los sectores que más ha tenido que adaptarse es el sector educativo, para esto han tenido que adaptarse a la educación virtual, el gobierno ha tratado de dar lineamientos a las diferentes instituciones educativas y estas han tratado de adaptarse en el tiempo más corto posible con los recursos que encontraban disponibles; para los estudiantes el adaptarse a esta situación les ha generado ansiedad y estrés, además del miedo al contagio y preocupación de sus situaciones económicas y sociales que también se han visto afectados. Debido a esta situación es necesario que las universidades mejoren la sensibilización frente a cómo estudiar de manera segura desde esta modalidad y por esto ha surgido la idea de automatizar el diagnóstico del estrés estudiantil universitario, que parece ser necesaria para ayudar al control de la salud mental en la actualidad.

1.2. DEFINICIÓN DEL PROBLEMA DE INVESTIGACIÓN

PROBLEMA GENERAL

¿Cómo las redes neuronales profundas determinan el estrés estudiantil universitario, en pandemia covid-19, 2021?

PROBLEMAS ESPECÍFICOS

a. ¿Cómo las redes neuronales profundas basado en sistemas conexionistas determinan el estrés estudiantil universitario como afecciones asociadas a la depresión y abatimiento, en pandemia covid-19, 2021?

b. ¿Cómo las redes neuronales profundas basado en funciones de activación determinan el estrés estudiantil universitario como afecciones asociadas a la ansiedad y angustia, en pandemia covid-19, 2021?

c. ¿Cómo las redes neuronales profundas basado en la propagación hacia atrás determinan el estrés estudiantil universitario como afecciones asociadas a trastornos digestivos y de sueño, en pandemia covid-19, 2021?

1.3. OBJETIVO GENERAL

Utilizar las redes neuronales profundas para determinar el estrés estudiantil universitario, en pandemia covid-19, 2021.

1.4. OBJETIVOS ESPECÍFICOS

a. Emplear las redes neuronales profundas basado en sistemas conexionistas para determinar el estrés estudiantil universitario como afecciones asociadas a la depresión y abatimiento, en pandemia covid-19, 2021.

b. Emplear las redes neuronales profundas basado en funciones de activación para determinar el estrés estudiantil universitario como afecciones asociadas a la ansiedad y angustia, en pandemia covid-19, 2021.

c. Emplear las redes neuronales profundas basado en la propagación hacia atrás para determinar el estrés estudiantil universitario como afecciones asociadas a trastornos digestivos y de sueño, en pandemia covid-19, 2021.

1.5 JUSTIFICACIÓN

Actualmente la pandemia ha modificado nuestra forma de vida, entre estos cambios que se han dado, se ha modificado la forma de enseñanza universitaria, la cual, al ser virtual, necesita evaluar no solo el rendimiento académico, sino también debería velar por su bienestar integral, es por eso que es importante empezar a tener un control sobre la salud mental estudiantil y ver los diferentes indicadores que lo describen.

Por lo tanto, esta investigación pretende dar una herramienta que ayude en la determinación del grado de estrés que desarrollan los estudiantes universitarios, la información que se obtenga de la investigación dará a las universidades una herramienta que contribuirá a mejorar la salud mental universitaria.

1.6 DELIMITACIÓN

La investigación se realizó con estudiantes peruanos universitarios que estén teniendo clases virtuales en la cuarentena, se estudió la información que se registró en la encuesta que sirvió como instrumentos para recoger los datos utilizados; la recolección de datos se realizó el segundo semestre del 2021.

CAPÍTULO II

MARCO TEÓRICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

Para Anguiano (2018) en su tesis sobre “Redes neuronales profundas en energías renovables”, Indica que el machine learning actualmente es un área en apogeo pues ha demostrado que puede resolver problemas de diferentes entornos, además puede procesar datos a gran escala y dar como resultados predicciones y clasificaciones de estos datos analizados. El aprendizaje de este campo es una ventaja en la actualidad, pues ayuda a la optimización y automatización de diferentes tareas cotidianas.

Según Huamán (2020) en su tesis “Estrés académico y miedo a contraer coronavirus en universitarios del primer y último año de la carrera de psicología de una universidad privada lima – sur”, concluyo que al analizar el tipo de respuesta que genera el estrés académico, se determinó que los estudiantes de últimos ciclos manifiestan mayores respuestas corporales y psicológicas como consecuencia del estrés académico.

Estrada, Mamani, M., Gallegos, Mamani, H., Zuloaga (2021), en su investigación “Estrés académico en estudiantes universitarios peruanos en tiempos de la pandemia del COVID-19” concluyeron que en la pandemia del covid-19 los estudiantes de educación, desarrollaron un mayor nivel de estrés académico, esto vario según el sexo, año de estudio, grupo socioeconómico. Por ellos recomienda a las diferentes universidades identificar el nivel de estrés de los estudiantes a manera temprana, para prevenir las consecuencias que a largo plazo podrían ser irremediables. También recomienda diseñar estrategias psicológicas de intervención que ayuden a mejorar la calidad de vida de los estudiantes.

2.2. MARCO TEÓRICO

2.2.1. COVID - 19

De acuerdo a la Organización Panamericana de la Salud (2020) los coronavirus son un grupo de virus ARN altamente diversos de la familia Coronaviridae, que se dividen en cuatro géneros alfa, beta, gamma y delta y causan enfermedades de leves a graves en humanos y animales. Existen coronavirus humanos endémicos como los alfa coronavirus 229E y

NL63 y los beta coronavirus OC43 y HKU1 que pueden causar enfermedades como resfriado o neumonía en humanos. Sin embargo, han surgido dos coronavirus zoonóticos que causan enfermedades graves en humanos: el coronavirus del síndrome respiratorio agudo grave (SARS-CoV) en 2002 y 2003 y el coronavirus del síndrome respiratorio de Oriente Medio (MERS-CoV).

Para inicios del año 2020 en China se detectó a un nuevo agente que generaba un diferente caso de neumonía, este fue determinado como una beta coronavirus, el 11 de febrero del 2020 se designó a este virus como coronavirus de síndrome respiratorio grave o llamado también SARS - COV2, ese día la Organización mundial de la salud denomino a la enfermedad producida por este virus como COVID - 19.

Tras los elevados casos de contagio del coronavirus el 11 de marzo del 2020, la Organización Mundial de la Salud (OMS) califico a esta enfermedad como una pandemia, Tedros Adhanom, director de la OMS, declaro que esta no es solo una crisis de salud pública, es una crisis que afectará a todos los sectores, por lo que cada sector y cada individuo deben participar en la lucha. Desde el principio, afirmé que todos los países debían de adoptar un enfoque coordinado entre gobiernos y sociedad, construyendo una estrategia integral para prevenir infecciones, salvar vidas y minimizar el impacto.

Según reportes del Ministerio de Salud del Perú (2021), actualmente el número de infectados y fallecidos por coronavirus ha ido disminuyendo, desde la segunda ola que fue a inicios del mes de mayo del 2021, siendo la región de Lima Metropolitana la que registra las estadísticas más altas seguida por el departamento de Arequipa.

2.2.2. ESTUDIANTES UNIVERSITARIOS

La ley universitaria (2014) define a los estudiantes universitarios son quienes, habiendo concluido los estudios de educación secundaria, aprobaron el proceso de admisión a la universidad y se encuentran matriculados en ella.

En la Política de aseguramiento de la calidad de la educación superior universitaria, se define cinco principios que rigen el sistema universitario, el primero es la Autonomía y rectoría responsables. La autonomía universitaria tiene como objetivo garantizar el libre proceso de producción, transmisión y difusión del conocimiento académico. Por su parte, la

rectoría del Estado busca garantizar el desarrollo integral del estudiante universitario, el bien común de la sociedad y la finalidad pública de toda formación universitaria; el segundo principio es el estudiante como centro. Todos los actores involucrados en el Sistema Universitario concentran sus acciones en el bienestar del estudiante y la mejora de la calidad del servicio educativo que este recibe.

El estudiante universitario es formado de forma integral, esto incluye la parte social, intelectual y el desarrollo de su integración con la sociedad, esto ayuda a desarrollar no solo sus capacidades académicas, sino también artísticas y morales. Para cuando el estudiante universitario concluye con su carrera, este se podrá integrarse con facilidad al mundo laboral, y desempeñarse en este como una persona competitiva y valorada en el mercado laboral.

El tercer principio es la inclusión y equidad, todos los actores involucrados en el Sistema Universitario promueven y garantizan el acceso, permanencia y culminación satisfactoria de los estudios universitarios a todos los jóvenes del país, sin distinción de lengua, etnia, religión, sexo u otra causa de discriminación. De esta manera, la educación superior universitaria permite una efectiva movilidad social y económica, convirtiéndose en factor estratégico para lograr mayor equidad e inclusión social.

El cuarto principio es la calidad y la excelencia académica, el desarrollo de las universidades, va relacionado al desarrollo de la sociedad y la calidad que se exige de la formación académica viene dictada por las ofertas y demandas del mercado laboral. El estado debe poner lineamientos que aseguren las condiciones básicas y normativas para asegurar el cumplimiento de estos lineamientos, con el fin de desarrollar profesionales de calidad, exigiendo a los estudiantes excelencia académica y que busquen reconocimientos internacionales.

El quinto principio es el desarrollo del país, la finalidad de la educación universitaria es la formación integral de los estudiantes, esto con el fin de que contribuyan a la solución de los diversos problemas de la sociedad, basados en los conocimientos desarrollados y en la innovación que estos generan. Al formar ciudadanos conscientes de su entorno, se asegura que el conocimiento sea un instrumento para el desarrollo del país.

2.2.3. ESTRÉS

Cannon identificó por primera vez la reacción de estrés como la respuesta de lucha o huida. Posteriormente, el término fue utilizado de forma generalizada por Selye, quien defendió el síndrome de adaptación general. En el pasado, el estrés se consideraba como estímulos o acontecimientos externos que amenazaban a un organismo o como las respuestas de un organismo a estresores. Además, se refería a la interacción entre un organismo y el entorno.

En la actualidad Bong, K (2018), define el estrés como estímulos externos o internos que son significativamente percibidos por la persona, activan las emociones y provocan cambios fisiológicos que amenazan la salud y la supervivencia. Este concepto de estrés subraya que las respuestas al estrés de un individuo dependen de su interpretación subjetiva de un acontecimiento, más que de un acontecimiento en sí mismo. Por lo tanto, un mismo acontecimiento o situación puede percibirse de forma diferente, lo que da lugar a resultados muy distintos: salud o enfermedad. Es decir, lo que puede ser intolerable para una persona puede ser aceptable para otra.

CAUSAS DEL ESTRÉS

Las principales causas del estrés se clasifican en factores psicosociales y ambientales. Esta clasificación se basa en las causas directas del estrés.

Factores psicosociales

Hay cuatro factores psicosociales, como la adaptación (por ejemplo, cambios en el trabajo, la escuela y la familia), la frustración (por ejemplo, los prejuicios, la discriminación relacionada con la religión, la raza origen nacional, generación y región, estatus social, sexo, características físicas, socioeconómicas, burocracias), la sobrecarga (por ejemplo, laboral, académica y doméstica), y privación (por ejemplo, aburrimiento, soledad).

Factores ambientales

Los factores ambientales se refieren a los estímulos, derivados de nuestra relación con el entorno, que producen respuestas de estrés en la mayoría de los individuos a través de un mecanismo biológico innato. Estos factores de estrés incluyen los ritmos biológicos (por ejemplo, la alteración del ciclo sueño-vigilia, la menstruación), el ruido, la contaminación y el cambio climático.

La Asociación Americana de Psicología (2010), ha clasificado el estrés en diferentes tipos: estrés agudo, estrés agudo episódico y estrés crónico.

Estrés agudo

El estrés agudo es la forma de estrés más común. Surge de las exigencias y presiones del pasado reciente y las exigencias y presiones anticipadas del futuro cercano. El estrés agudo es emocionante y fascinante en pequeñas dosis, pero cuando es demasiado resulta agotador.

Dado que es a corto plazo, el estrés agudo no tiene tiempo suficiente para causar los daños importantes asociados con el estrés a largo plazo. Los síntomas más comunes son:

- Agonía emocional: una combinación de enojo o irritabilidad, ansiedad y depresión, las tres emociones del estrés.
- Problemas musculares que incluyen dolores de cabeza tensos, dolor de espalda, dolor en la mandíbula y las tensiones musculares.
- Problemas estomacales e intestinales como acidez, flatulencia, diarrea, estreñimiento y síndrome de intestino irritable.
- Sobreexcitación pasajera que deriva en elevación de la presión sanguínea, ritmo cardíaco acelerado, transpiración de las palmas de las manos, palpitaciones, mareos, migrañas, manos o pies fríos, dificultad para respirar, y dolor en el pecho.

El estrés agudo puede presentarse en la vida de cualquiera, y es muy tratable y manejable.

Estrés agudo episódico

Por otra parte, están aquellas personas que tienen estrés agudo con frecuencia, cuyas vidas son tan desordenadas que son estudios de caos y crisis. Asumen muchas responsabilidades, tienen demasiadas cosas entre manos y no pueden organizar la cantidad de exigencias autoimpuestas ni las presiones que reclaman su atención. Parecen estar perpetuamente en las garras del estrés agudo.

Es común que las personas con reacciones de estrés agudo estén demasiado agitadas, tengan mal carácter, sean irritables, ansiosas y estén tensas. Esto describe personalidades propensas a los problemas cardíacos descrita por los cardiólogos Meter Friedman y Ray Rosenman, es similar a un caso extremo de estrés agudo episódico. Además, existe una forma de hostilidad sin razón aparente, pero bien racionalizada, y casi siempre una inseguridad profundamente arraigada.

Los síntomas del estrés agudo episódico son los síntomas de una sobre agitación

prolongada: dolores de cabeza tensos y persistentes, migrañas, hipertensión, dolor en el pecho y enfermedad cardíaca. Tratar el estrés agudo episódico requiere la intervención en varios niveles, que por lo general requiere ayuda profesional, la cual puede tomar varios meses.

Estrés crónico

Si bien el estrés agudo puede ser emocionante y fascinante, el estrés crónico no lo es. Este es el estrés agotador que desgasta a las personas día tras día, año tras año. El estrés crónico destruye al cuerpo, la mente y la vida. Hace estragos mediante el desgaste a largo plazo. El estrés crónico surge cuando una persona nunca ve una salida a una situación deprimente. Es el estrés de las exigencias y presiones implacables durante períodos aparentemente interminables. Sin esperanzas, la persona abandona la búsqueda de soluciones. El estrés crónico mata a través del suicidio, la violencia, el ataque al corazón, la apoplejía e incluso el cáncer. Las personas se desgastan hasta llegar a una crisis nerviosa final y fatal.

2.2.4. EDUCACION A DISTANCIA

Según Vásquez, Bongianino y Sosisky (2006) definen la educación virtual o a distancia como una comunicación, que se desarrolla con fines educativos, que involucra a un docente y a un estudiante, sin importar el espacio donde se encuentren con ayuda de herramientas tecnológicas que hacen de plataforma para la interacción de los participantes, es un tipo de modalidad que actualmente es una estrategia educativa.

La Superintendencia Nacional de Educación Superior Universitaria (SUNEDU) estableció las disposiciones para la autorización de programas, de universidades licenciadas, bajo las modalidades semipresencial y a distancia, dando cumplimiento a lo dispuesto en el artículo 3 del Decreto Legislativo N.º 1496, que establece disposiciones en materia de educación superior universitaria, en el marco del estado de emergencia sanitaria por Covid-19. Dicho decreto modificó el artículo 47 de la Ley Universitaria, que regulaba las características de la educación a distancia en el sistema universitario, ampliándose de dos a tres modalidades para la prestación del servicio educativo: presencial, semipresencial y a distancia o no presencial. Así establece que las universidades pueden desarrollar programas de educación a distancia, basados en entornos virtuales de aprendizaje.

La educación virtual o a distancia debe cumplir con los mismos lineamientos que se exige a la educación presencial, solo se debe diferenciar en el ámbito en el que se desarrolla.

2.2.5. INVENTARIO SISCO SV - 21

Barraza (2018), definió una segunda versión del inventario SISCO para el estudio del estrés académico, este instrumento fue validado con un nivel de fiabilidad de 87 %, este instrumento está constituido por 23 ítems distribuidos de la siguiente manera:

- Un ítem de filtro que, en términos dicotómicos (si-no), permite determinar si el encuestado es candidato o no a contestar el inventario. En caso de que el investigador lo considere puede utilizar este dato para reportar el nivel de presencia del estrés académico en la población encuestada.

- Un ítem que, en un escalamiento tipo Likert de cinco valores numéricos (del 1 al 5 donde uno es poco y cinco mucho), permite identificar el nivel de intensidad del estrés académico. Este dato el investigador lo puede utilizar como variable monoítem para medir la intensidad del estrés o, en su defecto, eliminarlo.

- Siete ítems que, en un escalamiento tipo Likert de seis valores categoriales (nunca, casi nunca, rara vez, algunas veces, casi siempre y siempre), permiten identificar la frecuencia en que las demandas del entorno son valoradas como estímulos estresores.

- Siete ítems que, en un escalamiento tipo Likert de seis valores categoriales (nunca, casi nunca, rara vez, algunas veces, casi siempre y siempre), permiten identificar la frecuencia con que se presentan los síntomas o reacciones ante un estímulo estresor.

- Siete ítems que, en un escalamiento tipo Likert de seis valores categoriales (nunca, casi nunca, rara vez, algunas veces, casi siempre y siempre), permiten identificar la frecuencia de uso de las estrategias de afrontamientos.

El instrumento tiene la siguiente forma:

1.- Durante el transcurso de este semestre ¿has tenido momentos de preocupación o nerviosismo (estrés)?

- Si

- No

En caso de seleccionar la alternativa “no”, el cuestionario se da por concluido, en caso de seleccionar la alternativa “si”, pasar a la pregunta número dos y continuar con el resto de las preguntas.

2.- Con la idea de obtener mayor precisión y utilizando una escala del 1 al 5 señala tu nivel de estrés, donde (1) es poco y (5) mucho.

1	2	3	4	5

3.-Dimensión estresores

Instrucciones: A continuación, se presentan una serie de aspectos que, en mayor o menor medida, suelen estresar a algunos alumnos. Responde, señalando con una X, ¿con que frecuencia cada uno de esos aspectos te estresa? tomando en consideración la siguiente escala de valores:

Tabla 2
Escala de valores.

Nunca	Casi nunca	Rara vez	Alguna vez	Casi siempre	Siempre
N	CN	RV	AV	CS	S

¿Con qué frecuencia te estresa:

Tabla 3
Listado de estresores.

Estresores	N	CN	RV	AV	CS	S
La sobrecarga de tareas y trabajos escolares que tengo que realizar todos los días						
La personalidad y el carácter de los/as profesores/as que me imparten clases						
La forma de evaluación de mis profesores/as (a través de ensayos, trabajos de investigación, búsquedas en Internet, etc.)						
El nivel de exigencia de mis profesores/as						
El tipo de trabajo que me piden los profesores (consulta de temas, fichas de trabajo, ensayos, mapas conceptuales, etc.)						
Tener tiempo limitado para hacer el trabajo que me encargan los/as profesores/as						
La poca claridad que tengo sobre lo que quieren los/as profesores/as						

4.- Dimensión síntomas (reacciones)

Instrucciones: A continuación, se presentan una serie de reacciones que, en mayor o menor medida, suelen presentarse en algunos alumnos cuando están estresados. Responde, señalando con una X, ¿con que frecuencia se te presentan cada una de estas reacciones cuando estás estresado? tomando en consideración la misma escala de valores del apartado anterior.

Con qué frecuencia se te presentan las siguientes reacciones cuando estás estresado:

Tabla 4
Listado de síntomas.

Sintomas	N	CN	RV	AV	CS	S
Fatiga crónica (cansancio permanente)						
Sentimientos de depresión y tristeza (decaído)						
Ansiedad, angustia o desesperación						
Problemas de concentración						
Sentimiento de agresividad o aumento de irritabilidad						
Conflictos o tendencia a polemizar o discutir						
Desgano para realizar las labores escolares						

5.- Dimensión estrategias de afrontamiento

Instrucciones: A continuación, se presentan una serie de acciones que, en mayor o menor medida, suelen utilizar algunos alumnos para enfrentar su estrés. Responde, encerrando en un círculo, ¿con que frecuencia utilizas cada una de estas acciones para enfrentar tu estrés? tomando en consideración la misma escala de valores del apartado anterior.

¿Con qué frecuencia utilizas cada una de estas acciones para enfrentar tu estrés:

Tabla 5
Listado de estrategias.

Estrategias	N	CN	RV	AV	CS	S
Concentrarse en resolver la situación que me preocupa						
Establecer soluciones concretas para resolver la situación que me preocupa						
Analizar lo positivo y negativo de las soluciones pensadas para solucionar la situación que me preocupa						
Mantener el control sobre mis emociones para que no me afecte lo que me estresa						
Recordar situaciones similares ocurridas anteriormente y pensar en cómo las solucione						
Elaboración de un plan para enfrentar lo que me estresa y ejecución de sus tareas						
Fijarse o tratar de obtener lo positivo de la situación que preocupa						

Control de calidad: para determinar como válidos los resultados de cada cuestionario, y por lo tanto aceptarlo e integrarlo a la base de datos, éste debe estar respondido en un porcentaje mayor al 70%. En ese sentido, se considera necesario que el cuestionario tenga contestados por lo menos 16 ítems de los 23 que lo componen, en caso contrario se anulará ese cuestionario en lo particular.

Baremo normativo centrado en la población en que se validó:

- De 0 a 48% nivel leve del estrés
- De 49% a 60% nivel moderado del estrés
- Del 61% al 100% nivel severo de estrés

Baremo indicativo centrado en el valor teórico de la variable:

- De 0 a 33% nivel leve del estrés
- De 34% a 66% nivel moderado del estrés
- Del 67% al 100% nivel severo de estrés

2.2.6. INTELIGENCIA ARTIFICIAL

Para Wolfgang (2017), la cuestión central para la informática, es la cuestión del machine learning o la máquina con comportamiento de persona, demostrando lograr obtener conocimientos de forma inteligente.

El atributo puede despertar conclusiones muy diferentes. Plantea la cuestión de si nuestro bien supremo, el alma, es algo que debemos entender o incluso intentar reconstruirlo.

En 1955, John McCarthy, un pionero de la inteligencia artificial, fue el primero que definió el término artificial, de la siguiente forma: La finalidad de la inteligencia artificial es desarrollar máquinas con comportamiento que parezca inteligente.

En la enciclopedia británica se encuentra una definición que dice así, la inteligencia artificial es la capacidad de los ordenadores para resolver problemas asociados a las capacidades superiores de procesamiento intelectual de los seres humanos. Pero esta definición también tiene puntos débiles. Según esta definición, todo ordenador es un sistema de IA.

Este dilema se resuelve elegantemente por la siguiente definición de Elaine Rich: la IA es el estudio de como las computadoras o maquinas logran realizar actividades, en las que los humanos somos superiores.
Tareas como la resolución de muchos cálculos en poco tiempo son los puntos fuertes de los ordenadores. Así, superan a los humanos por muchos múltiplos. Sin embargo, en muchas otras áreas, los humanos superan aun a las máquinas. De hecho, la investigación sobre robots autónomos es un tema importante y actual en la IA.

Sin embargo, sería peligroso decir de la definición de Rich que la inteligencia artificial sólo se ocupa de la aplicación pragmática de los procesos inteligentes. En el sentido de la definición de Rich, los sistemas inteligentes no pueden construirse sin los razonamientos humanos y de la acción inteligente, por lo que la neurociencia es importante para la IA.
Un punto fuerte de la inteligencia humana es la adaptabilidad y cambiar nuestro comportamiento es consecuencia del aprendizaje, la cual es muy superior a la de los de los ordenadores.

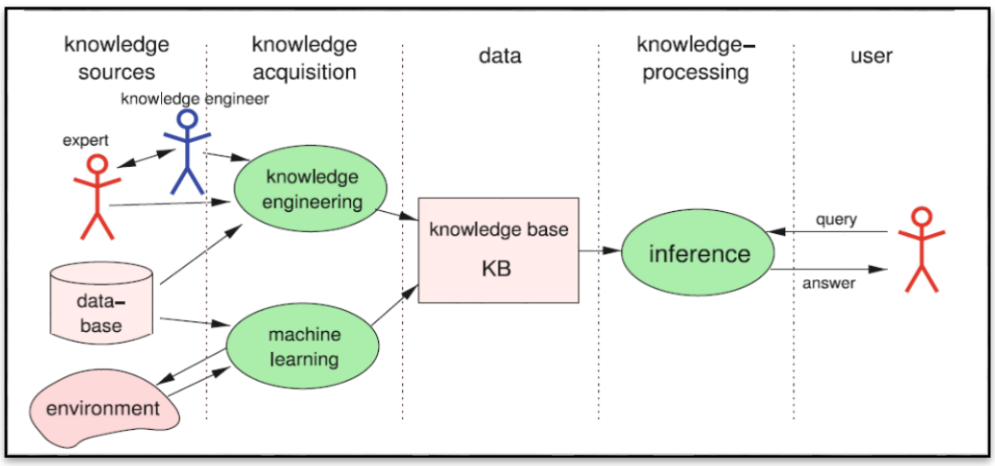
SISTEMAS BASADOS EN EL CONOCIMIENTO

Un agente es un programa que implementa un mapeo de percepciones a acciones. Para aplicaciones complejas en las que el agente debe ser capaz de confiar en una gran cantidad de información y debe realizar una tarea difícil, la programación del agente puede ser muy costosa y poco clara la forma de proceder. En este caso, la IA proporciona un camino claro que simplifica el trabajo.

En primer lugar, separamos el conocimiento del sistema o programa, que utiliza el conocimiento para, por ejemplo, llegar a conclusiones, responder a consultas o elaborar un plan. Este sistema se llama mecanismo de inferencia.

El conocimiento se almacena en una base de conocimientos (KB). La adquisición de conocimientos en la base de conocimientos se denomina ingeniería del conocimiento y se basa en diversas fuentes de conocimiento, como como los expertos humanos, el ingeniero del conocimiento y las bases de datos.

Figura 1
Estructura clásica de un Sistema de conocimientos.



Nota: Fuente (Wolfgang ,2017)

Avanzar hacia una separación del conocimiento y la inferencia tiene varias ventajas cruciales. La separación del conocimiento y la inferencia puede permitir que los sistemas de inferencia se implementen de una manera ampliamente independiente de la aplicación.

Al desvincular la base de conocimientos de la inferencia, el conocimiento puede almacenarse de forma declarativa. En la base de conocimientos sólo hay una descripción del conocimiento, que es independiente del sistema de inferencia en uso. Sin esta separación clara, el conocimiento y el procesamiento de los pasos de inferencia estarían entrelazados y cualquier cambio en el conocimiento sería muy costoso.

2.2.7. DATA SCIENCE

Según Dumbill, et al. (2013), el data science aplica métodos cuantitativos y cualitativos para solucionar problemas y predecir diferentes datos. Una de las revelaciones más destacadas de la actualidad, con la enorme y creciente cantidad de datos, es que el conocimiento y el análisis del dominio no se pueden separar.

Para Qamar (2020), Data science es un campo multidisciplinar centrado en el estudio de todos los aspectos de los datos, desde su generación y procesamiento hasta su conversión en una valiosa fuente. Como se ha dicho, es un campo multidisciplinar, por lo que utiliza conceptos de las matemáticas, la estadística, el aprendizaje automático y la inteligencia artificial, etc., con amplia gama de diferentes aplicaciones.

Al darse cuenta de la importancia de los datos, todas las organizaciones y empresas se esfuerzan por aplicar análisis y técnicas de ciencia de datos para el bien del negocio.

Las empresas más grandes, como Google, Amazon y Yahoo, etc., han demostrado que el almacenamiento cuidadoso de los datos y su posterior uso para la extracción del conocimiento para tomar decisiones y adoptar nuevas políticas son siempre un valor. Por lo tanto, también se están realizando esfuerzos para disminuir el costo de aplicar ciencia de datos. Ahora explicaremos las diferentes fases de vida de un proyecto de ciencia de datos.

Fase 1 Descubrimiento: La primera fase de un proyecto de ciencia de datos es el descubrimiento. Se trata de descubrimiento de las fuentes disponibles que tiene, que necesita, sus requisitos, sus necesidades, sus resultados, es decir, lo que desea obtener del proyecto, su viabilidad, la infraestructura necesaria, etc.

Fase 2 La preparación de los datos: Esta fase se relaciona a explorar los datos y su valor. Es posible que tenga que realizar un pre procesamiento que incluya el proceso ETL.

Fase 3 Planificación del modelo: Ahora se piensa en el modelo que se implementará para extraer el conocimiento de los datos. El modelo funcionará como base para descubrir patrones y correlaciones en los datos según el resultado que se desee.

Fase 4 Desarrollo del modelo: La siguiente fase es la construcción del modelo que tiene como base los datos de entrenamiento y de prueba. Puede utilizar varias técnicas como la clasificación y clustering, etc., en función de sus necesidades y de la naturaleza de los datos.

Fase 5 Operar: La siguiente fase consiste en implementar el proyecto. Esto puede incluir la entrega del código, la instalación del proyecto, la entrega de los documentos y la realización de demostraciones, etc.

Fase 6 Comunicar los resultados: La fase final es la evaluación, es decir, usted evalúa su proyecto en función de diversas medidas como la satisfacción del cliente, la consecución del objetivo para el que se desarrolló el proyecto y la precisión de los resultados del proyecto, etc.

2.2.8. MACHINE LEARNING

Litjens G, et al. (2017), consideran que los modelos de aprendizaje automático han demostrado un gran éxito en el aprendizaje de patrones complejos que les permiten hacer predicciones sobre datos no observados. Además de usar modelos para la predicción, la capacidad de interpretar lo que un modelo ha aprendido está recibiendo una cantidad cada vez mayor de atención. Sin embargo, este mayor enfoque ha llevado a una confusión considerable sobre la noción de interpretabilidad. En particular, no está claro cómo se relaciona la amplia gama de métodos de interpretación propuestos, y qué conceptos comunes se pueden utilizar para evaluarlos.

Según Ertel (2017), la tarea del machine learning o máquina de aprendizaje, es extraer conocimiento de los datos de entrenamiento. A menudo, el desarrollador o el usuario quieren que la máquina de aprendizaje haga que el conocimiento extraído sea legible también para los humanos. Es aún mejor si el desarrollador puede incluso alterar el conocimiento. Los árboles de decisión son un ejemplo de este método.

2.2.9. DATA MINING

Para Ertel (2017), el proceso de adquisición de conocimientos a partir de los datos, así como su representación y su aplicación, se denomina Data Mining o minería de datos. Los

métodos utilizados suelen estar tomados de la estadística o el aprendizaje automático y deben ser aplicables a cantidades muy grandes de datos a un coste razonable.

Bramer (2016), define que junto con los avances tecnológicos de almacenamiento, hacen cada vez más posible el almacenamiento de datos a un coste relativamente bajo, ha llegado de que esos datos contienen conocimientos enterrados que pueden ser fundamental para el crecimiento o el declive de una empresa, conocimientos que podrían conducir a importantes descubrimientos científicos, conocimientos que podrían permitirnos predecir el tiempo y las catástrofes naturales, conocimientos que podrían permitirnos identificar las causas y posibles curas de enfermedades mortales, conocimientos que significarían hacer la diferencia entre la vida y la muerte. Sin embargo los enormes volúmenes de datos, la mayoría de ellos se almacenan y no se examinan más que de forma superficial, si es que se hace. Se ha dicho, con razón, que el mundo se está volviendo "rico en datos, pero pobre en conocimientos".
El machine learning, ayuda con los datos de las organizaciones, gobiernos e individuos.

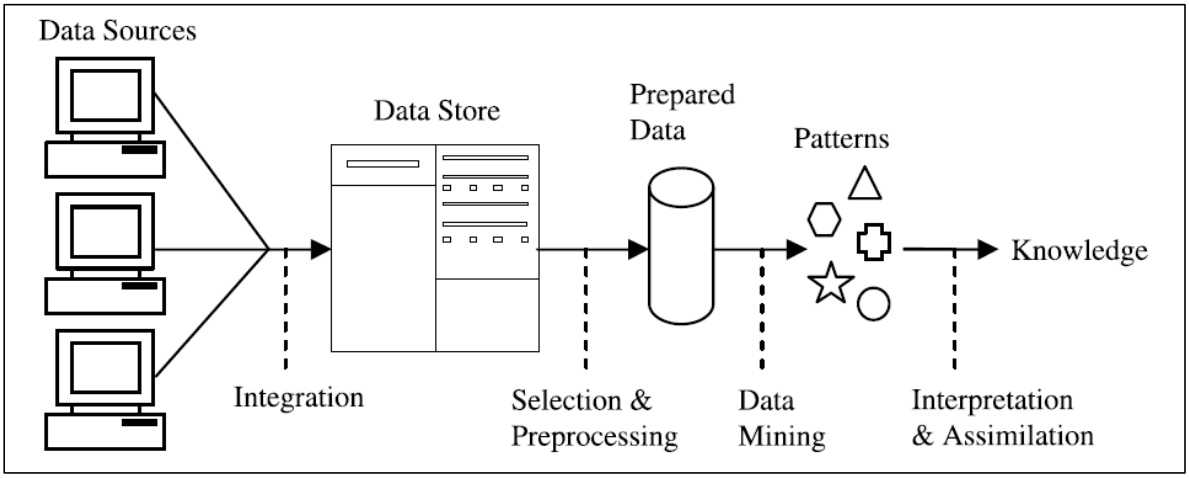
DESCUBRIMIENTO DEL CONOCIMIENTO

El descubrimiento de conocimientos define la "extracción no trivial de información implícita previamente desconocida y potencialmente útil de los datos". Se trata de un proceso del que la minería de datos es sólo una parte, aunque central.

Los datos llegan, posiblemente de muchas fuentes. Se integran y se colocan en algún almacén de datos común. Una parte de ellos se toma y se pre procesa en un formato estándar. Estos "datos preparados" se pasan a un algoritmo de minería de datos que produce una salida en forma de reglas o algún otro tipo de "patrones".

Figura 2

Proceso de descubrimiento del conocimiento



Nota: Fuente (Bramer ,2016)

Esta breve descripción deja claro que, aunque la minería de datos es importante, el pre procesamiento de los datos y su posterior interpretación de los resultados tienen gran importancia. Se trata de tareas especializadas que son mucho más un arte (o una habilidad aprendida de la experiencia) que una ciencia exacta.

2.2.10. REDES NEURONALES ARTIFICIALES

Según Galushkin (2007), las redes neuronales son un procedimiento con forma de un grafo con dirección, en la cual se inicia con entradas a las cuales se aplica acciones que dan por resultado muevas salidas, los elementos del proceso se llaman nodos de la red neuronal. El sistema de redes neuronales es una herramienta para describir un algoritmo basado en redes neuronales. Consideramos que son posibles cuatro enfoques para la investigación de las redes neuronales:

1. Enfoque psicológico, cuando es necesario modelar algún paradigma psicológico que requiere un desarrollo e investigación de la red neuronal con alguna estructura definida.

2. Enfoque neurofisiológico, cuando la red neuronal se desarrolla e investiga sobre la base del conocimiento de la estructura de alguna parte del cerebro. La red neuronal modela las funciones de esta parte del cerebro.

3. Enfoque algorítmico, cuando se formula algún problema matemático y se diseña una red neuronal adecuada con el correspondiente algoritmo ajustado a esta solución se

diseña a partir de esta formulación.

4. Enfoque sistemático, que combina todos los enfoques anteriores.

Para Ertel (2017), las redes neuronales son redes de células nerviosas del cerebro de los seres humanos y los animales. El cerebro humano tiene un promedio de 100.000 millones de células nerviosas. Los humanos debemos nuestra inteligencia y nuestra capacidad de aprendizaje de diversas capacidades motoras e intelectuales a los complejos relés y la adaptabilidad del cerebro. Alrededor de 1900 llegó la revolución de que estos pequeños bloques físicos del cerebro, las células nerviosas y sus conexiones, son responsables de la conciencia, las asociaciones, los pensamientos y la capacidad de aprendizaje.

El primer gran paso hacia las redes neuronales en la IA fue dado en 1943 por McCulloch y Pitts en un artículo titulado "Un cálculo lógico de las ideas inmanentes a la actividad nerviosa". Fueron los primeros en presentar un modelo matemático de la neurona como el elemento básico de conmutación del cerebro.

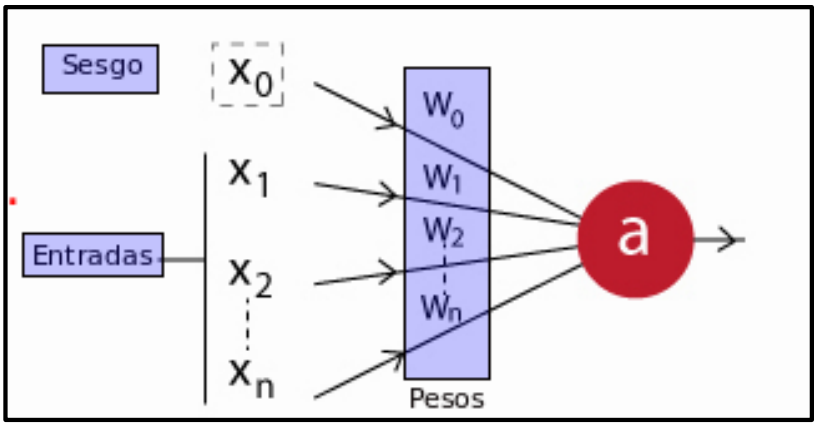
Este artículo sentó las bases para la construcción de redes neuronales artificiales y, por tanto, para esta importantísima rama de la IA. Podemos considerar a la modelización y la simulación de redes neuronales como ramas de la biónica dentro de la inteligencia artificial. Casi todas las áreas de la IA intentan recrear procesos cognitivos procesos cognitivos, como en la lógica o en el razonamiento probabilístico.

Sin embargo, los mecanismos utilizados para el modelado, es decir, las matemáticas, los diferentes lenguajes de programación y los ordenadores tienen muy poco en común con el cerebro humano. Con las redes neuronales artificiales el enfoque es diferente.

A partir de los conocimientos sobre el funcionamiento de las redes naturales, intentamos modelarlas, simularlas e incluso reconstruirlas en hardware.

Para López (2019), Las redes neuronales son un modelo computacional basado en un gran conjunto de unidades neuronales simples (neuronas artificiales), de forma aproximadamente análoga al comportamiento observado en los axones de las neuronas en los cerebros biológicos.
Cada una de estas neuronas simples, va a tener una forma similar a la figura 3.

Figura 3
Diagrama de Neurona simple.



Nota: Fuente (López ,2019)

En donde sus componentes son:

- X_1, X_2, …, Xn: Los datos de entrada en la neurona, los cuales también puede ser que sean producto de la salida de otra neurona de la red.

- X_0: La unidad de sesgo; un valor constante que se le suma a la entrada de la función de activación de la neurona. Generalmente tiene el valor 1. Este valor va a permitir cambiar la función de activación hacia la derecha o izquierda, otorgándole más flexibilidad para aprender a la neurona.

- W_0, W_1, W_2, …, Wn: Los pesos relativos de cada entrada. Tener en cuenta que incluso la unidad de sesgo tiene un peso.

- a: La salida de la neurona. Que va a ser calculada de la siguiente forma:

$$\alpha = f\left(\sum_{i=0}^{n} w_i . x_i\right)$$

Aquí f es la función de activación de la neurona. Esta función es la que le otorga tanta flexibilidad a las redes neuronales y le permite estimar complejas relaciones no lineales en los datos. Puede ser tanto una función lineal, una función logística, hiperbólica, etc.

Cada unidad neuronal está conectada con muchas otras y los enlaces entre ellas pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. Estos sistemas aprenden y se forman a sí mismos, en lugar de ser programados de forma explícita, y sobresalen en áreas donde la detección de soluciones o características es difícil de expresar con la programación convencional.

2.2.11. DEEP LEARNING

Para Aggarwal (2018), la idea básica del aprendizaje profundo es que la composición repetida de funciones a menudo puede reducir los requisitos sobre el número de funciones base (unidades computacionales) por un factor que está relacionado exponencialmente con el número de capas en la red. Por lo tanto, a pesar de que el número de capas en la red aumenta, el número de parámetros necesarios para aproximar la misma función se reduce drásticamente. Esto aumenta el poder de generalización de la red.

Según Bengio Y, et al. (2016), el aprendizaje profundo se basa en formalismos de modelado que los investigadores pueden usar para guiar sus esfuerzos de diseño y describir sus algoritmos. Uno de estos formalismos es la idea de modelos probabilísticos estructurados. Los modelos probabilísticos son un ingrediente clave de muchas de las investigaciones más importantes como temas de aprendizaje profundo.

Para Nakamoto (2017), el aprendizaje profundo es un término que indica un enfoque particular del diseño, desarrollo, prueba y especialmente entrenamiento de redes neuronales. El nacimiento y el desarrollo del aprendizaje profundo fue posible gracias a las mejoras tecnológicas que han aparecido a finales de la década de 2000, como GPU, es decir, grandes conjuntos de pequeños procesadores diseñados para procesar imágenes en tarjetas de video de Nvidia.

Esta tecnología, inicialmente creada para videojuegos, se descubrió que era muy rápida en el procesamiento de redes neuronales multicapa, con un rendimiento mucho mejor que las CPU normales. La segunda innovación que ha determinado que el advenimiento definitivo del aprendizaje profundo es el Big Data fenómeno, que es la disponibilidad de cantidades masivas y caleidoscópicas de datos gracias a la difusión y crecimiento de Internet y de los servicios conectado a él.

2.2.12. FUNCIONES DE ACTIVACIÓN

Para Andrade et. al (2017), existen 2 tipos de funciones de activación:

2.2.12.1. Partially differentiable activation functions

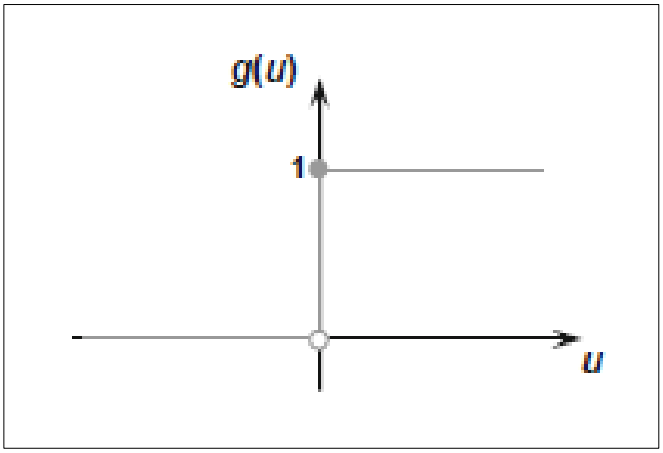
Las funciones de activación parcialmente diferenciables son funciones con puntos cuyo primer derivado de orden no existen. Las tres funciones principales de esta categoría son la siguientes: función de paso, función de paso bipolar y función de rampa simétrica.

a. Función de paso (limitador Heaviside/Hard)

El resultado producido por la función escalón asumirá valores unitarios positivos cuando el potencial de activación de la neurona es mayor o igual a cero; de lo contrario, el resultado será nulo. Así, tenemos:

$$g(u) = \begin{cases} 1, & if \ \ u \geq 0 \\ 0, & if \ \ u < 0 \end{cases}$$

Figura 4
Grafica de función de paso.



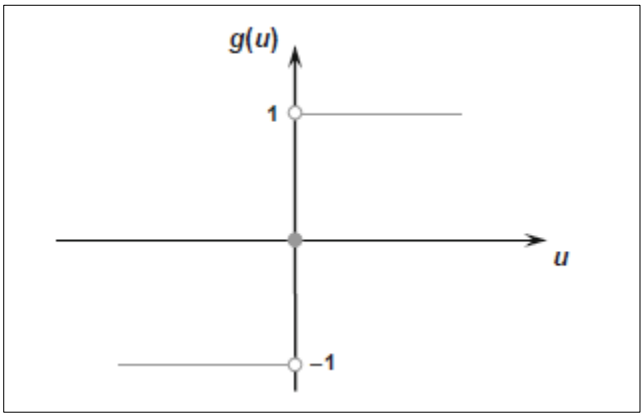
Nota: Fuente (Andrade et. al, 2017)

b. Función de paso bipolar o signal function (limitador duro simétrico)

El resultado producido por esta función asumirá valores positivos unitarios cuando el potencial de activación de la neurona es mayor que cero; valor nulo cuando el potencial también es nulo; y valores unitarios negativos cuando el potencial es menor que cero. Este comportamiento en notación matemática es:

$$g(u) = \begin{cases} 1, & if\ u > 0 \\ 0, & if\ u = 0 \\ -1, & if\ u < 0 \end{cases}$$

Figura 5
Grafica de la función de paso bipolar.



Nota: Fuente (Andrade et. al, 2017)

Los problemas relacionados con la clasificación de patrones, la función de paso bipolar puede ser aproximado por la siguiente expresión:

$$g(u) = \begin{cases} 1, & if\ u \geq 0 \\ -1, & if\ u < 0 \end{cases}$$

En esta circunstancia, otra alternativa es mantener la salida de la neurona sin cambios, por lo tanto:

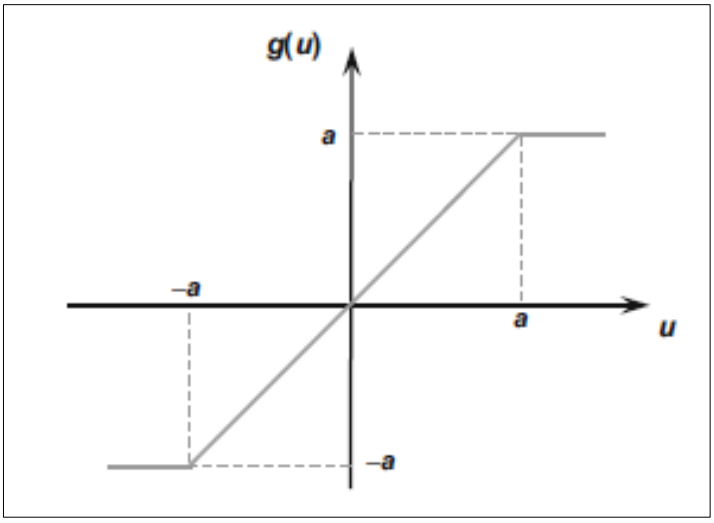
$$g(u) = \begin{cases} 1, & if\ u > 0 \\ previous\ output, & if\ u = 0 \\ -1, & if\ u < 0 \end{cases}$$

c. Función de rampa simétrica

Los valores devueltos por esta función son iguales a los valores de la activación potenciales en sí mismos cuando se definen dentro del rango [−a, a], y se limitan a los valores límite en caso contrario. La notación matemática para este comportamiento es como sigue:

$$g(u) = \begin{cases} a, & if\ u > a \\ u, & if\ -a \leq u < a \\ -a, & if\ u < a \end{cases}$$

Figura 6
Grafica de la función de rampa simétrica.



Nota: Fuente (Andrade et. al, 2017)

2.2.12.2. Fully differentiable activation functions

Las funciones de activación totalmente diferenciables son aquellas cuyas derivadas de primer orden existen para todos los puntos de su dominio de definición. Las cuatro funciones principales de esta categoría, que se pueden emplear en redes neuronales artificiales, son la función logística, tangente hiperbólica, función gaussiana y función lineal.

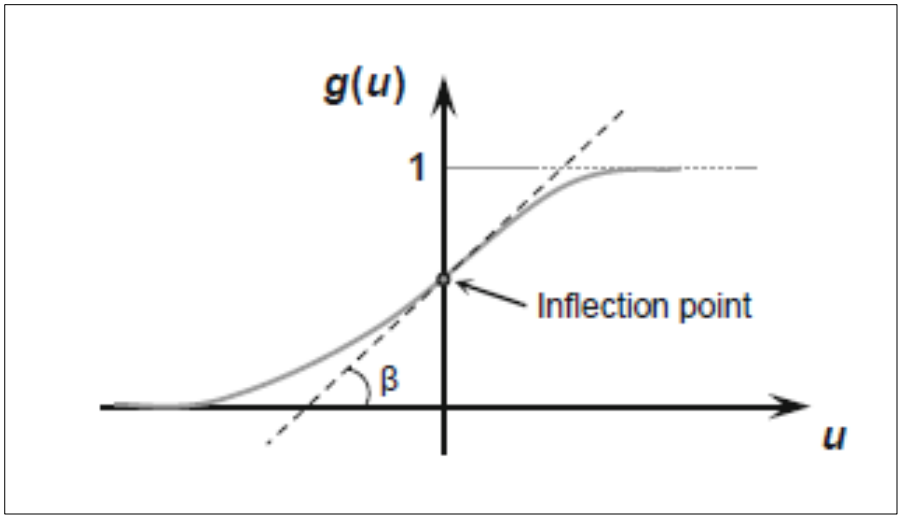
a. Función logística

El resultado de salida producido por la función logística siempre asumirá valores entre cero y uno. Su expresión matemática viene dada por:

$$g(u) = \frac{1}{1 + e^{-\beta . u}}$$

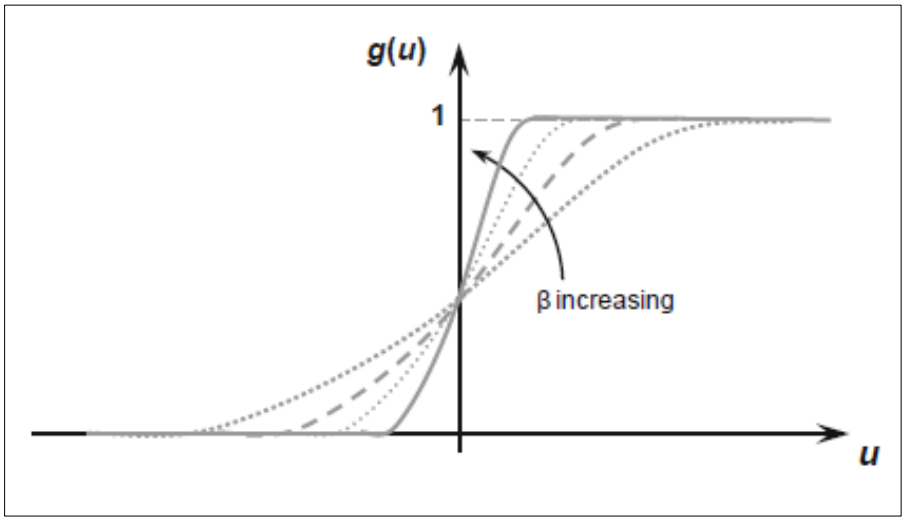
donde β es una constante real asociada a la función pendiente en su punto de flexión.

Figura 7
Grafica de función de activación logística.



Nota: Fuente (Andrade et. al, 2017)

Figura 8
Grafica de función de activación logística cuando el parámetro β cambia.



Nota: Fuente (Andrade et. al, 2017)

Del análisis de la figura 8, es posible concluir que la geometría formada de la función de activación logística es similar al de la función de paso, cuando β es muy alta, es decir, tiende a infinito. Sin embargo, a diferencia de la función de paso, la función logística es totalmente diferenciable en toda su definición dominio.

b. Función tangente hiperbólica

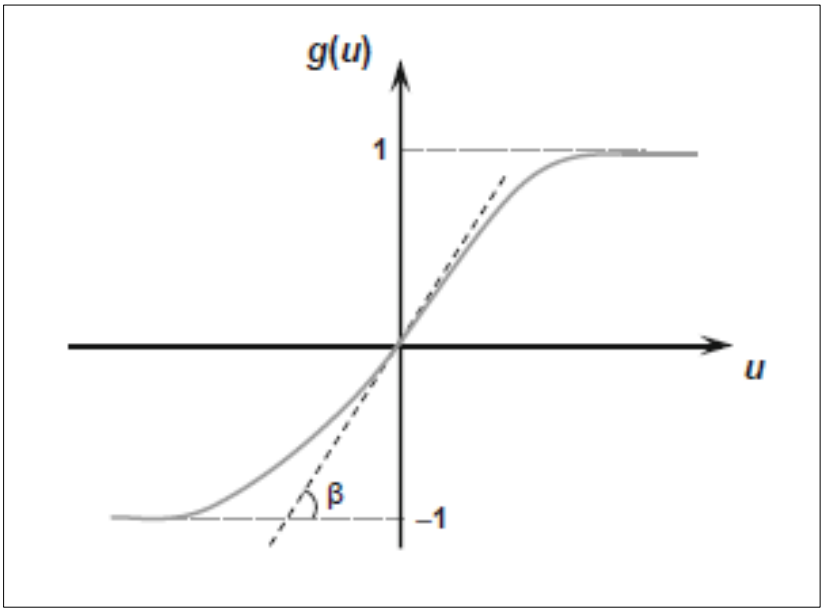
El resultado de salida, a diferencia del caso de la función logística, siempre asumirá valores reales entre −1 y 1, con la siguiente expresión matemática:

$$g(u) = \frac{1 - e^{-\beta . u}}{1 + e^{-\beta . u}}$$

donde β también está asociado a la pendiente de la función tangente hiperbólica con su punto de inflexión.

Figura 9

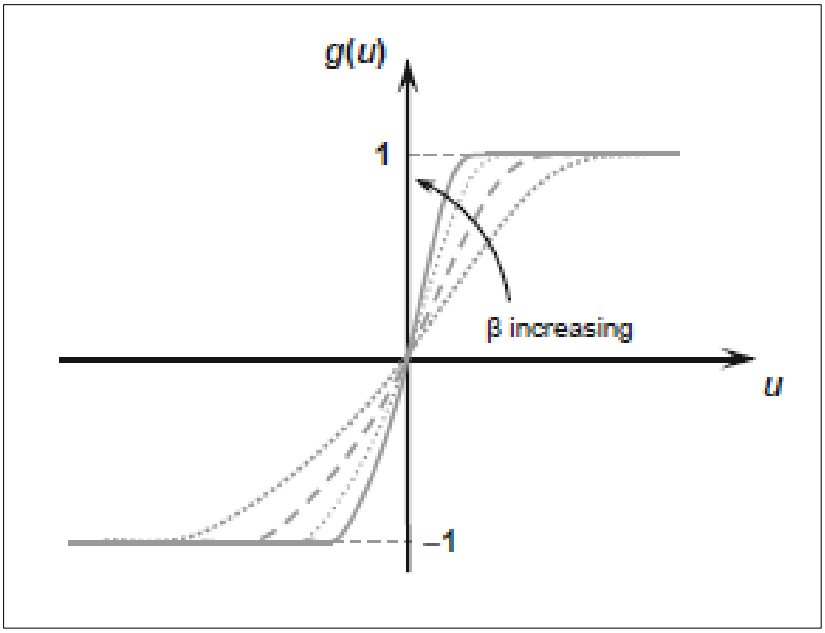
Grafica de función de activación tangente hiperbólica.



Nota: Fuente (Andrade et. al, 2017)

Figura 10

Grafica de la función de activación tangente hiperbólica cuando el parámetro β cambia.



Nota: Fuente (Andrade et. al, 2017)

Como se observa en la figura 10, cuanto mayor sea el valor de β, mayor será la pendiente de la función tangente hiperbólica, como en el caso de la función logística se aproximará a la función de paso bipolar (señal) cuando el valor de β es muy alto.

Es importante señalar que tanto las funciones tangentes logísticas como las hiperbólicas pertenecen a una familia de funciones llamadas sigmoidales.

c. Función de activación gaussiana

Para el caso de las funciones de activación gaussiana, la salida de la neurona producirá resultados iguales para aquellos valores de potencial de activación {u} colocados en la misma

distancia desde su centro (promedio). La curva es simétrica a este centro y la función gaussiana viene dada por:

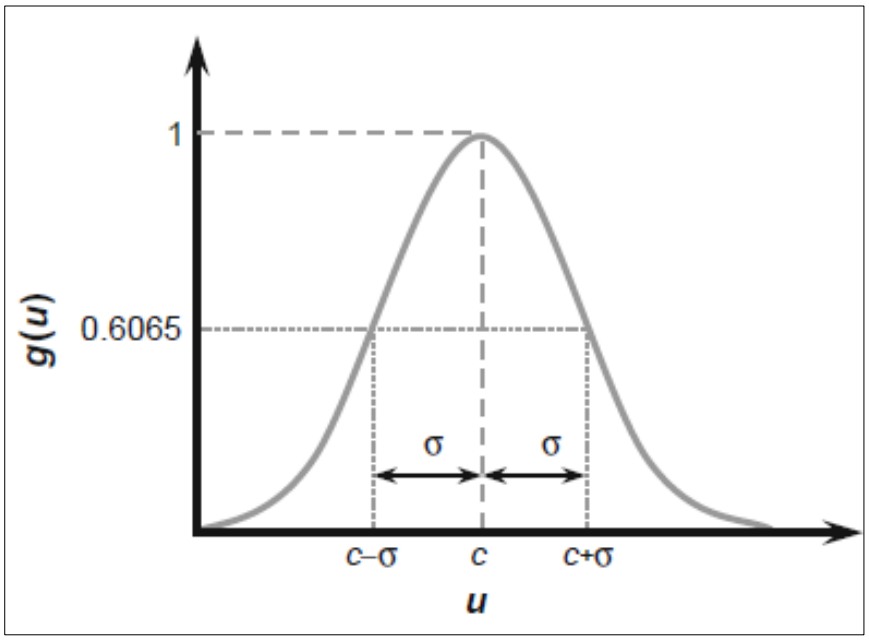
$$g(u) = e^{-\frac{(u-c)^2}{2\sigma^2}}$$

donde c es el parámetro que define el centro de la función Gaussiana y σ denota la

desviación estándar asociada, es decir, qué tan disperso está la curva con respecto a su centro.

Figura 11

Grafica de la función de activación gaussiana.



Nota: Fuente (Andrade et. al, 2017)

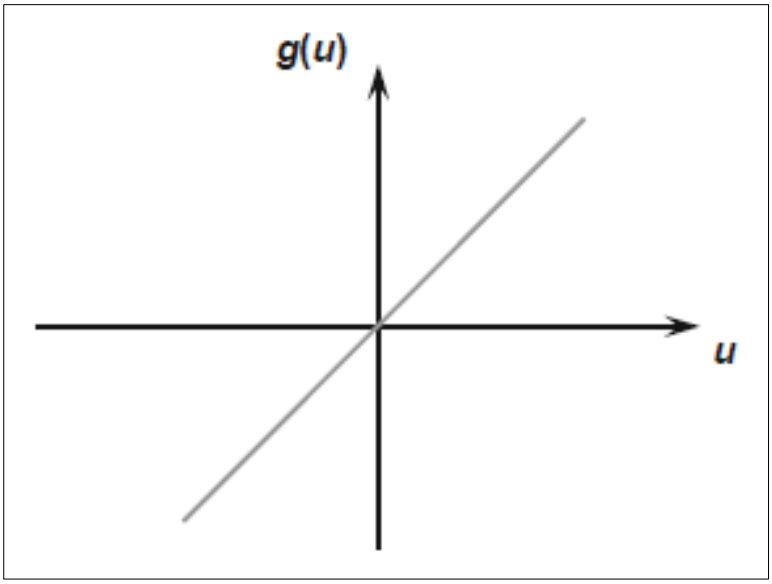
Es posible observar en esta figura que el parámetro de desviación estándar { σ } está directamente asociado con los puntos de inflexión de la función gaussiana, con σ^2 indicando su varianza.

d. Función de activación lineal

La función de activación lineal, o función de identidad, produce resultados de salida iguales al potencial de activación {u}, cuya expresión matemática viene dada por:

$$g(u) = u$$

Figura 12
Grafica de la función de activación lineal.



Nota: Fuente (Andrade et. al, 2017)

Una aplicación de las funciones de activación lineal es en redes neuronales artificiales realizando un ajuste de curva universal (aproximación de función), para mapear su comportamiento con las variables de entrada/salida del proceso.

2.2.13. ALGORITMO DE BACKPROPAGATION

Según Ertel (2017), con el algoritmo de backpropagation, presentamos ahora el modelo neuronal más utilizado. La razón de su uso generalizado es su versatilidad universal para tareas de aproximación. El algoritmo se origina directamente en la regla delta incremental. A diferencia de la regla delta, aplica una función sigmoidea no lineal sobre la suma ponderada de las entradas como función de activación.

Además, una red de retro propagación o backpropation puede tener más de dos capas de neuronas. En la Figura 13 se visualiza una red típica de retro propagación con una capa de entrada, una capa oculta y una capa de salida.

Dado que el valor de salida actual x_j^p de la neurona de la capa de salida, se compara con el valor de salida objetivo t_j^p, éstos se dibujan paralelos entre sí.

Aparte de las neuronas de entrada, todas las neuronas calculan su valor actual x_j mediante la regla:

$$x_j = f\left(\sum_{i=1}^{n} w_{ji} x_i\right)$$

donde n es el número de neuronas.

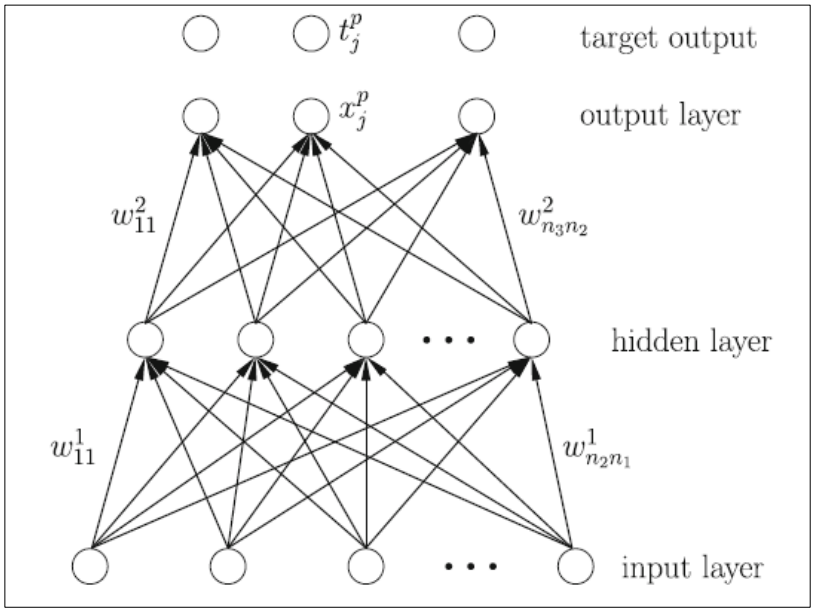
Utilizamos la función sigmoidea función:

$$f(X) = \frac{1}{1 + e^{-x}}$$

De forma análoga la regla delta incremental, los pesos también se modifican de forma proporcional a la gradiente negativa de la función de error cuadrática sumada sobre las neuronas de salida.

$$E_p(\boldsymbol{w}) = \frac{1}{2} \sum_{k \in output} (t_k^p - x_k^p)^2$$

Figura 13
Una red de tres capas de retro propagación.



Nota: Fuente (Ertel, 2017)

para el patrón de entrenamiento p:

$$\triangle_p \omega_{ji} = -\eta \frac{\partial E_p}{\partial \omega_{ji}}$$

Para derivar la regla de aprendizaje, la expresión anterior se sustituye por E_p. Dentro de la expresión, xk se sustituye por función de activación xj. Dentro de la ecuación, los output xj de las neuronas de la siguiente capa más profunda se producen recursivamente, etc. Mediante múltiples aplicaciones de la regla de la cadena obtenemos la regla de aprendizaje de retro propagación.

$$\triangle_p \omega_{ji} = \eta \delta_j^p x_j^p$$

Donde

$$\delta_j^p = \begin{cases} x_j^p \left(1 - x_j^p\right)\left(t_j^p - x_j^p\right) & si\ j\ es\ una\ neurona\ de\ salida \\ x_j^p \left(1 - x_j^p\right) \sum_k \delta_k^p \omega_{kj} & si\ j\ es\ una\ neurona\ oculta \end{cases}$$

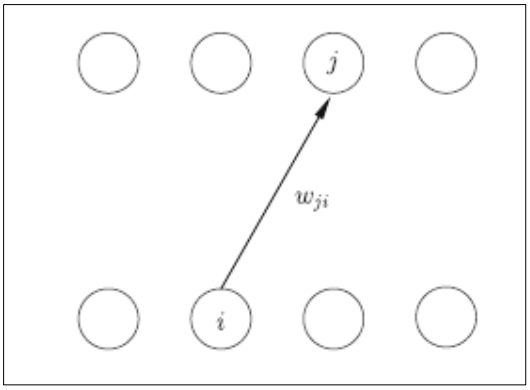
También se denota como regla delta generalizada, para todas las neuronas, la fórmula debe cambiar el peso wji de la neurona i a la neurona j contiene, como la regla de Hebb, un término $nx^p{}_i x^p{}_j$ .El nuevo factor (1 – x_j^p) crea la simetría que falta en la regla de Hebb, entre las activaciones 0 y 1 de la neurona j.

Para las neuronas de salida, el factor ($t_j^p - x_j^p$) se encarga de un cambio de peso proporcional al error.

Para las neuronas ocultas, el valor S_j^p de la neurona j se calcula recursivamente a partir de todos los cambios S_k^p de las neuronas del nivel inmediatamente superior.

Figura 14

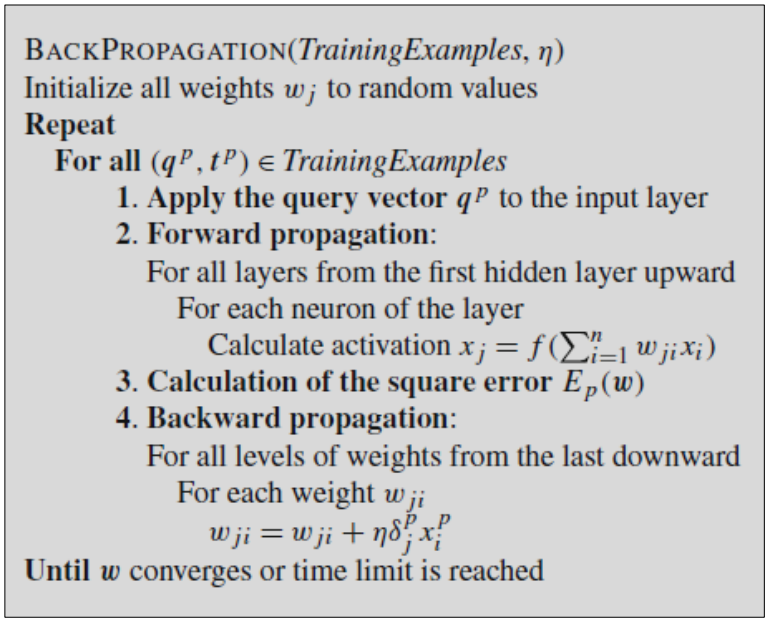
Designación de las neuronas y pesos para la aplicación de la regla de retro propagación.



Nota: Fuente (Ertel, 2017)

Después de calcular la salida de la red (propagación hacia adelante) para un ejemplo de entrenamiento, se calcula el error de aproximación. Esto se utiliza durante la propagación hacia atrás para alterar los pesos hacia atrás de una capa a otra.

Figura 15
Algoritmo de backpropagation.



Nota: Fuente (Ertel, 2017)

Todo el proceso se aplica a todos los ejemplos de entrenamiento y se repite hasta que los pesos dejan de cambiar o se alcanza un límite de tiempo. Si construimos la red con una capa oculta, se pueden aprender mapeos no lineales.

Sin capas ocultas, las neuronas de salida no son más potentes que una neurona lineal, a pesar de la función sigmoidea. La razón es que la función sigmoidea es estrictamente monótona.

Lo mismo ocurre con las redes multicapa que sólo utilizan una función lineal como función de activación, por ejemplo, la función de identidad. Esto se debe a que las ejecuciones encadenadas de los mapeos lineales son lineales en conjunto.

2.2.14. ARQUITECTURA DE MULTIPLES CAPAS

Para Andrade et. al (2017), las redes con múltiples capas se componen de una o más capas neuronales ocultas. Se emplean en la solución de diversos problemas, como los relacionados a la aproximación de funciones, clasificación de patrones, identificación de sistemas, proceso control, optimización, robótica, etc.

La Figura 15 muestra una red feedforward con múltiples capas, compuestas de una capa de entrada con n señales de muestra, dos capas neuronales ocultas que consisten en n1 y

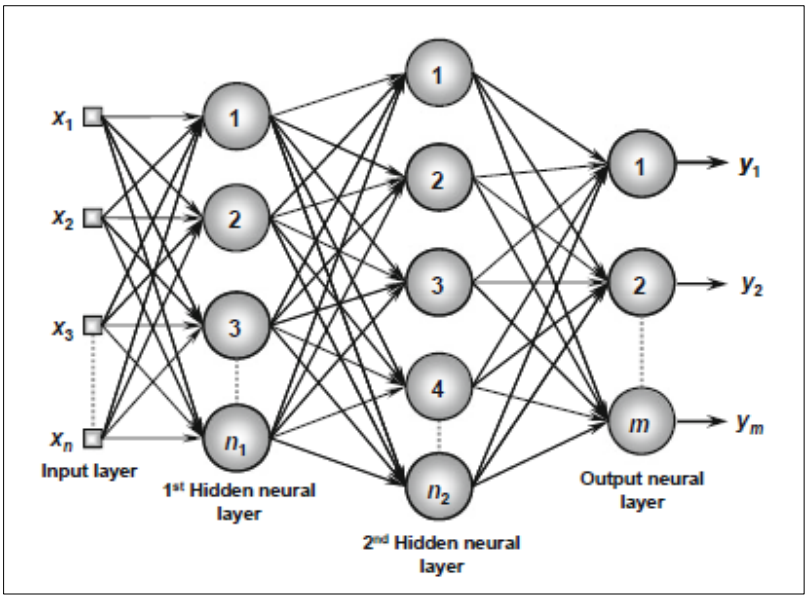
n2 neuronas respectivamente y, finalmente, una capa neuronal de salida compuesta por m neuronas que representan los respectivos valores de salida del problema que se analiza.

Entre las principales redes que utilizan arquitecturas feedforward de múltiples capas se encuentran las Perceptrón Multicapa (MLP) y la Función de Base Radial (RBF), cuyo aprendizaje algoritmos utilizados en sus procesos de formación se basan respectivamente en la regla delta y la regla competitiva/delta.

De la figura 16, es posible entender que la cantidad de neuronas que componen la primera capa oculta suele ser diferente del número de señales que componen la capa de entrada de la red. De hecho, el número de capas ocultas y sus respectivas neuronas depende de la naturaleza y la complejidad del problema que se está tratando, mapeados por la red, así como la cantidad y calidad de los datos disponibles sobre el problema.

No obstante, al igual que para las redes feedforward de capa simple, la cantidad de señales de salida siempre coincidirá con el número de neuronas de esa capa respectiva.

Figura 16
Ejemplo de red neuronal con arquitectura multicapas.



Nota: Fuente (Andrade et. al, 2017)

2.2.15. THE LOSS FUNCTION

Según Moolayil (2019), la función perdida es la métrica que ayuda a una red a comprender como está aprendiendo, si está en la dirección correcta o no. Para enmarcar la función de pérdida en simples palabras, considéralo como la puntuación que obtienes en un examen, tu apreciación para varias pruebas sobre el mismo tema: ¿qué métrica usaría para entender su desempeño en cada prueba? Obviamente, la puntuación de la prueba.

Se obtuvo 56, 60, 78, 90 y 96 de 100 en cinco pruebas de idioma consecutivas. Se ve claramente que el mejor de los puntajes de las pruebas es una indicación de que estás actuando bien.

Si los puntajes de las pruebas hubieran estado disminuyendo, entones el veredicto será que tu rendimiento disminuye y se tendría que cambiar de método de estudio para mejorar.

Del mismo modo, ¿cómo entiende una red si está mejorando su proceso de aprendizaje en cada iteración? utiliza la función de pérdida, que es análoga a la puntuación de la prueba. La función de pérdida mide esencialmente la pérdida del objetivo.

Supongamos que está desarrollando un modelo para predecir si un el estudiante aprobará o reprobará y la posibilidad de aprobar o reprobar se define por la probabilidad. Entonces, 1 indicaría que aprobará con 100% de certeza y 0 indicaría que definitivamente fallará. El modelo aprende de los datos y predice una puntuación de 0,87 para el estudiante para aprobar.

Entonces, la pérdida real aquí sería 1.00 – 0.87 = 0.13. Si se repite el ejercicio con algunas actualizaciones de parámetros para mejorar y ahora logra una pérdida de 0.40, entendería que los cambios que tiene hechos no están ayudando a la red a aprender apropiadamente.

Alternativamente, una nueva pérdida de 0.05 indicaría que las actualizaciones o cambios del aprendizaje van en la dirección correcta. Según el tipo de resultado de los datos, tenemos varias pérdidas estándar funciones definidas en ML y DL. Para casos de uso de regresión (es decir, donde la predicción final sería un número continuo como las calificaciones obtenidas por un estudiante, la cantidad de unidades de producto vendidas por una tienda, la cantidad de llamadas recibido de los clientes en un centro de contacto, etc.), aquí hay algunas populares funciones de pérdida disponibles:

- Error cuadrático medio: diferencia cuadrática media entre el valor real y el previsto. el cuadrado diferencia hace que sea fácil penalizar más al modelo por una diferencia mayor. Entonces, una diferencia de 3 resultaría en una pérdida de 9, pero la diferencia de 9 arrojaría una pérdida de 81. El equivalente matemático sería

$$\sum_{n=1}^{k} \frac{(Actual - Predicted)^2}{k}$$

- Error absoluto medio: el error absoluto medio entre lo real y lo previsto. El equivalente matemático sería

$$\sum_{n=1}^{k} |Actual - Predicted|$$

Para resultados categóricos, su predicción sería para una clase, como si un estudiante aprobará (1) o reprobará (0), si el cliente hará una compra o no, si el cliente incumplirá el pago o no, y así. Algunos casos de uso pueden tener varias clases como resultado, como clasificar tipos de enfermedades (Tipo A, B o C), clasificar imágenes como gatos, perros, coches, caballos, paisajes, etc. En tales casos, las pérdidas definidas en el párrafo anterior no podrán ser utilizadas por razones obvias.

Tendríamos que cuantificar el resultado de la clase como probabilidad y definir las pérdidas basadas en las estimaciones de predicciones. Algunas opciones populares para pérdidas por resultados categóricos en Keras son como sigue:

- **Entropía cruzada binaria o Binary cross-entropy**: Define la pérdida cuando el resultado categórico es una variable binaria, es decir, con dos resultados posibles: (Pasa/Reprueba) o (Sí/No). La forma matemática sería

$$Loss = [y * \log(p)] + (1 - y) * \log(1 - p)$$

- **Entropía cruzada categórica o Categorical cross-entropy**: Define la pérdida cuando los resultados categóricos no son binarios, es decir, mayores a 2 posibles resultados: (Sí/No/Tal vez) o (Tipo 1/ Tipo 2/… Tipo n). La forma matemática sería

$$Loss = -\sum_{i}^{n} Y_i \log_2 Y_i$$

2.2.16. OPTIMIZADORES

Para Moolayil (2019), la parte más importante del entrenamiento del modelo es el optimizador. Para agregar más contexto, imagine la estructura del modelo que tiene definido para clasificar, si un estudiante aprobará o reprobará, la estructura creada al definir la secuencia de capas con el número de neuronas, las funciones de activación, y la forma de entrada y salida se inicializa con pesos aleatorios al principio.

Los pesos que determinaron se actualizara, la influencia de una neurona en la siguiente neurona o la salida final durante el proceso de aprendizaje por la red. En pocas palabras, una red con pesos aleatorios y una estructura no definida es el punto de partida de un modelo.

El modelo puede hacer una predicción en este punto, pero casi siempre carecería de valor. La red toma una muestra de entrenamiento y usa sus valores como entradas para las neuronas en la primera capa, que luego produce una salida con la función de activación. La salida ahora se convierte en una entrada para la siguiente capa, y así. La salida de la capa final sería la predicción para la muestra de entrenamiento. Aquí es donde la función de pérdida entra en escena.

La función de pérdida ayuda a la red a comprender qué tan bien o mal está la corriente. conjunto de pesos se ha realizado en la muestra de entrenamiento. El siguiente paso para el modelo es reducir la pérdida. La función del optimizador le ayuda a entender este paso, es una función matemática, un algoritmo que utiliza derivadas, derivadas parciales y la regla de la cadena en cálculo para comprender cuánto cambio verá la red en la función de pérdida al hacer un pequeño cambio en el peso de las neuronas. Los cambios en la función de pérdida, que sería un aumento o disminución, ayuda a determinar la dirección del cambio requerido en el peso de la conexión.

El cálculo de una muestra de entrenamiento de la capa de entrada a la capa de salida se llama pase. Por lo general, el entrenamiento se haría en lotes debido a limitaciones de memoria en el sistema. Un lote es una colección. de muestras de entrenamiento de toda la entrada. La red actualiza sus pesos después de procesar todas las muestras en un lote. Esto se llama una iteración (es decir, un paso exitoso de todas las muestras en un lote seguido de una actualización de peso en la red). El cálculo de todas las muestras de entrenamiento proporcionadas en los datos de entrada con actualizaciones de peso lote por lote se denomina época.

En cada iteración, la red aprovecha la función del optimizador para hacer un pequeño cambio en sus parámetros de peso (que se inicializaron aleatoriamente al principio) para mejorar la predicción final mediante la reducción de la función de pérdida. Paso a paso, con varias iteraciones y luego varias épocas, la red actualiza su pesa y aprende a hacer una predicción correcta para el entrenamiento dado muestras.

2.2.17. PROCESOS DE ENTRENAMIENTO Y PROPIEDADES DEL APRENDIZAJE

Para Andrade et. al (2017), las características más relevantes en las redes neuronales artificiales es la capacidad de aprendizaje a partir de la presentación de muestras (patrones), que expresa el sistema comportamiento. Por lo tanto, después de que la red haya aprendido la relación entre las entradas y salidas, puede generalizar soluciones, lo que significa que la red puede producir una salida que está cerca de la salida esperada (o deseada) de cualquier valor de entrada dado.

Por tanto, el proceso para entrenar una red neuronal consiste en aplicar los pasos ordenados requeridos para sintonizar los pesos sinápticos y los umbrales de las neuronas, con el fin de generalizar las soluciones producidas por sus salidas. El conjunto de pasos ordenados utilizados para entrenar la red se llama algoritmo de aprendizaje. Durante su ejecución, la red podrá así extraer discriminantes características sobre el sistema que se mapea a partir de muestras adquiridas del sistema. Por lo general, el conjunto completo que contiene todas las muestras disponibles del comportamiento del sistema se divide en dos subconjuntos, que se denominan subconjunto de prueba y subconjunto de entrenamiento.

Los subconjuntos de entrenamiento, compuesto por 60–90 % de muestras aleatorias del conjunto completo, se utilizará fundamentalmente en el proceso de aprendizaje.

Por otro lado, el subconjunto de prueba, que se compone del 10 al 40 % del conjunto completo de la muestra, se utilizará para verificar si las capacidades de la red para generalizar soluciones están dentro de niveles aceptables, permitiendo así la validación de una topología dada. No obstante, al dimensionar estos subconjuntos, también se deben considerar las características estadísticas de los datos.

Durante el proceso de entrenamiento, cada iteración completa de las muestras pertenecientes a los datos de entrenamiento, con el fin de ajustar la pesos y umbrales sinápticos, se denominará época de entrenamiento.

2.2.17.1. Aprendizaje supervisado

Para Andrade et. al (2017), la estrategia de aprendizaje supervisado consiste en tener disponibles los resultados deseados para un conjunto de datos de entrada; en otras palabras, cada muestra de entrenamiento se compone de las señales de entrada y sus correspondientes salidas.

En adelante, se requiere una mesa con datos de entrada/salida, también llamada tabla de atributos/valores, que representa el proceso y su comportamiento. Es a partir de esta información que las estructuras neuronales formularán “hipótesis” sobre el sistema que se está aprendiendo.

En este caso, la aplicación del aprendizaje supervisado solo depende de la disponibilidad de esa tabla de atributos/valores, y se comporta como si un “entrenador” estuviera enseñando la red.

Los pesos y umbrales sinápticos de la red se ajustan continuamente mediante la aplicación de acciones comparativas, ejecutadas por el algoritmo de aprendizaje misma, que supervisa la discrepancia entre las salidas producidas con respecto a las salidas deseadas, utilizando esta diferencia en el procedimiento de ajuste, la red se considera “entrenado” cuando esta discrepancia está dentro de un rango de valores aceptables, teniendo en cuenta los propósitos de generalizar soluciones.

De hecho, el aprendizaje supervisado es un caso típico de inferencia inductiva pura, donde las variables libres de la red se ajustan al conociendo a priori, la deseada salida para el sistema investigado.

Para Qamar y Summair (2020), los algoritmos de machine learning se basan hoy en día en técnicas supervisadas. Aunque se trata de un dominio completo que requiere una discusión, se trata de un campo completo que requiere una discusión profunda por separado, aquí sólo proporcionaremos una breve visión general del tema.

En el aprendizaje automático supervisado, el programa ya conoce la salida. Tenga en cuenta que esto es lo contrario a la programación convencional, en la que alimentamos la entrada al programa y el programa da la salida. En este caso, damos la entrada y la salida al mismo tiempo para hacer que el programa aprenda que en caso de cualquiera de estas entradas relacionadas, lo que el programa tiene que salir.

Este proceso de aprendizaje se llama construcción de modelos. Significa que, a través de la entrada y la salida, el sistema tendrá que construir un modelo que mapee el proceso, para que la próxima vez que se le dé la entrada al sistema, el sistema proporcione la salida utilizando este modelo. En términos matemáticos, la tarea del algoritmo de aprendizaje automático es encontrar el valor de la variable dependiente utilizando el modelo con las variables independientes. Cuanto más preciso sea el modelo, más eficiente será el algoritmo, y más eficientes son las decisiones basadas en este modelo.

Las variables dependientes e independientes se proporcionan al algoritmo mediante un conjunto de datos de entrenamiento. Dos técnicas importantes utilizados por el aprendizaje automático supervisado son:

CLASIFICACIÓN

La clasificación es una de las principales técnicas de aprendizaje automático que utilizan el aprendizaje supervisado. Clasificamos los datos desconocidos utilizando el aprendizaje supervisado, por ejemplo, podemos clasificar a los estudiantes de una clase en hombres y mujeres, podemos clasificar el correo electrónico en dos clases como spam y no-spam, etc.

REGRESIÓN

En la regresión, intentamos encontrar el valor de las variables dependientes a partir de las variables independientes a partir de los datos ya proporcionados; sin embargo, una diferencia básica, es que los datos proporcionados aquí están en forma de valores reales.

2.2.17.2. Aprendizaje sin supervision

Para Andrade et. al (2017), a diferencia del aprendizaje supervisado, la aplicación de un algoritmo basado en el aprendizaje no supervisado no requiere ningún conocimiento de las salidas deseadas. Así, la red necesita organizarse cuando existen particularidades entre los elementos que componen todo el conjunto muestral, identificando subconjuntos (o clusters) que presentan similitudes.

El algoritmo de aprendizaje ajusta los pesos sinápticos y umbrales de la red para reflejar estos grupos dentro de la red sí mismo. Alternativamente, el diseñador de la red puede especificar (a priori) la cantidad máxima de estos posibles clusters, utilizando su conocimiento sobre el problema.

2.2.17.3. Aprendizaje reforzado

Según Sutton y Barto (1998), los métodos basados en el aprendizaje por refuerzo se consideran una variación del aprendizaje supervisado, ya que analizan continuamente la diferencia entre la respuesta que produce la red y la correspondiente salida deseada.

Los algoritmos de aprendizaje utilizados en el aprendizaje por refuerzo ajustan los parámetros neuronales internos que se basan en cualquier información cualitativa o cuantitativa adquirido a través de la interacción con el sistema (entorno) que se está mapeando, utilizando esta información para evaluar el rendimiento del aprendizaje.

El proceso de aprendizaje de la red generalmente se realiza por ensayo y error porque la única respuesta disponible para una entrada dada es si fue satisfactoria o insatisfactoria. Si es satisfactorio, los pesos y umbrales sinápticos se incrementan gradualmente hasta reforzar (recompensar) esta condición de comportamiento involucrada con el sistema. Varios algoritmos de aprendizaje utilizados por el aprendizaje por refuerzo se basan en métodos estocásticos que seleccionan probabilísticamente las acciones de ajuste, considerando un conjunto finito de posibles soluciones que pueden ser recompensadas si tienen posibilidades de generar resultados satisfactorios.

2.2.17.4. Offline learning

Para Reed y Marks II (1999), en el aprendizaje fuera de línea, también llamado aprendizaje por lotes, los ajustes en los vectores de peso y los umbrales de la red se realizan después de que se presenta todo el conjunto de entrenamiento, ya que cada paso de ajuste tiene en cuenta el número de errores observados dentro las muestras de entrenamiento con respecto a los valores deseados para sus salidas.

Por lo tanto, las redes que utilizan el aprendizaje fuera de línea requieren, al menos, una época de entrenamiento para ejecutar un paso de ajuste en sus pesos y umbrales. Por lo tanto, todas las muestras de entrenamiento deben estar disponibles durante todo el proceso de aprendizaje.

2.2.17.5. Online learning

Para Reed y Marks II (1999) es un aprendizaje opuesto al aprendizaje fuera de línea, el aprendizaje en línea, los ajustes en los pesos y los umbrales se realizan después de presentar cada muestra de entrenamiento.

Así, después de ejecutar el paso de ajuste, la muestra respectiva puede ser descartada. El aprendizaje en línea con esta configuración se suele utilizar cuando el comportamiento del sistema que se está mapeando cambia rápidamente, por lo que la adopción del aprendizaje fuera de línea es casi impráctico porque las muestras utilizadas en un momento dado pueden no más representar el comportamiento del sistema en momentos posteriores.

Sin embargo, dado que los patrones se presentan de uno en uno, el peso y el umbral las acciones de ajuste están bien ubicadas y son puntuales, y reflejan un determinado comportamiento circunstancia del sistema. Por lo tanto, la red comenzará a proporcionar información precisa respuestas después de presentar un número significativo de muestras.

2.2.18. CAPAS DE SALIDA SOFTMAX

Según Buduma (2015), muchas veces, queremos que nuestro vector de salida sea una distribución de probabilidad sobre un conjunto de etiquetas mutuamente excluyentes. Por ejemplo, supongamos que queremos construir una red neuronal para reconocer dígitos escritos a mano del conjunto de datos MNIST. Cada etiqueta (0 a 9) es mutuamente excluyentes, pero es poco probable que podamos reconocer dígitos con 100% confianza. El uso de una distribución de probabilidad nos da una mejor idea de la confianza estamos

en nuestras predicciones. Como resultado, el vector de salida deseado tiene la siguiente forma, donde:

$$\sum_{i=0}^{9} p_i = 1$$

$[P_0\, P_1\, P_2\, P_3 \ldots P_9]$

Esto se logra mediante el uso de una capa de salida especial llamada capa softmax. A diferencia de otros tipos de capas, la salida de una neurona en una capa softmax depende de las salidas de todas las otras neuronas en su capa. Esto se debe a que requerimos la suma de todas las salidas. Sea igual a 1. Si Zi es el logit de la i-ésima neurona softmax, podemos lograr esta normalización estableciendo su salida en:

$$y_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

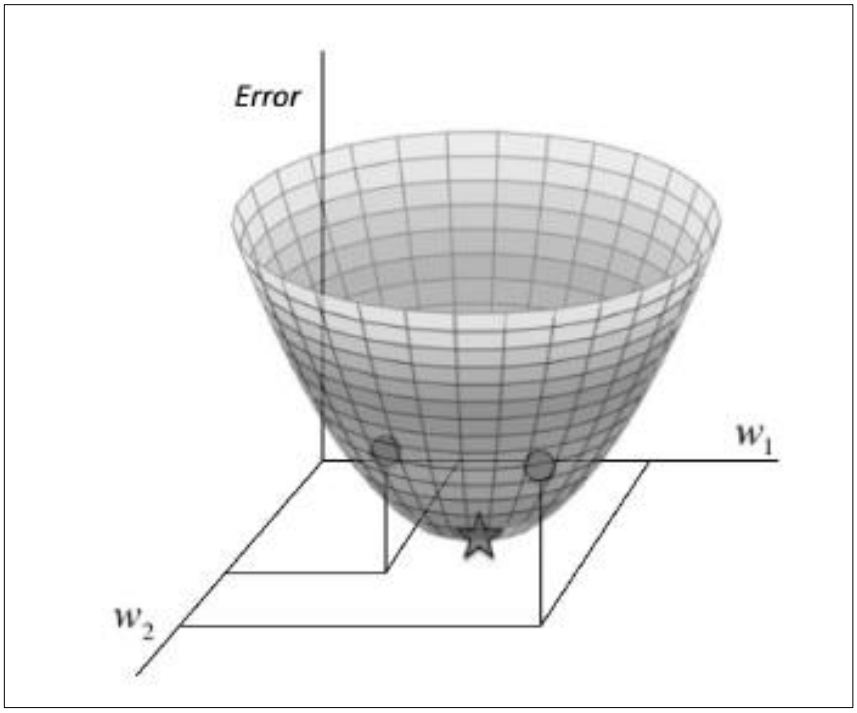
Una predicción fuerte tendría una sola entrada en el vector cercana a 1 mientras que las entradas restantes estaban cerca de 0. Una predicción débil tendría múltiples posibles etiquetas que son más o menos igualmente probables.

2.2.19. GRADIENTE DESCENDIENTE

Buduma (2015), visualicemos cómo podemos minimizar el error cuadrático en todo el entrenamiento, simplificando el problema. Digamos que nuestra neurona lineal solo tiene dos entradas (y por lo tanto solo dos pesos, W1 y W2). Entonces podemos imaginar un espacio tridimensional donde las dimensiones horizontales corresponden a los pesos W1 y W2, y las verticales dimensión corresponde al valor de la función de error E.

En este espacio, los puntos en el plano horizontal responden a diferentes ajustes de los pesos y la altura, en esos puntos corresponde al error incurrido. Si consideramos los errores que cometemos sobre todos los pesos posibles, obtenemos una superficie en este espacio tridimensional, en particular, un tazón cuadrático como se ve en la figura 17.

Figura 17
La superficie de error cuadrático para una neurona lineal.

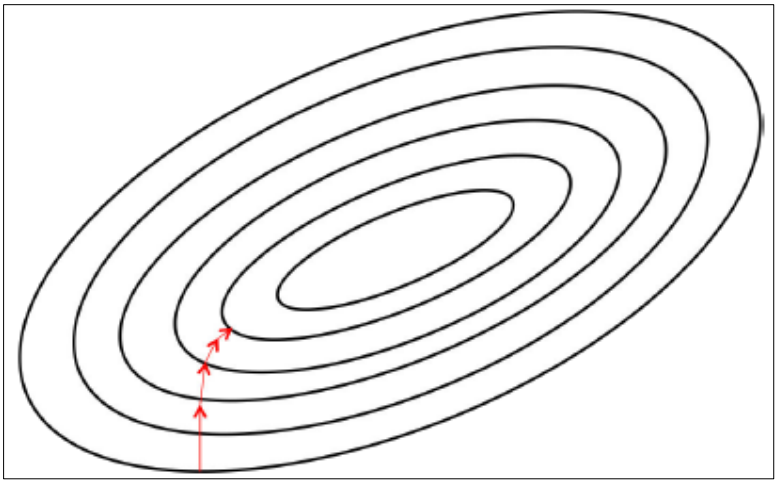


Nota: Fuente (Buduma, 2015)

También podemos visualizar convenientemente esta superficie como un conjunto de contornos elípticos, donde el error mínimo está en el centro de las elipses. En esta configuración, estamos trabajando en un plano dimensional, donde las dimensiones corresponden a los dos pesos.

Los contornos corresponden a los ajustes de W1 y W2 que se evalúan al mismo valor de E. Cuanto más cerca las curvas de nivel son entre sí, cuanto más empinada es la pendiente. De hecho, resulta que la dirección del descenso más empinado es siempre perpendicular a los contornos, esta dirección es expresado como un vector conocido como el gradiente.

Figura 18
Visualización de la superficie de error como un conjunto de contornos.



Nota: Fuente (Buduma, 2015)

Ahora podemos desarrollar una estrategia de alto nivel sobre cómo encontrar el valor de los pesos que minimizan la función de error. Supongamos que inicializamos aleatoriamente los pesos de nuestra red, por lo que nos encontramos en algún lugar en el plano horizontal.

Al evaluar el gradiente en nuestra posición actual, podemos encontrar la dirección de descenso más empinado y podemos puede dar un paso en esa dirección. Entonces nos encontraremos en una nueva posición que es más cerca del mínimo de lo que estábamos antes. Podemos volver a evaluar la dirección de descenso más empinado tomando la pendiente en esta nueva posición y dando un paso en esta nueva dirección. Es fácil ver que, como se muestra en la Figura 18, seguir esta estrategia eventualmente nos llevara al punto de mínimo error.

Este algoritmo se conoce como gradiente descendiente, y lo usaremos para abordar el problema de entrenar neuronas individuales y el desafío más general de entrenar redes enteras.

2.2.20. LEARNING RATE O RANGO DE APRENDIZAJE

Para Buduma (2015), entrenar nuestra neurona necesita de los hiperparámetros. Además de los parámetros de peso definidos en nuestra red neuronal, los algoritmos de aprendizaje

también requieren un par de parámetros adicionales para llevar a cabo el proceso de entrenamiento. Uno de estos llamados hiperparámetros es el rango de aprendizaje.
En la práctica, en cada paso de movimiento perpendicular al contorno, tenemos que determinar la distancia que queremos recorrer antes de recalcular nuestra nueva dirección. Esta distancia debe depender de la inclinación de la superficie. Cuanto más cerca estemos del mínimo más corto será el paso que queramos dar. Sabemos que estamos cerca del mínimo porque la superficie es mucho más plana, por lo que podemos utilizar la inclinación como un indicador de cómo de lo cerca que estamos del mínimo. Sin embargo, si nuestra superficie de error es bastante suave, el entrenamiento puede requerir una gran cantidad de tiempo.

Como resultado, a menudo multiplicamos el gradiente por un factor, el rango de aprendizaje. Elegir el rango de aprendizaje es un problema difícil. Como acabamos de discutir, si elegimos una tasa de aprendizaje demasiado pequeña, corremos el riesgo de tardar demasiado durante el proceso de entrenamiento. Pero si elegimos una tasa de aprendizaje demasiado grande, lo más probable es que empecemos a divergir del mínimo, es por eso que para su análisis se utilizara optimizadores que nos ayudaran con el resultado.

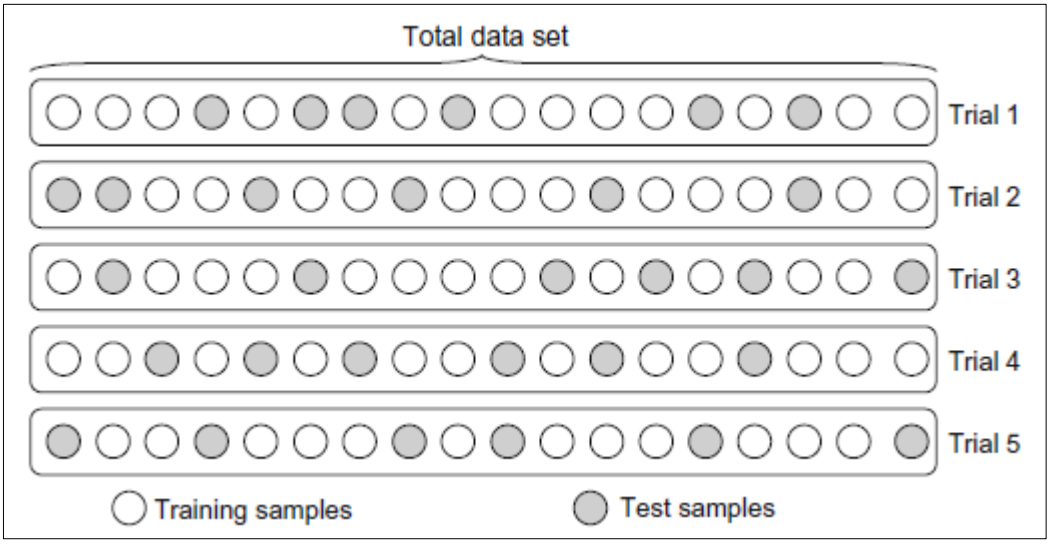
2.2.21. MÉTODOS DE VALIDACIÓN CRUZADA

Para Andrade et. al (2017), el primer método, se llama validación cruzada de sub muestreo aleatorio, donde el conjunto de datos (muestras) disponible se dividirán aleatoriamente en dos subconjuntos: entrenamiento y prueba (validación). Más específicamente, el subconjunto de entrenamiento, se utilizará para entrenar todas las topologías candidatas, mientras que el subconjunto de prueba solo se aplica para seleccionar la topología que proporciona mejores resultados de generalización
De hecho, como las muestras del subconjunto de prueba no participan en el proceso de aprendizaje, es posible evaluar el rendimiento de generalización de cada topología candidata comparando los resultados producido por sus salidas con los valores deseados al usar el subconjunto de prueba.

En términos prácticos, del conjunto total de datos disponibles, alrededor del 60-90 % de los elementos se eligen al azar para el subconjunto de entrenamiento. Este sistema de partición debe repetirse varias veces durante el proceso de aprendizaje de las topologías candidatas

para proporcionar (para cada ensayo) diferentes muestras en ambos subconjuntos.
El rendimiento global de cada topología candidata se obtendrá a partir de la media de las actuaciones en cada prueba.

Figura 19
Método de validación cruzada de submuestreo aleatorio.

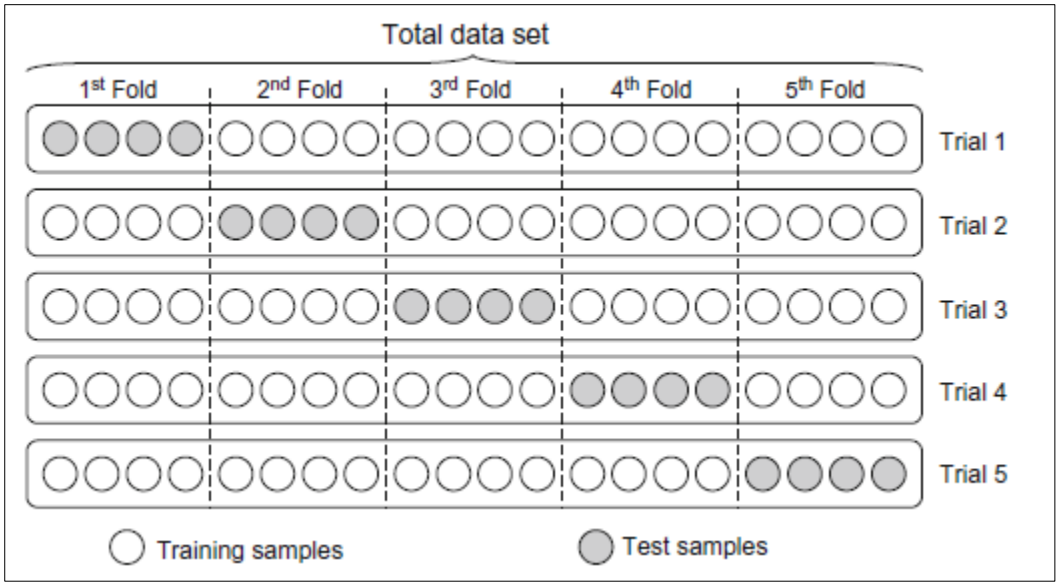


Nota: Fuente (Andrade et. al, 2017)

La figura 19 ilustra la estrategia para seleccionar muestras para el entrenamiento y la prueba, subconjuntos para cada uno de los cinco ensayos, seis de las dieciocho muestras disponibles serán parte del subconjunto de prueba, mientras que los doce restantes formarán parte del subconjunto de entrenamiento. En cada prueba, las muestras del subconjunto de prueba se elegirán al azar considerando todas las muestras disponibles (sin verificar si algunas de ellas ya han sido elegidas en ensayos anteriores o no).

El segundo método de validación cruzada utilizado para diseñar redes MLP es conocido como validación cruzada k-fold. Consiste en dividir el conjunto muestral total en k particiones, donde (k − 1) se utilizarán particiones para componer el subconjunto de entrenamiento y la partición restante se utilizará para componer el subconjunto de prueba. Después de esta división, el proceso de aprendizaje se repite k veces hasta que todas las particiones han sido utilizados como un subconjunto de prueba.

Figura 20
Método de validación cruzada de k-fold.



Nota: Fuente (Andrade et. al, 2017)

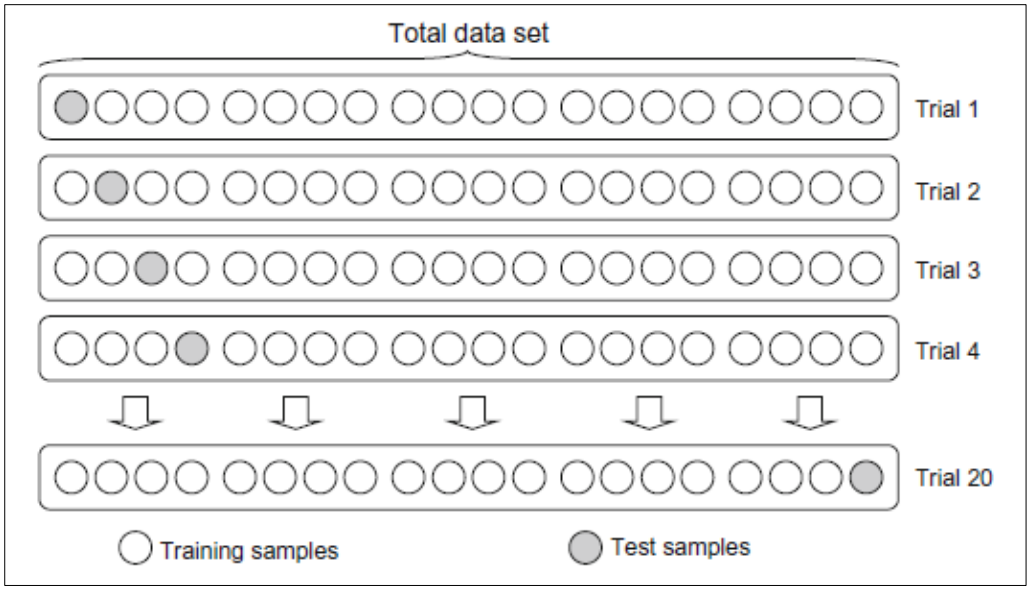
La Figura 20 ejemplifica la estrategia de este mecanismo para 20 muestras, con k = 5, lo que implica realizar 5 ensayos. El valor del parámetro k está ligado al número total de muestras disponibles, y suele definirse entre 5 y 10.

El desempeño global de cada candidato la topología se obtiene evaluando el promedio de los rendimientos individuales en cada uno de los k intentos.

Finalmente, el tercer método (y menos habitual) se llama dejar uno fuera, consiste en utilizar una sola muestra para el subconjunto de prueba, mientras que las otras muestras se utilizan en el subconjunto de entrenamiento.

El proceso de aprendizaje entonces se repite hasta que todas las muestras se utilizan individualmente como el subconjunto de prueba. Esta última técnica es un caso particular del método k-fold cuando el parámetro k asume el valor correspondiente al número total de muestras. Sin embargo, hay es un gran esfuerzo computacional en este caso, porque el proceso de aprendizaje se repetirá, para cada topología candidata, tantas veces como el tamaño de la topología completa conjunto de muestra, la figura 21 resume esta estrategia de validación cruzada para 20 muestras.

Figura 21
Método de validación cruzada uno fuera.



Nota: Fuente (Andrade et. al, 2017)

2.2.22. MATRIZ DE CONFUSIÓN

Según Beysolow (2017), un método para evaluar los modelos de clasificación es la matriz de confusión, una representación gráfica de las predicciones de los clasificadores contra las etiquetas reales de unas observaciones dadas.

A partir de esta visualización, derivamos los valores para las estadísticas. enumerados anteriormente que, en última instancia, nos ayudan a evaluar con precisión el modelo de clasificación actuación.

Figura 22
Matriz de confusión.

	P′ (Predicted)	n′ (Predicted)
P (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

Nota: Fuente (Beysolow, 2017)

La figura 22 muestra un ejemplo visual de una matriz de confusión. Interpretar los valores dentro de una matriz de confusión a menudo es una tarea subjetiva que es depende del lector determinar. En algunos casos, los falsos positivos, como determinar si los usuarios deben comprar un producto o no, no será tan perjudicial para resolver el problema en cuestión.

En otros casos, como determinar si el motor de un automóvil está defectuoso, los falsos positivos en realidad pueden ser perjudiciales. Los lectores deben ser conscientes de la tarea que están realizando y ajustar el modelo para limitar los falsos positivos y/o falsos negativos respectivamente.

2.2.23. REGRESIÓN LOGÍSTICA

Para Mehta y Patel (1995), existe dos clases de modelos: regresión logística para datos binarios no estratificados y logística de regresión para datos binarios estratificados. En esta sección discutimos un método uniforme de inferencia exacta para ambos modelos, con base en distribuciones permutacionales de estadísticos apropiados.

2.2.23.1. Regresión logística para datos binarios no estratificados

Considere un conjunto de variables aleatorias binarias independientes, Y1, Y2, … , Yn. correspondiente a cada variable aleatoria, Yj, donde un (p x 1) vector $X_{j=}\left(X_{1j},\ X_{2j},\ \ldots, X_{pj}\right)'$ de variables explicativas o covariables. Sea π_j la probabilidad de que Yj = 1. La regresión logística modela la dependencia de π_j en X_j a través de la relación.

$$\log\left(\frac{\pi_j}{1-\pi_j}\right) = \gamma + \chi_j{}'\beta$$

Donde γ y β = ($\beta_1, \beta_2, \ldots, \beta_p$) son parámetros desconocidos. La función de verosimilitud, o probabilidad de un conjunto observado de valores, $y_1, y_2, \ldots, y_n$ es:

$$\Pr(Y_1 = y_1, Y_2 = y_2\ , \ldots, Y_n = y_n) = \frac{exp\left[\sum_{y=1}^{n} y_j\left(X_j'\beta + \gamma\right)\right]}{\prod_{j=1}^{n}\left[1 + \exp\left(X_j'\beta + \gamma\right)\right]}$$

La forma habitual de hacer inferencias sobre γ y β es maximizar la ecuación anterior con respecto a estas regresiones.

Supongamos que nos interesan las inferencias sobre β y consideramos γ como un parámetro molesto. Después, en lugar de estimar γ a partir de la función de verosimilitud incondicional anterior, podemos eliminarla mediante condicionado al valor observado de su estadístico suficiente.

$$m = \sum_{j=1}^{n} y_j$$

Esto produce la función de verosimilitud condicional,

$$\Pr(Y_1 = y_1, Y_2 = y_2, \dots, Y_n = y_n | m) = \frac{exp\left(\sum_{y=1}^{n} y_j \, X_j' \beta\right)}{\sum_R \left(\exp \sum_{j=1}^{n} y_j \, X_j' \beta\right)}$$

donde la suma exterior en el denominador de la ecuación está sobre el conjunto

$$R = \left\{ (y_1, y_2, \dots, y_n) : \sum_{j=1}^{n} y_j = m \right\}$$

Ahora podemos abordar la inferencia sobre β de dos maneras: asintótica y exacta, un asintótico enfoque es maximizar la función de verosimilitud condicional de la ecuación Pr. Este es un caso especial del Método de Breslow y Day para el manejo de datos estratificados. Exacto la inferencia sobre β se basa en la distribución permutacional de sus estadísticas suficientes. Uno puede observar a partir de la forma de la ecuación de Pr que el vector (px1) de estadísticas suficientes para β es:

$$t = \sum_{j=1}^{n} y_j x_j$$

Y su distribución es:

$$\Pr\left(T_1 = t_1, T_2 = t_2, \dots, T_p = t_p\right) = \frac{c(t)\, e^{t\beta}}{\sum_u c(u)\, e^{u\beta}}$$

Donde

$$S(t) = \left\{ (y_1, y_2, \dots, y_n) : \sum_{j=1}^{n} y_j = m, \sum_{j=1}^{n} y_j x_{ij} = t_j, i = 1,2, \dots, p \right\},$$

|S| denota el número de elementos distintos en el conjunto S, y la suma en el denominador es sobre todo u para el cual c(u) > 1. En otras palabras, c(t) es la cuenta del número de secuencias binarias de la forma $(y_1, y_2, \ldots, y_n)$ tal que $\sum_j^1 y_j = m$ y $\sum_j^1 y_j x_{ij} = t_p$, donde i = 1,2, ... , p . Inferencia exacta donde β requiere el cálculo de coeficientes tales como c(t) en el que algunos de las estadísticas se fijan en sus valores observados y se requiere que otras varíen sobre sus valores permisibles de sus rangos.

2.2.23.2. Regresión logística para datos binarios estratificados

Supongamos que hay N estratos, con respuestas binarias en cada uno de ellos. Que el estrato i-ésimo tenga unas respuestas y n_i - m_i no tengan respuestas. Para todo 1 < i < N, y 1 < j < ni, sea Yij=1, si el j-ésimo respondió el individuo del i-ésimo estrato; 0 de lo contrario.

Definir π_{ij} = Pr(Yij | Xij) donde Xij es un vector p-dimensional de covariables para el j-ésimo individuo en el i-ésimo estrato. La regresión logística para π_{ij} es de la forma

$$\log\left(\frac{\pi_{ij}}{1-\pi_{ij}}\right) = \gamma_i + \chi_j{}'\beta$$

donde γ_i es un parámetro escalar específico del estrato y β es un vector (p x 1) de parámetros comunes en todos los estratos N. El interés habitual está en las inferencias sobre β, con los γ_i considerados como una molestia parámetros. Uno podría, por supuesto, estimar estos parámetros molestos por la máxima verosimilitud del método. Sin embargo, la teoría asintótica habitual de la estimación de máxima verosimilitud requiere que la dimensión del espacio de parámetros se fija a la medida que aumentara el número de observaciones.

Cox y Hinkley observan que cuando la dimensión del espacio de parámetros es grande comparable al número de observaciones, el MLE puede tener un sesgo serio, la estimación de la razón de probabilidades común a partir de datos de pares emparejados.

Un modelo de regresión logística para dichos datos contendría un conjunto de parámetros molestos específicos, uno para cada par coincidente, y un único parámetro de razón de probabilidades, común a todos los pares emparejados. Si uno estima los parámetros de

molestia específicos del estrato a partir de los datos, la estimación de la razón de probabilidades común se ha demostrado que converge al cuadrado de su verdadero valor.

En lugar de estimar todos los parámetros específicos del estrato, se popularizó un enfoque alternativo por Breslow y Day, es eliminar estos parámetros molestos condicionando en su estadística suficiente, el número de respuestas, mi, en cada estrato.

La condicional verosimilitud, o probabilidad condicional de observar Yij = yij, j = 1,2, ... , ni, i = 1,2, .. . , N es entonces:

$$\Pr(Y_1 = y_1, Y_2 = y_2\ , \dots, Y_n = y_n | m_1, m_2, \dots, m_N) = \frac{exp\left(\sum_{y=1}^{n} \sum_{j=1}^{n} y_j\ \left(X_j'\beta\right)\right)}{\sum_{i=1}^{N} \sum_{R} \left(\exp \sum_{i=1}^{N} \sum_{j=1}^{n} y_j\ X_j'\beta\right)}$$

donde las dos sumas externas en el denominador están sobre los conjuntos

$$R_i = \left\{ (y_1, y_2\ , \dots, y_n) : \sum_{j=1}^{n} y_j = m_i \right\}$$

para i = 1,2, ..., N. Observe que los parámetros molestos, γ_i, se han factorizado fuera de la anterior probabilidad condicional. El enfoque de Breslow y Day es hacer inferencias asintóticas sobre β maximizando la ecuación Pr. La inferencia exacta se basa en las estadísticas suficientes para β.De la ecuación Pr podemos ver que el vector de estadísticas suficientes para β es :

$$t = \sum_{i=1}^{n} \sum_{j=1}^{n} y_j x_{ij}$$

y su distribución condicional es:

$$\Pr\left(T_1 = t_1, T_2 = t_2\ , \dots, T_p = t_p\right) = \frac{c(t)\ e^{\beta t}}{\sum_u c(u)\ e^{\beta u}}$$

Donde

$$c(t) = |S_N(t)|$$

$$S_N(t) = \left\{ \left(y_{ij}, j = 1, \dots, n_{ij} = 1, .., N\right) : \sum_{i=1}^{n} \sum_{j=1}^{n} y_j x_{ij} = t, \sum_{j=1}^{n} y_j = m_i \right\}$$

$|S_N|$denota el número de elementos distintos en el conjunto S_N, y la sumatoria en el denominador, es sobre todo u para lo cual c(u) > 1. En otras palabras, c(t) es la cuenta del número de formas de seleccionar la secuencia binaria $\left(y_{ij}, i = 1, \dots, N, j = 1, .., n_i\right)$ para satisfacer las dos condiciones

$$\sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij} x_{ij} = t,$$

y

$$\sum_{j=1}^{n} y_j = m_i$$

Observe que la distribución de T es de la misma forma tanto para la logística estratificada como para la no estratificada. Regresión. Esto hace posible desarrollar un único algoritmo numérico para ambos casos.

2.2.24. GOOGLE COLAB

Según Google, Google Colab que es la abreviatura de Google Colaboratory, es un servicio en la nube que ofrece una base de datos, ofrece un espacio de trabajo científico para Python muy similar a Jupyter Notebook.

El cuaderno se puede cargar directamente en el servicio en la nube de Colab, los cuadernos de Colab se almacenan en Google Drive y se pueden compartir como lo haría con documentos u Hojas de cálculo de Google. Simplemente haga clic en el botón Compartir en la parte superior derecha de cualquier Colaboratory portátil o siga las instrucciones para compartir de Google Drive.

Los recursos de Colab no son ilimitados, tienen ciertas restricciones y límites de uso. La asignación de recursos que te ofrece Colab dependerá del uso que le das, estas limitaciones ayudan a que más usuarios puedan acceder sin coste adicional.

En Colab, los recursos se asignan dando prioridad a los casos prácticos interactivos. Están prohibidas las acciones asociadas a operaciones informáticas en bloque, las acciones que afecten negativamente a otras y las acciones orientadas a eludir nuestras políticas. No se permite hacer lo siguiente en entornos de ejecución de Colab:

- Alojar archivos, servir contenido multimedia u ofrecer otros servicios web que no estén relacionados con la computación interactiva de Colab.
- Descargar torrents o compartir archivos de punto a punto.
- Usar un escritorio remoto o SSH.
- Conectarse a proxies remotos.
- Minar criptomonedas.
- Ejecutar ataques de denegación de servicio.
- Craquear contraseñas.
- Usar varias cuentas para eludir las restricciones de acceso o de uso de recursos.
- Crear ultra falsos.

Todos los cuadernos de Colab se almacenan en Google Drive o puedes cargarlos desde GitHub.

El código se ejecuta en una máquina virtual dedicada a tu cuenta. Las máquinas virtuales se eliminan cuando pasan un tiempo inactivas y tienen un ciclo de vida máximo que aplica el servicio de Colab.

Colab puede ofrecer recursos sin coste adicional porque, por un lado, tiene límites de uso dinámicos que a veces fluctúan y, por otro lado, no ofrece recursos ilimitados ni garantizados. Por tanto, los límites de uso generales, los tiempos de espera por inactividad, la duración máxima de las máquinas virtuales, los tipos de GPU disponibles y otros factores pueden variar a lo largo del tiempo. Colab no publica estos límites. Uno de los motivos es

que pueden (y suelen) variar rápidamente.

El acceso a GPUs y TPUs a veces se concede de forma prioritaria a los usuarios que utilizan Colab de una manera interactiva en lugar de para ejecutar operaciones informáticas de larga duración y a los usuarios que han usado menos recursos en Colab recientemente. Eso significa que los usuarios que usan Colab para ejecutar operaciones informáticas de larga duración o que han usado más recursos recientemente tienen más posibilidades de que se les establezcan límites de uso y de que se les restrinja temporalmente el acceso a las GPUs y TPUs.

2.2.25. PYTHON

Para Lee (2014) Python es el lenguaje de programación, para ejecutar un programa de Python, se necesita un intérprete. El intérprete de Python es un programa que lee un programa de Python y luego ejecuta las declaraciones que se encuentran en él, una vez que su programa esté escrito y esté listo para probarlo, le dirá a Python intérprete para ejecutar su programa Python para que pueda ver lo que hace.
Python es gratuito y está disponible para su descarga desde Internet, dentro de los últimos algunos años hubo algunos cambios en el lenguaje de programación Python entre Python 2 y Python 3, estas diferencias entre las dos versiones de Python son bastante menores.

Para escribir programas en Python, necesita un editor para escribir el programa. Es conveniente tener un editor diseñado para escribir programas en Python. un editor que está diseñado específicamente para escribir programas se llama IDE o entorno de desarrollo integrado. Un IDE es más que un simple editor. Proporciona resaltado y sangría que puede ayudar mientras escribe un programa. También proporciona una forma de ejecutar su programa directamente desde el editor. Dado que normalmente ejecutará su programa muchas veces mientras lo escribe, es útil tener una forma de ejecutarlo rápidamente. Este autor recomienda el uso de Wing IDE 101. Este IDE es fácil de instalar y es gratuito, para uso educativo, Wing IDE 101 está disponible para Mac OS X, Microsoft Windows y Linux.

2.2.26. TENSORFLOW

Según Pramod y Avinash (2020) tensorflow se usa ampliamente como biblioteca de implementación de aprendizaje automático. Fue creado por Google como parte del proyecto Google Brain y luego estuvo disponible como producto de código abierto, ya que había múltiples marcos de aprendizaje profundo que captaban la atención de los usuarios. Con disponibilidad de código abierto, más y más personas en las comunidades de inteligencia artificial y aprendizaje automático pudieron adoptar TensorFlow y crea características y productos sobre él, no solo ayudó a los usuarios con la implementación del aprendizaje automático estándar y algoritmos de aprendizaje profundo, sino también les permitió implementar diferentes algoritmos para aplicaciones comerciales y diversos fines de investigación. De hecho, pronto se convirtió en una de las bibliotecas más populares en las comunidades de inteligencia artificial y aprendizaje automático, tanto que la gente ha estado creando una gran cantidad de aplicaciones usando TensorFlow. Esto se debe principalmente al hecho de que el propio Google utiliza TensorFlow en la mayoría de sus productos, ya sea Google Maps, Gmail, u otras aplicaciones.

2.2.27. KERAS

Según Ketkar (2017), keras es una API de redes neuronales, una biblioteca que proporciona funciones muy potentes y abstractas, bloques de construcción para construir redes de aprendizaje profundo. Keras provee de bloques de construcción, estos se pueden hacer usando como base Tensorflow o Theano. Keras tiene la posibilidad de realizar cálculos con GPU Y CPU. Presentaremos una serie de bloques de construcción clave que proporciona Keras y luego construiremos una CNN y LSTM utilizando Keras..

CAPÍTULO III

MATERIAL Y MÉTODOS

3.1. TIPO DE INVESTIGACIÓN

A. TIPO DE INVESTIGACIÓN

De acuerdo a Hernández, Fernández y Baptista (2014), la Investigación aplicada se distingue por tener propósitos prácticos inmediatos definidos, es decir, se investiga para actuar, transformar, modificar o producir cambios en un determinado sector de la realidad. Para realizar investigaciones aplicadas es muy importante contar con el aporte de las teorías científicas, que son producidas por la investigación básica y sustantiva".

Por esta consideración el tipo de investigación es aplicada.

B. NIVEL DE INVESTIGACIÓN

De acuerdo al autor Hernández Sampieri et. al (2014), las investigaciones descriptivas son "los estudios descriptivos buscan especificar las propiedades, las características y los perfiles importantes de personas, grupos, comunidades o cualquier otro fenómeno que se someta a análisis." Una de las funciones principales de la investigación descriptiva es la capacidad para seleccionar las características fundamentales del objeto de estudio y su descripción detallada de las partes, categorías o clases de dicho objeto"; y agrega "En tales estudios se muestran, narran, reseñan o identifican hechos, situaciones, rasgos característicos de un objeto de estudio, o se diseñan productos, modelos, prototipos, guías, etcétera.

3.2. DISEÑO DE INVESTIGACIÓN

De acuerdo con el autor Hemández Sampieri et. al. (2014), se puede definir la investigación no experimental, "como aquella investigación que se realiza sin manipular deliberadamente las variables, se trata de estudios donde no hacemos variar en forma intencional las variables independientes para ver su efecto sobre otras variables. Lo que hacemos en la investigación no experimental es observar fenómenos tal como se dan en su contexto natural, para posteriormente analizarlos" (p. 191).

De acuerdo a Hernández et. al. (2014), una investigación de diseño transversal es “cuando se recolectan datos en un solo momento o en un tiempo único y su propósito es describir variables y analizar los hechos tal como se dan”. Los instrumentos de recolección de datos, son usados durante el proceso de forma única.

En esta investigación se tiene que conocer procesos, características, estudiar rasgos y entender funcionalidades para encontrar datos necesarios para la construcción modelo operativo; por lo tanto, el diseño de investigación es no experimental de tipo transversal descriptivo.

3.3. HIPÓTESIS DE LA INVESTIGACIÓN

“Las investigaciones de tipo descriptivo no requieren formular hipótesis; es suficiente plantear algunas preguntas de investigación que, como ya se anotó, surgen del planteamiento del problema, de los objetivos y, por supuesto, del marco teórico que soporta el estudio” (Bernal, 2010, p. 136).

La investigación que se desarrollara es de tipo descriptivo, por lo que no se pretende pronosticar hallar o verificar lo planteado en los objetivos, se optó por no plantear hipótesis.

3.4. POBLACIÓN Y MUESTRA

A. POBLACIÓN

La población la formaran los estudiantes universitarios que se matricularon en el segundo semestre del 2021.

B. MUESTRA

La muestra se dará por un muestreo aleatorio simple basada en proporción:

$$n = \frac{N \times Z^2 \times p \times (1-p)}{(N-1) \times e^2 + Z \times p \times (1-p)} = 150$$

Donde:

$N = 1000$
$Z = 1.96$
$e = 0.05\ (Error)$
$NC = 0.95\ (Nivel\ de\ confiabilidad)$
$\alpha = 0.05$
$p = 0.5$

3.5. DEFINICIÓN CONCEPTUAL DE LAS VARIABLES

VARIABLE DE ESTUDIO 1

Redes Neuronales Profundas. Son una de las arquitecturas tecnológicas más importantes usadas en la Deep Learning, son redes neuronales artificiales con varias capas ocultas entre las capas de entrada y salida. Para que una red neuronal sea considerada profunda debe tener al menos dos capas intermedias, es decir más de tres contando entradas y salidas.

DIMENSIONES

a. Sistemas conexionistas. Llamadas también Redes Neuronales Artificiales (RNA), son sistemas de procesamiento de la información cuya estructura y funcionamiento están inspirados en las redes neuronales biológicas.

Consisten en un conjunto de elementos simples de procesamiento llamados nodos o neuronas conectadas entre sí por conexiones que tienen un valor numérico modificable llamado peso. La actividad que una unidad de procesamiento o neurona artificial realiza en un sistema de este tipo es simple. Normalmente, consiste en sumar los valores de las entradas (inputs) que recibe de otras unidades conectadas a ella, comparar esta cantidad con el valor umbral y, si lo iguala o supera, enviar activación o salida (output) a las unidades a las que esté conectada.

b. **Función de activación.** La función de activación devuelve una salida que será generada por la neurona dada una entrada o conjunto de entradas. Cada una de las capas que conforman la red neuronal tienen una función de activación que permitirá reconstruir o predecir. Además, se debe considerar que en la red neuronal se usará una función no lineal debido a que le permite a el modelo adaptarse para trabajar con la mayor cantidad de datos.

c. **Propagación hacia atrás.** Es un mecanismo de propagación hacia atrás, funciona de la siguiente manera:

1) Se presentan las entradas a la red neuronal y se asignan valores aleatorios para los pesos de cada conexión entre neuronas, por lo que se obtiene un valor para la salida.

2) Se compara el valor obtenido con el valor real, se obtiene un error.

3) El error se distribuye a las neuronas de derecha a izquierda.

4) se ajustan los pesos tomando en cuenta la información el error.

5) se repite el proceso para la siguiente muestra de datos.

VARIABLE DE ESTUDIO 2

Estrés estudiantil. Son estímulos externos o internos que son significativamente percibidos por el estudiante, activan las emociones y provocan cambios fisiológicos que amenazan la salud y la supervivencia. Las respuestas al estrés de un estudiante dependen de su interpretación subjetiva de un acontecimiento. Por lo tanto, un mismo acontecimiento o situación puede percibirse de forma diferente.

DIMENSIONES

a. La depresión y abatimiento. Son un trastorno del estado de ánimo, puede ser temporal o permanente, la depresión se caracteriza por el abatimiento, infelicidad, en el que aparece la desmotivación, la pérdida de autoestima, el desinterés, el malhumor e incapacidad para tomar decisiones y muchas veces todo ello acompañado de síntomas físicos o somáticos como el cansancio, abatimiento, falta de energía, cansancio, falta de apetito y pérdida de peso consecuentemente. La persona presenta alteraciones no sólo en cómo se comporta, sino también en la percepción que tiene sobre sí misma y el entorno que le rodea.

b. **Ansiedad y angustia.** La ansiedad es una parte de la existencia humana, siendo ésta una respuesta adaptativa. En general, el término ansiedad alude a la combinación de distintas manifestaciones físicas y mentales que no son atribuibles a peligros reales, sino que se manifiestan ya sea en forma de crisis o bien como un estado persistente y difuso, pudiendo llegar al pánico; no obstante, pueden estar presentes otras características neuróticas tales como síntomas obsesivos o histéricos que no dominan el cuadro clínico. Si bien la ansiedad se destaca por su cercanía al miedo, se diferencia de éste en que, mientras el miedo es una perturbación cuya presencia se manifiesta ante estímulos presentes, la ansiedad se relaciona con la anticipación de peligros futuros, indefinibles e imprevisibles.

   Así también, la angustia es la emoción más universalmente experimentada por el ser humano, tiene un efecto de inmovilización y conduce al sobrecogimiento en innumerables ocasiones; se define como una emoción compleja, difusa y desagradable que conlleva serias repercusiones psíquicas y orgánicas en el sujeto.

Ambos son sentimientos vinculado a situaciones de desesperación, donde la característica principal es la pérdida de la capacidad de actuar voluntariamente por parte del sujeto.

c. **Trastornos digestivos y de sueño.** Los trastornos digestivos tienen relación directa con los trastornos de sueño que pueden generar un impacto negativo en la calidad de vida. El insomnio y la acidez estomacal son los trastornos más frecuentes que están asociados.

.

3.6. DEFINICIÓN OPERACIONAL DE LAS VARIABLES

VARIABLE DE ESTUDIO 1

Redes Neuronales Profundas

DIMENSIONES

a. Sistemas conexionistas (Redes neuronales Artificiales)
b. Función de activación
c. Propagación hacia atrás

VARIABLE DE ESTUDIO 2

Estrés estudiantil

DIMENSIONES

a. La depresión y abatimiento
b. Ansiedad y angustia
c. Trastornos digestivos y de sueño

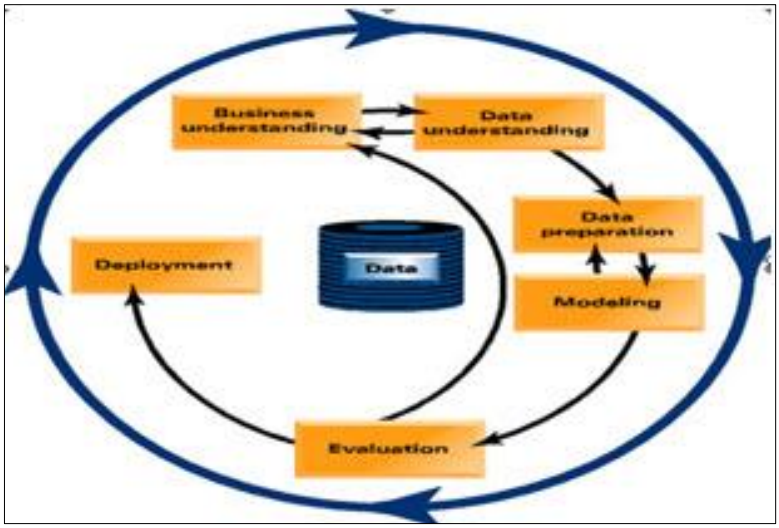
3.7. ASUM – DM

IBM (2015), ASUM-DM es un acrónimo de Analytics Solutions Unified Method for Data Mining que significa método unificado de soluciones analíticas para minería de datos, es una iniciativa de la empresa IBM para extender y refinar el método CRISP-DM.

ASUM-DM fue publicada en el año 2015, IBM provee esta metodología vía un instalador que genera un documento web con un mapa del sitio, una descripción de los procesos que conforman ASUM-DM.

ASUM-DM está definido por IBM como un proceso iterativo para la implementación de proyectos de analítica predictiva de minería de datos basado en una combinación de la metodología ASUM de IBM con una metodología CRISP-DM.

Figura 23
Ciclo de extracción de datos de ASUM – DM.

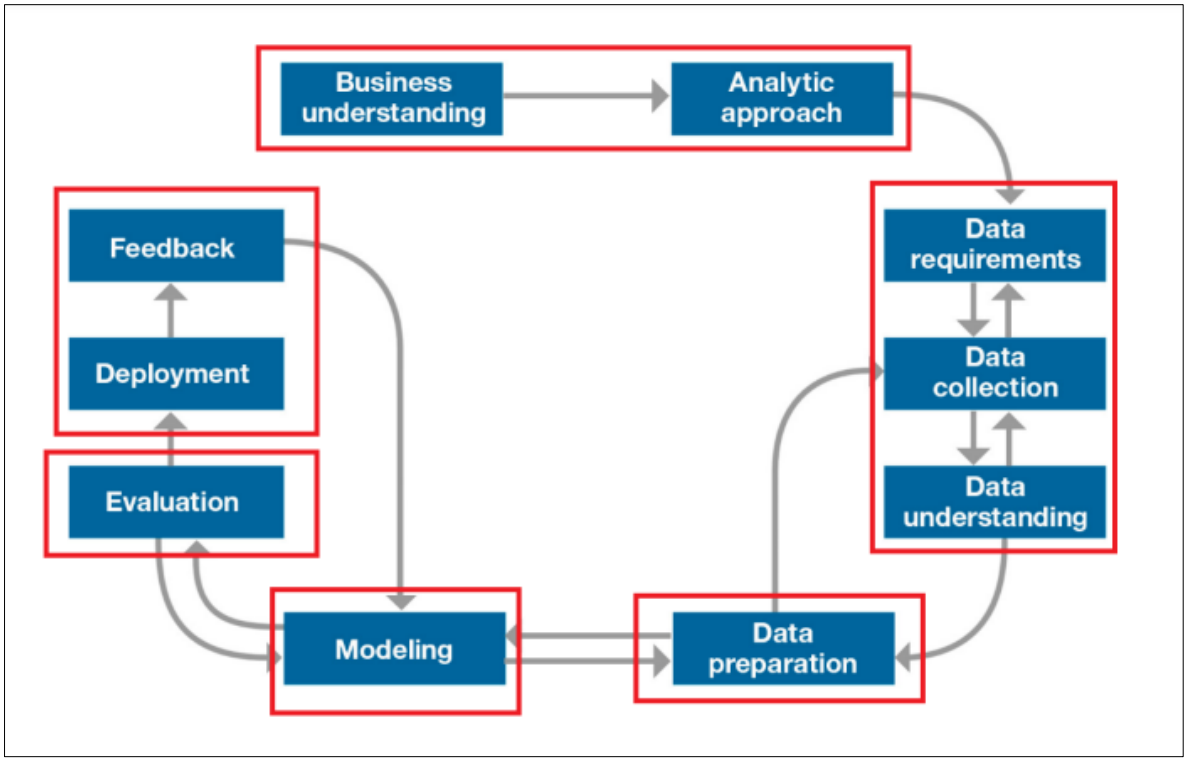


Nota: Fuente (IBM, 2015)

Esta metodología se concentra en 5 grupos de fases globales: 1 Analizar, 2 Diseñar, 3 Configurar y Construir, 4 Desplegar, 5 Operar y Optimizar. Dentro de cada uno de los grupos, se despliegan las 10 fases que componen la metodología ASUM-DM.

A continuación, se habla brevemente sobre cada elemento, para un mejor entendimiento, tomando como punto de partida la figura 23, en la cual se evidencia el agrupamiento de cada uno de los elementos dentro de las fases previamente mencionadas.

Figura 24
Metodología ASUM – DM de IBM.



Nota: Fuente (IBM, 2015)

Comprensión del Negocio: Esta primera etapa establece las bases para abordar de forma exitosa el proyecto ya que se orienta al entendimiento de un negocio, con el fin de definir proyectos y soluciones alienados con las estrategias de las organizaciones.

Enfoque Analítico: Implica traducir el problema de negocio a un problema técnico. Una vez que el problema de negocio de la empresa ha sido claramente establecido, el científico de datos puede definir el enfoque analítico para resolver el problema. Esta etapa implica expresar el problema en el contexto de las técnicas estadísticas y de aprendizaje automático, para que la organización pueda identificar y seleccionar las más adecuadas para el proyecto.

Requisitos de los datos: La elección del enfoque analítico determina los requisitos de los datos, ya que los métodos analíticos que se utilizan requieren un contenido de datos, formatos y representaciones particulares, guiados por el conocimiento del dominio.

Recolección de los datos: El científico de datos identifica y reúne los recursos de datos necesarios, relevantes para el dominio del problema. Al encontrar brechas en la recopilación de datos, el científico de datos podría necesitar revisar los requisitos de datos y recopilar más información.

Entendimiento de los datos: Técnicas de visualización o estadística pueden ayudar a un científico de datos a comprender el contenido de los datos, evaluar su calidad y tener hallazgos iniciales de interés para el proyecto.

Preparación de los datos: La etapa de preparación de datos comprende aquellas actividades para construir el conjunto de datos que se utilizará en la etapa de modelado. Estos incluyen la limpieza de datos y otras técnicas de análisis de datos para satisfacer la necesidad de tener un conjunto robusto para la construcción de modelos apropiados para abordar el problema.

Modelamiento: A partir de la primera versión del conjunto de datos preparado, la etapa de modelado se centra en el desarrollo de modelos predictivos de acuerdo con el enfoque analítico previamente definido. Con modelos predictivos, los científicos utilizan un conjunto de formación (datos históricos en los que se conoce el resultado del interés) para construir el modelo. Esta etapa suele ser iterativa, lo que lleva a refinamientos en la preparación de los datos y la especificación del modelo.

Evaluación: El científico de datos evalúa el modelo para entender su calidad y asegurarse de que aborda adecuada y completamente el problema del negocio. La evaluación del modelo implica el cálculo de diversas medidas de diagnóstico.

Despliegue: Después de que se ha desarrollado un modelo con resultados satisfactorios en su evaluación, se despliega en el entorno de producción o en un entorno de prueba comparable.

Retroalimentación: Al recolectar los resultados del modelo implementado, la organización obtiene retroalimentación sobre el rendimiento del modelo y observa cómo afecta su entorno de despliegue.

Una vez descritos los niveles que componen la metodología, cabe aclarar que esta es una

metodología planteada para el desarrollo del proyecto en un equipo de trabajo. Esto es un elemento fundamental para el cumplimiento de los objetivos planteados por el mismo. Por esta razón, implica un conocimiento base de cada uno de los integrantes del grupo en las diferentes áreas del conocimiento de un proyecto para tener un lenguaje común.

3.8. TÉCNICAS E INSTRUMENTOS

A. TÉCNICAS

Se escogió como técnica, el “Análisis Documental”, para revisar la información necesaria en el análisis de las variables y para construir los componentes que usamos al desarrollar la investigación.

Además, se uso la entrevista como técnica para validar con los expertos los instrumentos utilizados.

B. INSTRUMENTOS

Se elaboró un cuestionario basado en el instrumento: “Inventario de Sisco de Estrés Académico”, adicionando preguntas que nos ayudaran a adaptar este cuestionario al contexto actual de pandemia covid - 19, para obtener información que es necesaria para la investigación. El cuestionario de estrés académico se presenta en el Anexo A.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

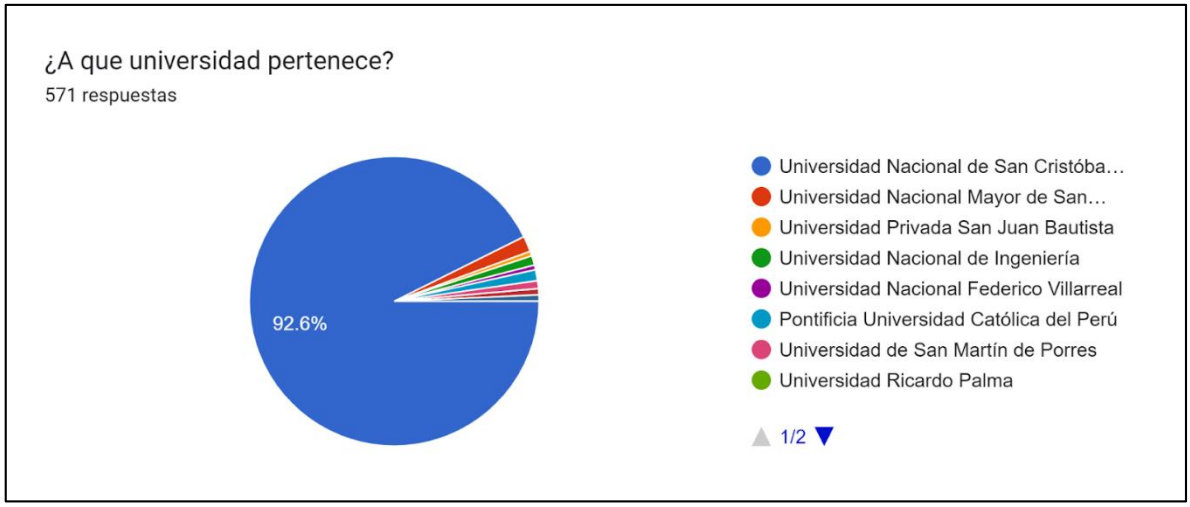
4.1. RECOLECCIÓN DE DATOS

Lo más importante para la investigación es encontrar datos de calidad, lo primero es tener un instrumento que nos ayude a esto, el cual ya hemos seleccionado y es el cuestionario de estrés académico, en el periodo covid-19, este se difundió en forma web a través de la herramienta de formulario de google, así mismo se requirió que estudiantes de pre grado llenaran este cuestionario, la difusión de este cuestionario se hizo a través de publicaciones en diversas páginas de distintas universidades, también se contó con el apoyo de docentes de la Universidad Nacional San Cristóbal de Huamanga.

El formulario se publicó el 10 de diciembre del 2021 y estuvo vigente para su llenado hasta el 20 de enero del 2022 en total se logró registrar 576 respuestas, de las cuales podemos ver los siguientes datos generales:

El 92.6 % que son 529 registros fueron de alumnos de la UNSCH, seguido del 1.8 % de alumnos de la Universidad Nacional Mayor de San Marcos y en menor porcentaje de diferentes universidades del país como se muestra en la figura 25.

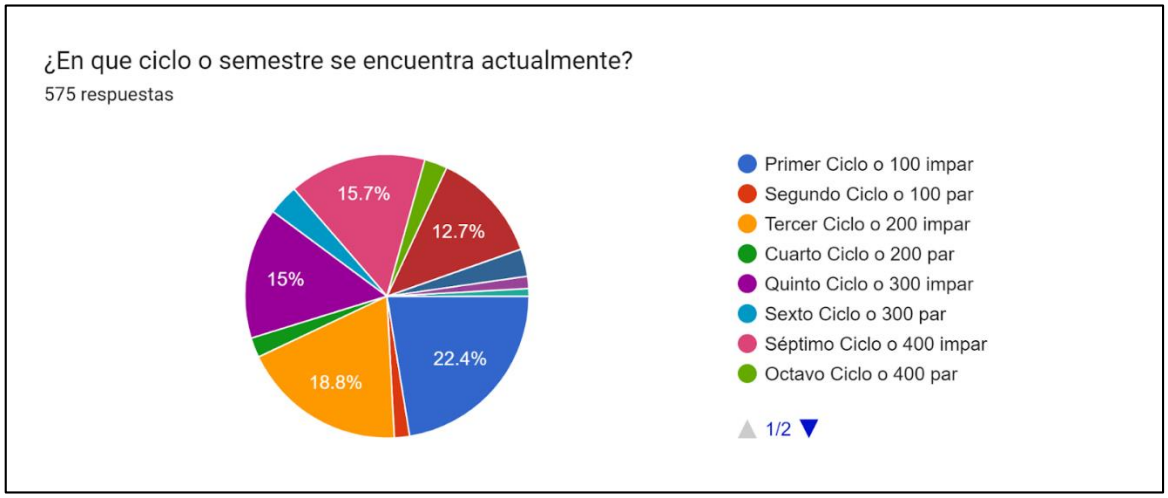
Figura 25
Gráfico de universidades a las que pertenecen los encuestados.



El 22.4 % que son 129 registros fueron de alumnos del primer ciclo o del semestre 100 impar, seguido del 18.8 % de alumnos de del tercer ciclo o del semestre 200 impar y ya en menores porcentajes de otros ciclos o semestres como se ve en la figura 26.

Figura 26

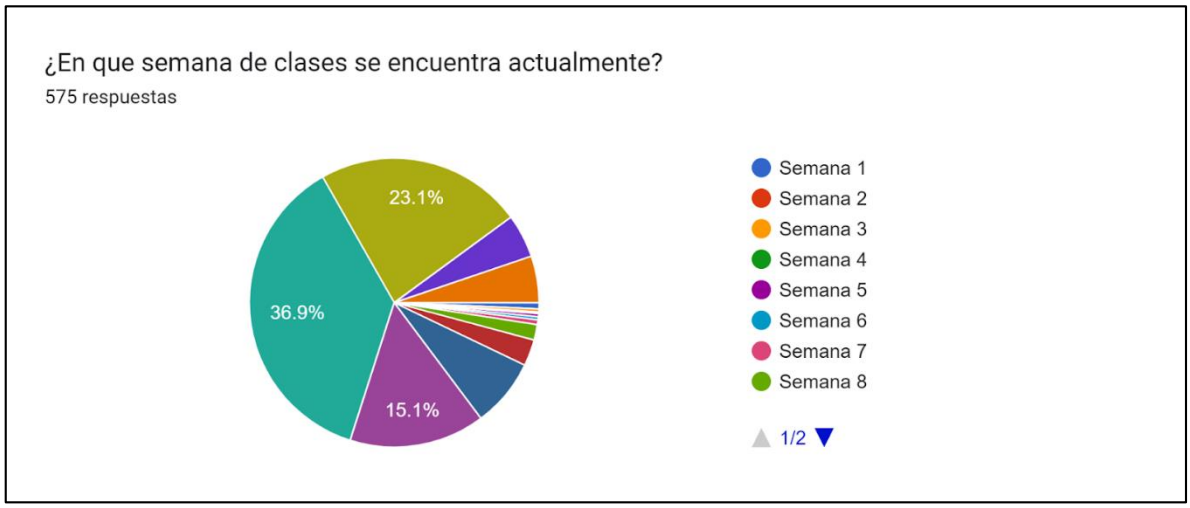
Gráfico de ciclos a los que pertenecen los encuestados.



El 36.9 % que son 212 registros fueron de alumnos que están en la semana 12 de su respectivo ciclo o del semestre, seguido del 23.1 % de alumnos que están en la semana 14 y ya en menores porcentajes se encuentran en diferentes semanas de ciclo o semestre como se muestra en la gráfica 27.

Figura 27

Gráfico de las semanas del semestre en que están los encuestados.

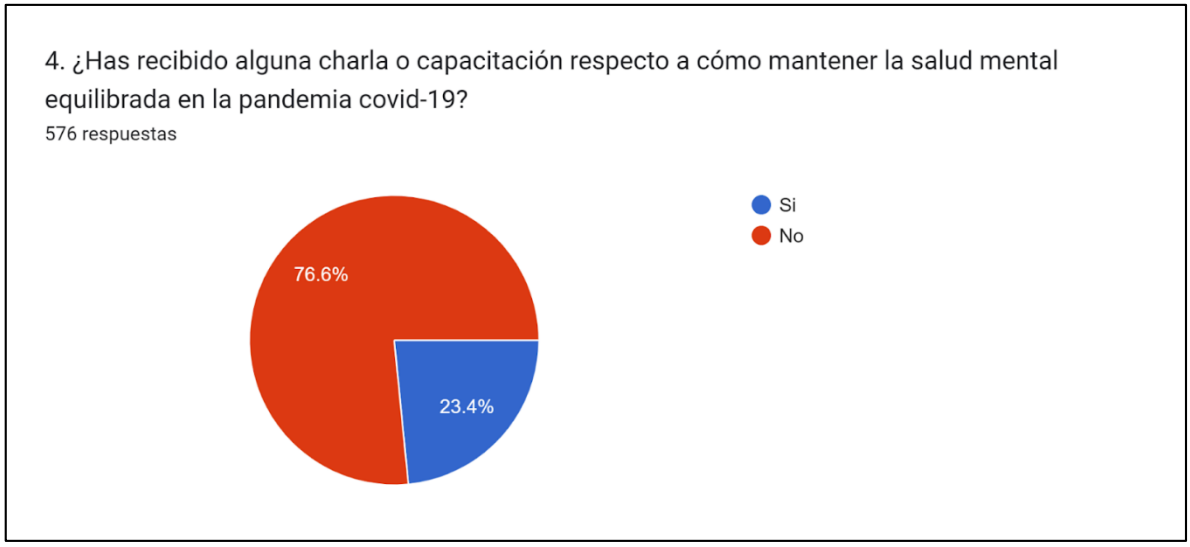


Además, para entender un poco más la realidad de los encuestados se realizó preguntas de su contexto respecto al covid-19:

Sobre si habían tenido alguna charla respecto a cómo mantener su salud mental equilibrada, la mayoría representada por el 76.6 % o 441 alumnos respondieron de forma negativa, mientras que el 23.4 % respondieron de forma afirmativa.

Figura 28

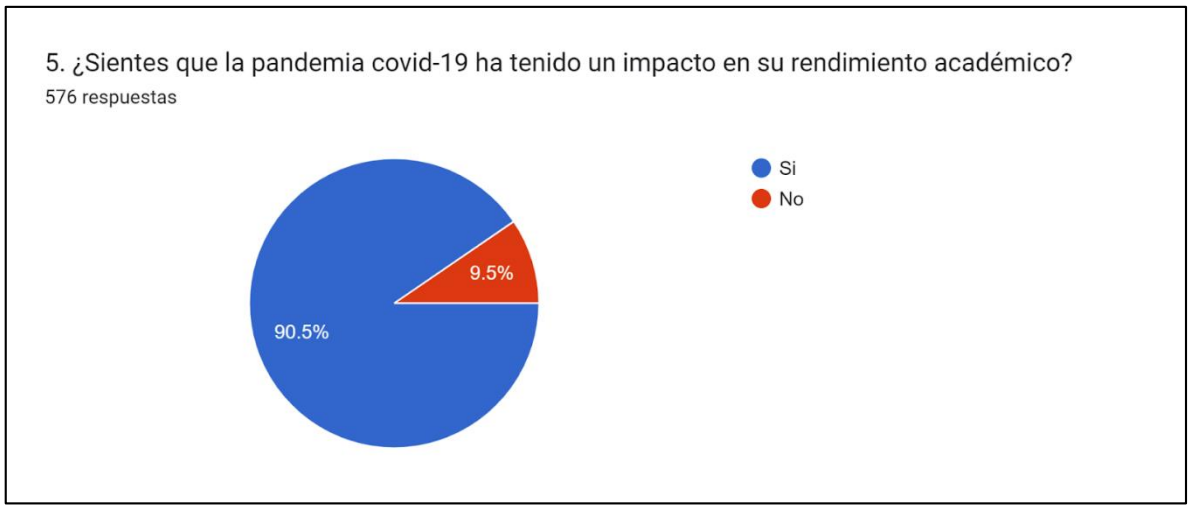
Gráfico de las respuestas sobre capacitaciones respecto a salud mental.



Sobre si sienten que la situación de covid-19 ha tenido un impacto en su rendimiento académico, la mayoría en un 90.5 % respondió que sí y solo el 9.5 % respondió que no.

Figura 29

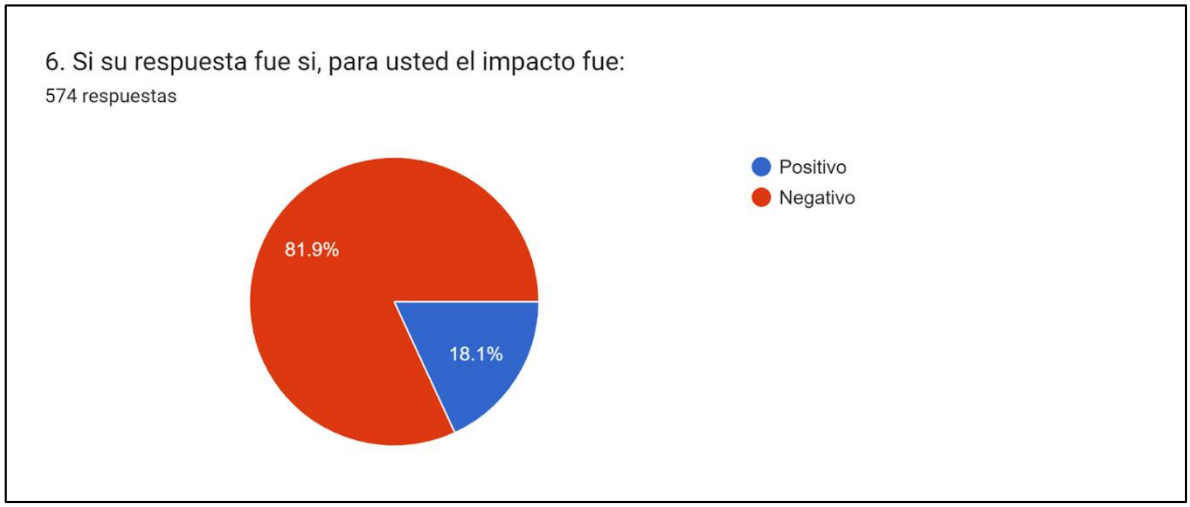
Gráfico de las respuestas sobre impacto de covid-19 en rendimiento académico.



A las personas que contestaron que sí, se les consulto si para ellos era un impacto negativo o positivo, la mayoría con un 81.9 % considera que fue un impacto negativo, mientras que el 18.1 % considera que fue un impacto positivo.

Figura 30

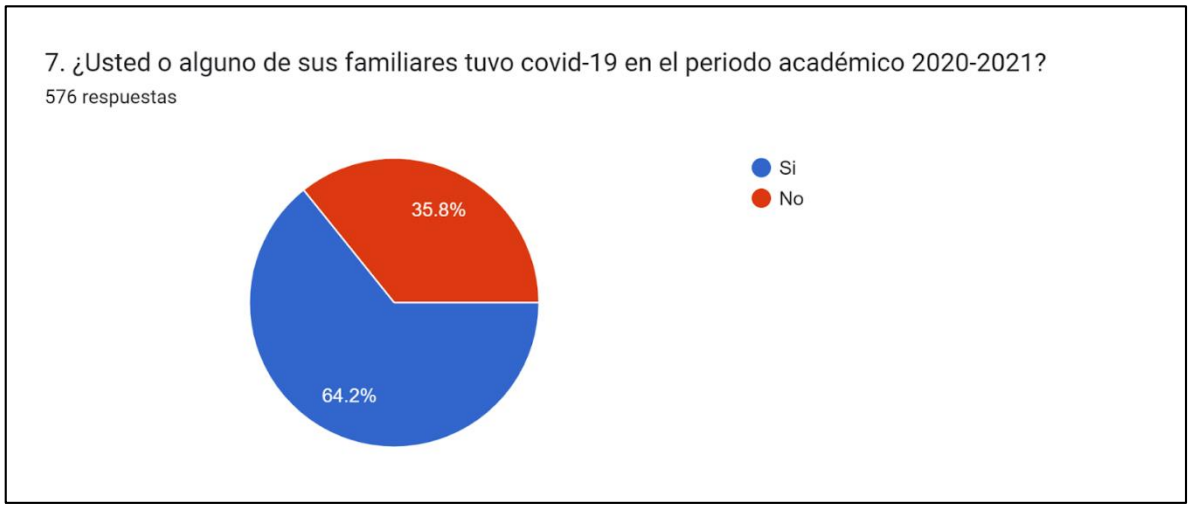
Gráfico de las respuestas sobre si el impacto de covid-19 fue positivo o negativo.



Sobre si los alumnos o alguno de sus familiares tuvo covid-19 en el periodo académico 2020-2021, la mayoría con un 64.2 % indico que sí, mientras que el 35.8 % respondió que no.

Figura 31

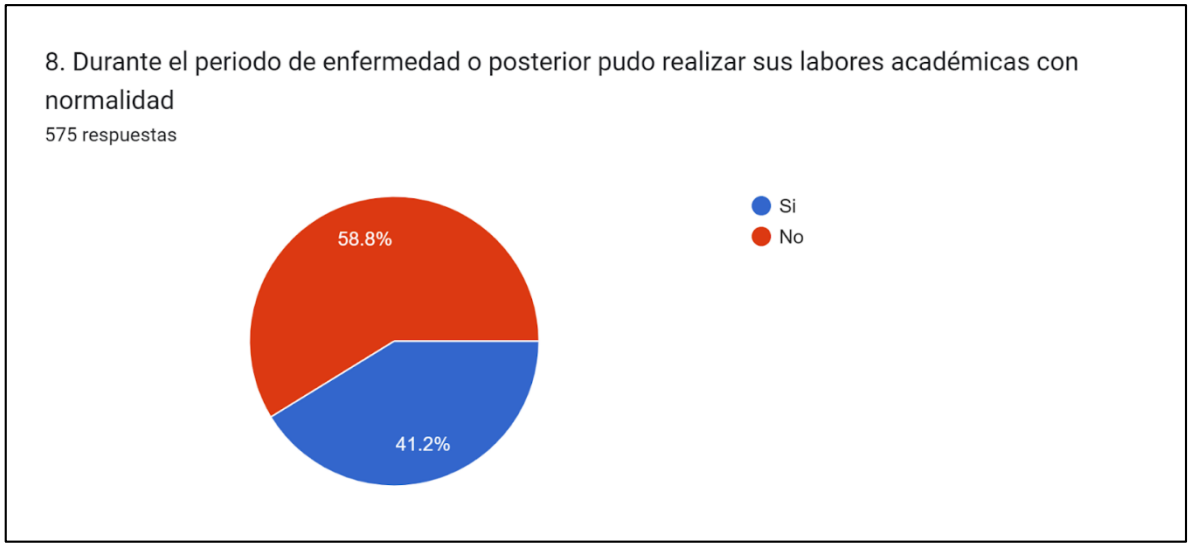
Gráfico de las respuestas sobre si tuvo contagio de covid-19.



Por último, se consultó si durante el periodo de enfermedad o posterior pudo realizar sus labores académicas con normalidad, a lo que la mayoría con un 58.8 % respondió que no, mientras que el 41.2 % respondió que sí.

Figura 32

Gráfico de las respuestas sobre si pudo realizar sus labores de forma cotidiana.



Las otras preguntas realizadas en el cuestionario son propias de la medición de indicadores de las variables de investigación usadas para determinar el estrés estudiantil que será usado en el análisis de datos.

4.2. LIMPIEZA Y TRANSFORMACIÓN DE DATOS

4.2.1. Limpieza de datos

Como se mencionó anteriormente se logró obtener un total de 576 registros realizados por estudiantes de pregrado, pero no todos estos registros nos sirven, se procedió a depurar registros que habían sido llenados con datos inconsistente, además al ser información que servirá para sacar patrones del estrés estudiantil, se eliminó todo registro incompleto, también se eliminó registros de personas que no fueran alumnos de pregrado universitario. Esta depuración de datos se realizó con ayuda de la herramientas y filtros que ofrece el programa de Microsoft Excel, obteniendo finalmente un total de 571 registros.

4.2.2. Feature engineering

Al tener la data limpia se procedió a empezar con el feature engineering o ingeniería de datos, en esta parte de la investigación se empieza a trabajar sobre los atributos del problema a investigar, esta parte es de vital importancia, para esto se consultó con la licenciada Sharon Villacorta Cabrera sobre si los atributos elegidos serían los adecuados para lograr determinar el nivel de estrés universitario, que es objetivo final

Se escogió 19 Features basados en la segunda variable de estudio, que es el estrés estudiantil, basada a la vez en las dimensiones de esta, donde se identificó los sintomas y estresores más comunes como se muestra en el siguiente cuadro:

Tabla 6

Elección de variables para determinar estrés estudiantil.

ESTRÉS ESTUDIANTIL	**Ansiedad y Angustia**	**Preocupacion - nerviosismo**
		Nivel de Inquietud
		Nivel de Ansiedad, angustia o desesperación
		Sentimiento de agresividad o aumento de irritabilidad
		Rascarse, morderse las uñas, frotarse, etc.
		Conflictos o tendencia a polemizar o discutir
	Depresion y Abatimiento	**Nivel de depresion-tristeza**
		Fatiga crónica (cansancio permanente)
		Aislamiento de los demás
		Desgano para realizar las labores educativas
	Trastornos digestivos y de sueño	**Trastornos en el sueño (insomnio o pesadillas)**
		Dolores de cabeza o migrañas
		Problemas de digestión, dolor abdominal o diarrea
		Cambios en los horarios de alimentación
		Cambios en los horarios de sueño
		Somnolencia o mayor necesidad de dormir
	Trastornos de aprendizaje	**Problemas de concentración**
		Sobrecarga de trabajos educativos
		No entender los temas que se abordan en la clase

Además, por desarrollarse la investigación en un contexto especifico, se está considerando 6 atributos propios del entorno, esto lo podemos ver en el siguiente cuadro:

Tabla 7
Elección de variables entorno de covid -19.

Ambito/ Entorno	**Nivel de miedo al coronavirus(covid-19)**
	Nivel de miedo de perder la vida a causa del coronavirus(covid-19)
	Nivel de Impacto de noticias sobre el coronavirus(covid-19).
	Recibio capacitaciones respecto a mantener la salud mental en la pandemia covid-19
	Impactodel Covid 19 en su rendimiento académico
	Tuvo covid-19 en el periodo académico 2020-2021

Por ultimo tendremos la definición del label que en este caso es el nivel de estrés estudiantil en los estudiantes de pre grado, esta variable se determinó en función a los features definidos, y el tipo de niveles que presenta se detalla en el siguiente cuadro.

Tabla 8
Niveles de estrés estudiantil.

LABEL	NIVEL	TIPO DE DATO
Nivel de Estrés Academico	**Nivel leve de Estrés**	**1**
	Nivel moderado de Estrés	**2**
	Nivel severo de Estrés	**3**

4.2.3. Transformación de datos

Una vez escogidas las variables a usar, se comenzó a mapear el tipo de formato que tienen los datos, se inició al hacer una conversión de datos categóricos a datos numéricos, pues la mayoría de las variables trabaja con la escala Likert de 1(nunca) - 2(rara vez) - 3(algunas veces) - 4(casi siempre) - 5(siempre) y preguntas de si y no, estas transformaciones están especificadas en el siguiente cuadro:

Tabla 9

Tipos de respuestas y transformación por variable.

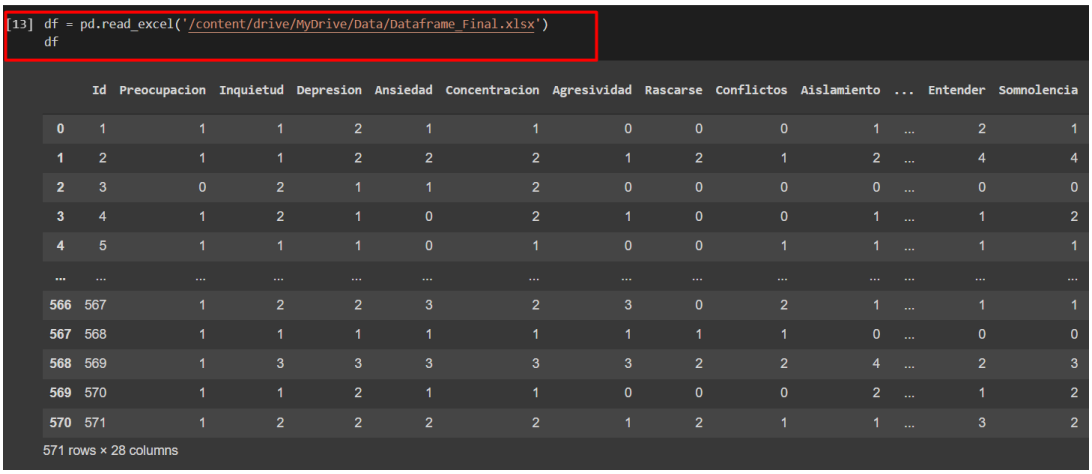
ATRIBUTOS	TIPO DE RESPUESTA/DATO	TRANSFORMACION
Recibio capacitaciones respecto a mantener la salud mental en la pandemia covid-19 Impacto del Covid 19 en su rendimiento académico Tuvo covid-19 en el periodo académico 2020-2021 Preocupacion - nerviosismo	SI / NO	0 / 1
Nivel de Inquietud Nivel de Ansiedad, angustia o desesperación Sentimiento de agresividad o aumento de irritabilidad Rascarse, morderse las uñas, frotarse, etc. Conflictos o tendencia a polemizar o discutir Nivel de depresion-tristeza Fatiga crónica (cansancio permanente) Aislamiento de los demás Desgano para realizar las labores educativas Trastornos en el sueño (insomnio o pesadillas) Dolores de cabeza o migrañas Problemas de digestión, dolor abdominal o diarrea Cambios en los horarios de alimentación Cambios en los horarios de sueño Somnolencia o mayor necesidad de dormir Problemas de concentración Sobrecarga de trabajos educativos No entender los temas que se abordan en la clase Nivel de miedo al coronavirus(covid-19) Nivel de miedo de perder la vida a causa del coronavirus(covid-19) Nivel de Impacto de noticias sobre el coronavirus(covid-19).	NUNCA/ RARA VEZ/ ALGUNAS VECES/ CASI SIEMPRE/ SIEMPRE	0 / 1 / 2 / 3 / 4

4.3. VALIDACIÓN DE DATAFRAME

Con la limpieza y transformación de los datos procederemos a subir nuestro dataframe al Drive de nuestro correo institucional, utilizaremos el Google Colaboratory o Colab como la herramienta que ejecuta el código Python a través del navegador, esta herramienta nos ayudara a realizar la validación del dataframe, validar que la data que tenemos sea de calidad, es validar que nos sirva para encontrar patrones, para esto usaremos el método de validación cruzada o Cross validation.

Lo primero estableceremos conexión al repositorio de nuestros datos, que están en el Google Drive y leeremos el dataframe con nuestros actuales valores.

Figura 33
Conexión a archivo de datos



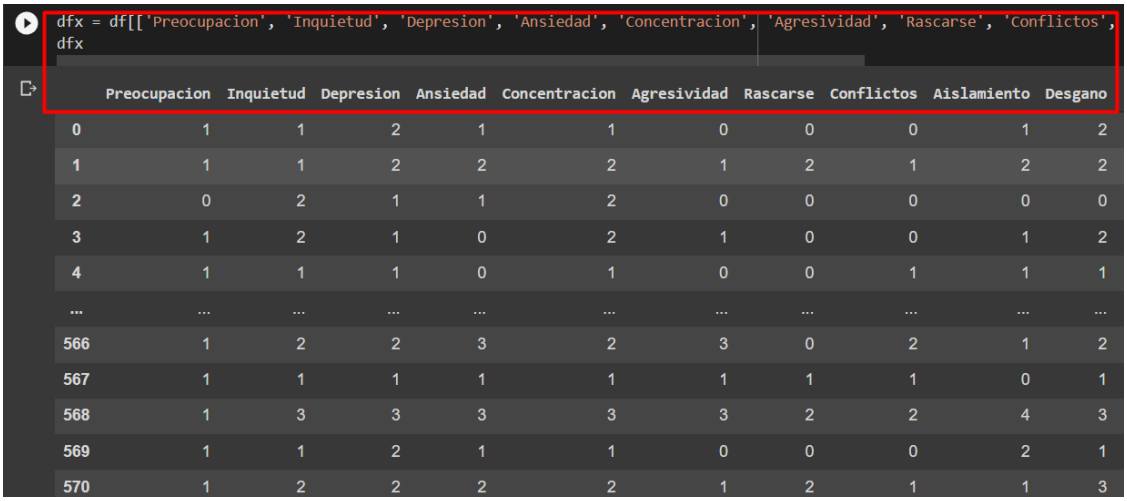
```
[13] df = pd.read_excel('/content/drive/MyDrive/Data/Dataframe_Final.xlsx')
     df
```

	Id	Preocupacion	Inquietud	Depresion	Ansiedad	Concentracion	Agresividad	Rascarse	Conflictos	Aislamiento	...	Entender	Somnolencia
0	1	1	1	2	1	1	0	0	0	1	...	2	1
1	2	1	1	2	2	2	1	2	1	2	...	4	4
2	3	0	2	1	1	2	0	0	0	0	...	0	0
3	4	1	2	1	0	2	1	0	0	1	...	1	2
4	5	1	1	1	0	1	0	0	1	1	...	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
566	567	1	2	2	3	2	3	0	2	1	...	1	1
567	568	1	1	1	1	1	1	1	1	0	...	0	0
568	569	1	3	3	3	3	3	2	2	4	...	2	3
569	570	1	1	2	1	1	0	0	0	2	...	1	2
570	571	1	2	2	2	2	1	2	1	1	...	3	2

571 rows × 28 columns

Definimos los dataframe con los features, que son las 25 variables

Figura 34
Definición de features.



```
dfx = df[['Preocupacion', 'Inquietud', 'Depresion', 'Ansiedad', 'Concentracion', 'Agresividad', 'Rascarse', 'Conflictos',
dfx
```

	Preocupacion	Inquietud	Depresion	Ansiedad	Concentracion	Agresividad	Rascarse	Conflictos	Aislamiento	Desgano
0	1	1	2	1	1	0	0	0	1	2
1	1	1	2	2	2	1	2	1	2	2
2	0	2	1	1	2	0	0	0	0	0
3	1	2	1	0	2	1	0	0	1	2
4	1	1	1	0	1	0	0	1	1	1
...	...	...	...	...	...	...	...	...	...	...
566	1	2	2	3	2	3	0	2	1	2
567	1	1	1	1	1	1	1	1	0	1
568	1	3	3	3	3	3	2	2	4	3
569	1	1	2	1	1	0	0	0	2	1
570	1	2	2	2	2	1	2	1	1	3

Al tener valores categóricos en las diferentes variables, se procederá a crear variables Dummy para que la neurona pueda procesar correctamente los datos, agregando nuevas columnas al Dataframe, quedando al final con 109 columnas.

Figura 35
Código de inserción de variables dummy.

```
[70] def createdummies(dfx,varname):
       dummy = pd.get_dummies(dfx[varname], prefix=varname)
       dfx = dfx.drop(varname, axis = 1)
       dfx = pd.concat([dfx, dummy], axis = 1)
       return dfx

[71] df1=createdummies(dfx, "Inquietud")
     df1
```

Inquietud_0	Inquietud_1	Inquietud_2	Inquietud_3	Inquietud_4
0	1	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	1	0	0
0	1	0	0	0
...	...	...	...	...
0	0	1	0	0
0	1	0	0	0

```
df21 = createdummies(df20, "Noticias")
df21
```

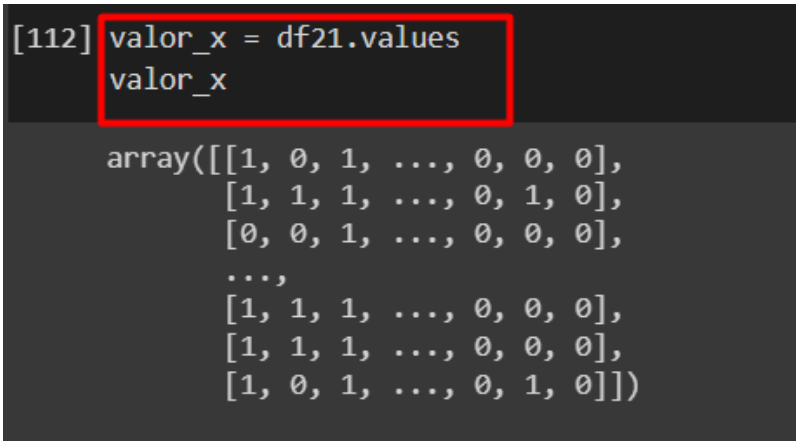
	Preocupacion	Capacitacion	Rendimiento	Covid	Inquietud_0	Inquietud_1	Inquietud_2
0	1	0	1	1	0	1	0
1	1	1	1	1	0	1	0
2	0	0	1	1	0	0	1
3	1	0	1	1	0	0	1
4	1	0	1	1	0	1	0
...	...	...	...	...	...	...	...
566	1	0	1	1	0	0	1
567	1	1	1	1	0	1	0
568	1	1	1	1	0	0	0
569	1	1	1	1	0	1	0
570	1	0	1	1	0	0	1

571 rows × 109 columns

Llevaremos todos los valores de la feature a una matriz.

Figura 36

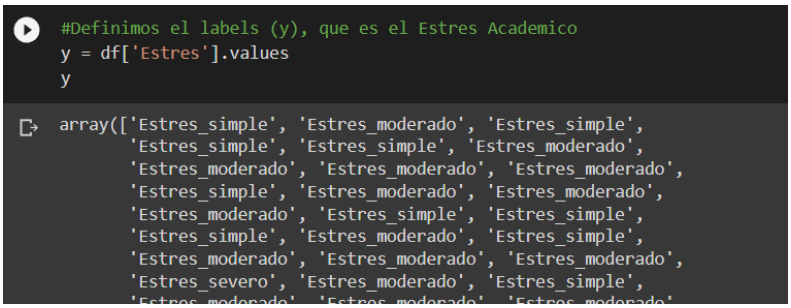
Features insertadas a matriz.



Definimos el Label, que es el nivel de estrés académico

Figura 37

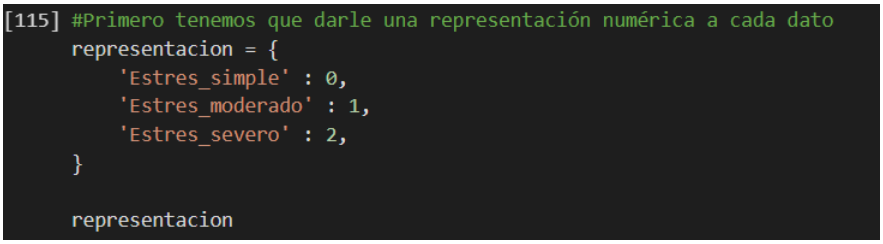
Definicion de Label



Al ser también una variable categórica se llevará a una representación numérica.

Figura 38

Representación de Labels a variables numéricas.



Lo llevaremos a una matriz que nos representa la categoría que representa.

Figura 39

Representación en matriz de los labels.

```
[117] from keras.utils.np_utils import to_categorical

[118] #Creamos tres columnas de salidas
      y_categorico = to_categorical(y_numerico)
      y_categorico

      array([[1., 0., 0.],
             [0., 1., 0.],
             [1., 0., 0.],
             ...,
             [0., 1., 0.],
             [1., 0., 0.],
             [0., 1., 0.]], dtype=float32)
```

Definimos una función que se encarga de crear y configurar un modelo de 3 perceptrones, con una regresión logística, el tipo de perceptrón será softmax al ser multivarible.

Figura 40

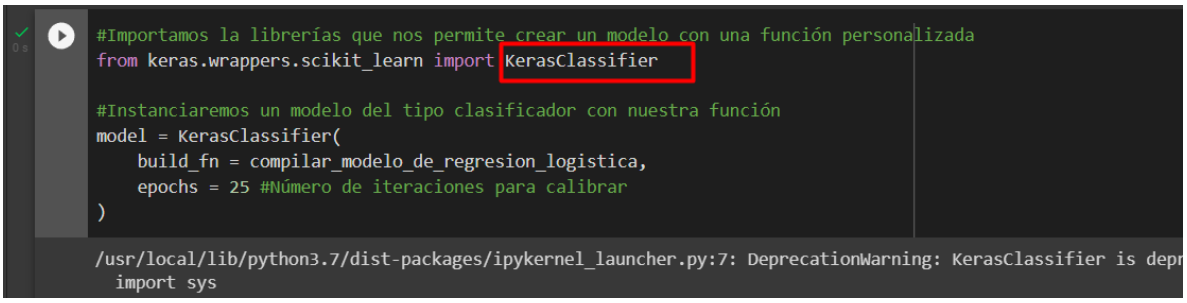
Código inicial del modelo.

```
#Vamos a definir una función que se encarga de crear y configurar un modelo
def compilar_modelo_de_regresion_logistica():
  #Instanciamos un modelo vacio
  model = Sequential()

  #Como estamos ante un problema de regresión logística, agregamos tres perceptrones
  #con activacion softmax por ser multivariable
  model.add(Dense(3, input_shape=(109, ), activation='softmax'))

#Compilamos el modelo
  model.compile(SGD(lr=0.5), 'categorical_crossentropy', metrics=['accuracy'])

  return model
```

Procedemos a importar el objeto KerasClassifier de la librería Keras, para poder redefinir el modelo en cada validación que realizamos.

Figura 41

Llamada a la libreria KerasClassifier.



Procedemos a definir nuestra validación cruzada que use el modelo anteriormente definido, para esto importamos el objeto KFold, que definirán el número de validación cruzadas que se realizara y el objeto cross_val_score que nos ayuda a ver el valor de cada una de las validaciones.

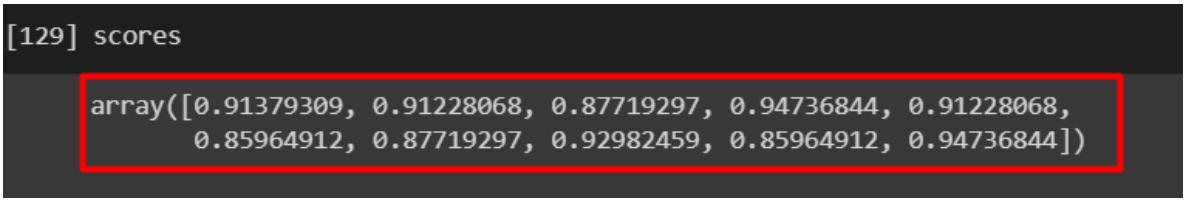
Figura 42

Ejecución de validaciones.

Utilizamos la sentencia scores para ver el resultado de la validación.

Figura 43

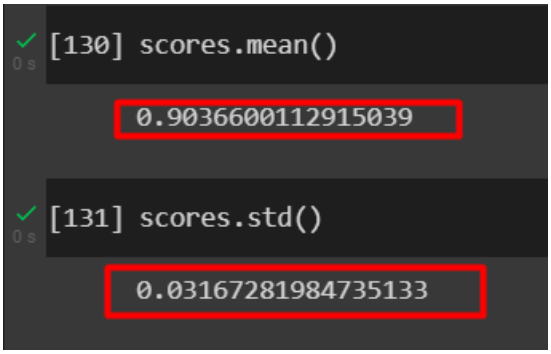
Resultado de la validación.



Evaluando la desviación estándar obtenemos que:

Figura 44

Resultado de la validación estándar.



La aleatoriedad de la data es de un 3%, se acepta como estándar que, si la desviación estándar es menor a 15%, esta data es válida para aplicarla a un modelo.

4.4. CALIBRACIONES

4.4.1. Calibración del ratio de aprendizaje o learning rate

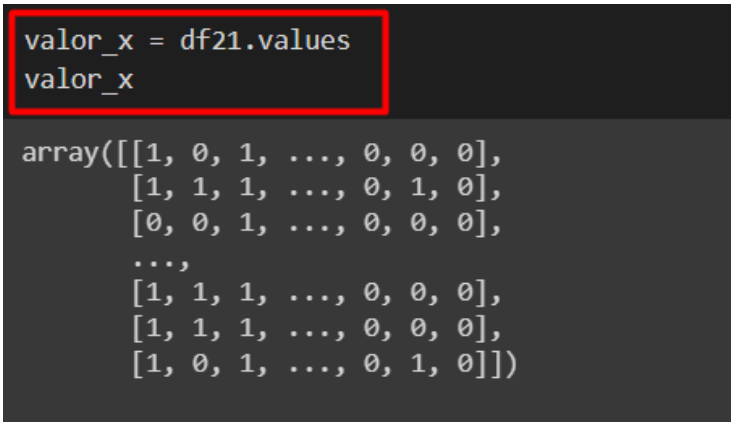
Para poder aplicar nuestro modelo de forma correcta, se determinó el valor optimo del ratio de aprendizaje, para esto se analizó e hizo pruebas con diversos valores, como se muestra a continuación:

Primero realizaremos la separación del dataframe en los features y label, como se hizo en la validación del dataframe.

Los valores de las features se encontrarán en la matriz llamada valor_x

Figura 45

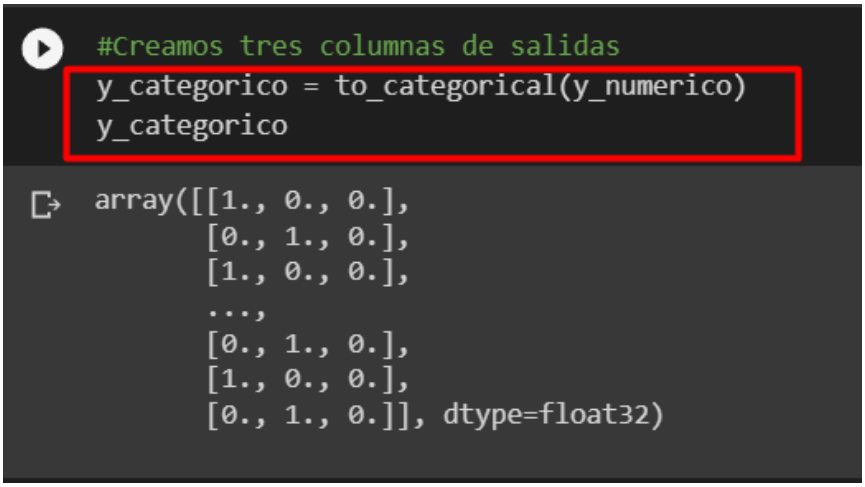
Valores en matriz de la feature.



Segundo definiremos el label en la matriz llamada y_categorico

Figura 46

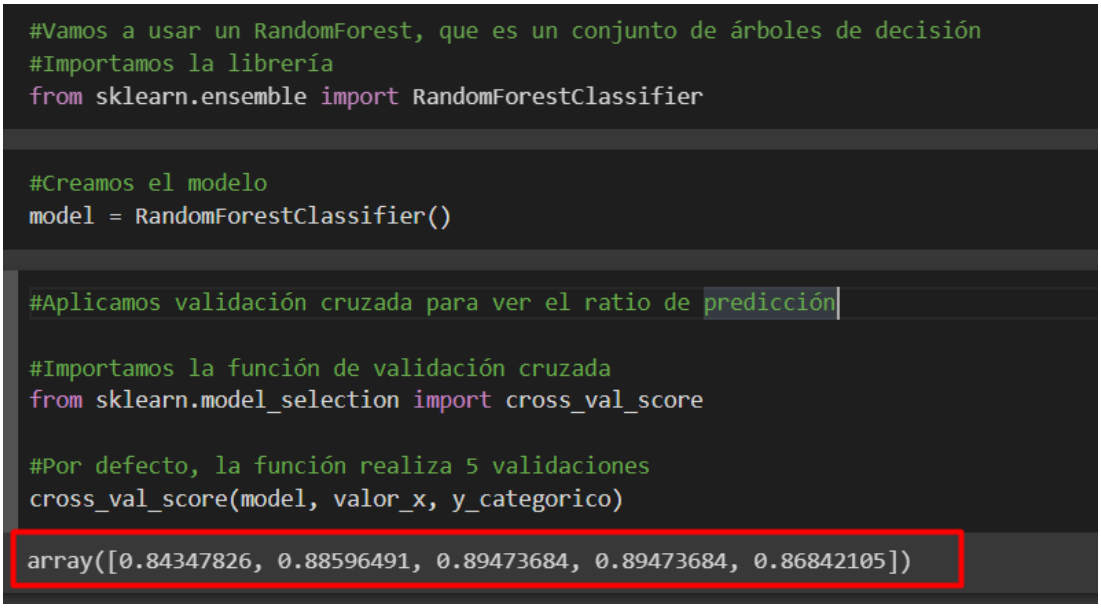
Valores categóricos en matriz.



Tercero haremos una validación cruzada con el RandomForestClassifier que es un método de machine learning, donde valimos la calidad de la data, como se muestra a continuación.

Figura 47

Validación cruzada.



Cuarto paso construiremos una red neuronal con los feature y label, los dividiremos en datos de entrenamiento y datos de validación.

Figura 48
Código de la división de la data.

```
#Importamos la librería para dividir los datos
from sklearn.model_selection import train_test_split

#Dividimos los datos en datos de entrenamiento (x_train, y_train) y datos de validación (x_test, y_test)
x_train, x_test, y_train, y_test = train_test_split(valor_x, y_categorico, test_size = 0.2)
```

Como quinto paso definiremos un modelo básico para calibrar el rango de aprendizaje.

Figura 49
Calibración de rango de aprendizaje.

```
#Instaciamos un modelo vacio con "Sequential"
model = Sequential()

#Agregamos sólo 3 neurona con 109 entradas con la F.A. más adecuada es la "softmax"
model.add(Dense(3, input_shape=(109,), activation='softmax'))

#Compilamos probando un "learning rate" (lr) pequeño de 0.01
model.compile(
  loss='categorical_crossentropy',
  optimizer=SGD(lr=0.01),
  metrics=['accuracy']
)

/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(SGD, self).__init__(name, **kwargs)
```

Al empezar a entrenar el modelo vemos que el valor puesto al rango de aprendizaje es óptimo, por su nivel de crecimiento.

Figura 50
Validación de entrenamiento del modelo.

```
#Entrenamos el modelo
#Esta vez al momento de entrenar, vamos a guardar la descripción del entrenamiento
descripcion_de_entrenamiento = model.fit(x_train, y_train, epochs = 20)

Epoch 1/20
15/15 [==============================] - 0s 2ms/step - loss: 0.5516 - accuracy: 0.7807
Epoch 2/20
15/15 [==============================] - 0s 2ms/step - loss: 0.5321 - accuracy: 0.7939
Epoch 3/20
15/15 [==============================] - 0s 2ms/step - loss: 0.5140 - accuracy: 0.7982
Epoch 4/20
15/15 [==============================] - 0s 2ms/step - loss: 0.4981 - accuracy: 0.8070
Epoch 5/20
15/15 [==============================] - 0s 2ms/step - loss: 0.4829 - accuracy: 0.8158
Epoch 6/20
15/15 [==============================] - 0s 2ms/step - loss: 0.4694 - accuracy: 0.8246
Epoch 7/20
15/15 [==============================] - 0s 2ms/step - loss: 0.4571 - accuracy: 0.8224
Epoch 8/20
15/15 [==============================] - 0s 2ms/step - loss: 0.4456 - accuracy: 0.8333
Epoch 9/20
15/15 [==============================] - 0s 2ms/step - loss: 0.4349 - accuracy: 0.8377
Epoch 10/20
15/15 [==============================] - 0s 2ms/step - loss: 0.4250 - accuracy: 0.8399
Epoch 11/20
15/15 [==============================] - 0s 2ms/step - loss: 0.4159 - accuracy: 0.8465
Epoch 12/20
15/15 [==============================] - 0s 2ms/step - loss: 0.4071 - accuracy: 0.8487
```

Para determinar el mejor rango de aprendizaje procederemos a probar valores que limiten con el rango de aprendizaje probado anteriormente, para esto ordenaremos los valores de evolución de ratio en un array de descripciones.

Figura 51

Valores de LR.

```
#Probaremos diferentes valores para lr
array_lr = [0.01, 0.05, 0.1, 0.5, 1 , 1.5, 2.5]

#Creamos un array en donde guardaremos las descripciones de cada entrenamiento
array_descripciones = []
```

Procederemos a crear un bucle que itere cada uno de las ratios de aprendizaje, y que guarde la descripción de los resultados de la evolución de entrenamiento en un dataframe.

Figura 52

Código de calibración de LR.

```
#Iteramos cada calibración
for lr in array_lr:
  model = Sequential()
  model.add(Dense(3, input_shape=(109,), activation='softmax'))

  #Agregamos el "lr" de la calibración
  model.compile(
    loss='categorical_crossentropy',
    optimizer=SGD(lr=lr),
    metrics=['accuracy']
  )
    #Entrenamos la neurona
  #Para evitar que el output de entrenamiento se muestre en pantalla desactivamos la variable "verbose"
  descripcion_de_entrenamiento = model.fit(x_train, y_train, epochs = 20, batch_size=16, verbose=0)

  #Convertimos la descripción del entrenamiento en un dataframe
  df = pd.DataFrame(
      descripcion_de_entrenamiento.history,
      index = descripcion_de_entrenamiento.epoch
  )
  #Agregamos el dataframe a la lista de descripciones de entrenamiento
  array_descripciones.append(df)
```

Vemos los resultados en el dataframe de esta forma, separados según el valor del rango de aprendizaje.

Figura 53
Rango de aprendizaje de LR

```
dfDescripcion.columns = nuevas_columnas
dfDescripcion
```

lr	0.01		0.05		0.10		0.50		1.00		1.50		2.50	
metricas	loss	accuracy	loss	accuracy	loss	accuracy	loss	accuracy	loss	accuracy	loss	accuracy	loss	accuracy
0	1.132744	0.425439	0.758476	0.640351	0.625576	0.754386	0.455842	0.811404	0.511734	0.769737	0.655587	0.787281	0.831729	0.787281
1	0.873827	0.642544	0.474868	0.826754	0.370840	0.859649	0.210901	0.927632	0.167889	0.929825	0.155806	0.932018	0.154550	0.927632
2	0.752838	0.708333	0.379172	0.861842	0.295275	0.883772	0.154393	0.956140	0.114746	0.964912	0.120035	0.958333	0.094664	0.964912
3	0.674649	0.732456	0.328093	0.881579	0.259056	0.903509	0.133707	0.960526	0.107183	0.969298	0.083010	0.967105	0.181718	0.929825
4	0.615522	0.752193	0.296141	0.894737	0.229854	0.925439	0.111930	0.978070	0.078611	0.978070	0.066750	0.978070	0.072748	0.971491
5	0.568618	0.782895	0.273069	0.901316	0.210822	0.932018	0.102884	0.973684	0.066597	0.989035	0.050111	0.989035	0.028971	0.993421
6	0.529587	0.798246	0.253776	0.905702	0.194583	0.940789	0.096520	0.982456	0.059496	0.993421	0.043064	0.995614	0.016814	1.000000
7	0.497340	0.815789	0.239273	0.923246	0.182648	0.945175	0.085013	0.984649	0.053528	0.997807	0.038372	0.997807	0.017962	1.000000
8	0.469961	0.826754	0.226704	0.921053	0.172333	0.945175	0.074652	0.993421	0.045080	0.997807	0.036246	0.997807	0.012673	1.000000
9	0.446230	0.835526	0.216205	0.927632	0.163696	0.962719	0.071027	0.991228	0.044428	0.997807	0.030426	0.997807	0.012960	1.000000
10	0.426023	0.846491	0.206992	0.927632	0.155911	0.960526	0.065835	0.997807	0.039543	0.997807	0.027521	0.997807	0.012166	1.000000
11	0.408400	0.850877	0.198925	0.936404	0.149806	0.962719	0.063330	0.995614	0.038246	0.997807	0.023933	1.000000	0.011769	1.000000

Como sexto paso empezaremos con el análisis del loss, para lo cual crearemos un nuevo dataframe con solo estos valores.

Figura 54
Rango de error de LR.

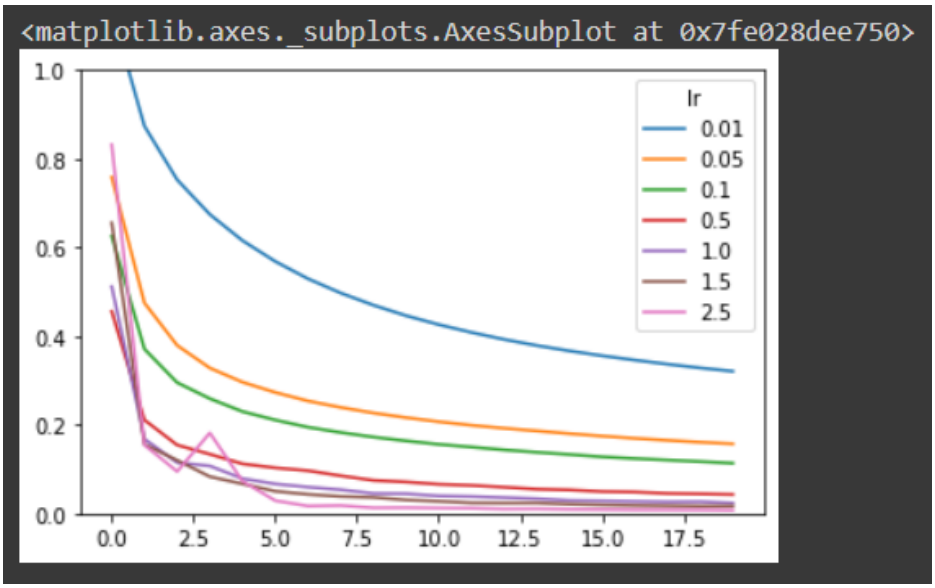
```
dfLoss = dfDescripcion.xs('loss', level = 'metricas', axis = 1)
dfLoss
```

lr	0.01	0.05	0.10	0.50	1.00	1.50	2.50
0	1.132744	0.758476	0.625576	0.455842	0.511734	0.655587	0.831729
1	0.873827	0.474868	0.370840	0.210901	0.167889	0.155806	0.154550
2	0.752838	0.379172	0.295275	0.154393	0.114746	0.120035	0.094664
3	0.674649	0.328093	0.259056	0.133707	0.107183	0.083010	0.181718
4	0.615522	0.296141	0.229854	0.111930	0.078611	0.066750	0.072748
5	0.568618	0.273069	0.210822	0.102884	0.066597	0.050111	0.028971
6	0.529587	0.253776	0.194583	0.096520	0.059496	0.043064	0.016814
7	0.497340	0.239273	0.182648	0.085013	0.053528	0.038372	0.017962
8	0.469961	0.226704	0.172333	0.074652	0.045080	0.036246	0.012673
9	0.446230	0.216205	0.163696	0.071027	0.044428	0.030426	0.012960
10	0.426023	0.206992	0.155911	0.065835	0.039543	0.027521	0.012166
11	0.408400	0.198925	0.149806	0.063330	0.038246	0.023933	0.011769
12	0.392593	0.192153	0.143392	0.059282	0.035759	0.023589	0.010095

Nos ayudaremos en el análisis con la evolución del porcentaje de error de manera gráfica.

```
#Vamos a graficar este dataframe
#Recordemos que mientras más pequeño sea el "loss" en cada iteración significa que el error se está minimizando
dfLoss.plot(ylim=(0,1))
```

Figura 55
Gráfico de la evolución del loss.



En el grafico nos es más claro ver que la mejor opción para el ratio de aprendizaje esta entre 0.5 y 1.0, se debe escoger el ratio de aprendizaje menor, que sea más óptimo, para este caso elegiríamos el valor de 0.5

Como Séptimo paso procederemos a analizar los valores del accuracy.

Figura 56
Valores de accuracy de LR.

```
dfAccuracy = dfDescripcion.xs('accuracy', level = 'metricas', axis = 1)
dfAccuracy
```

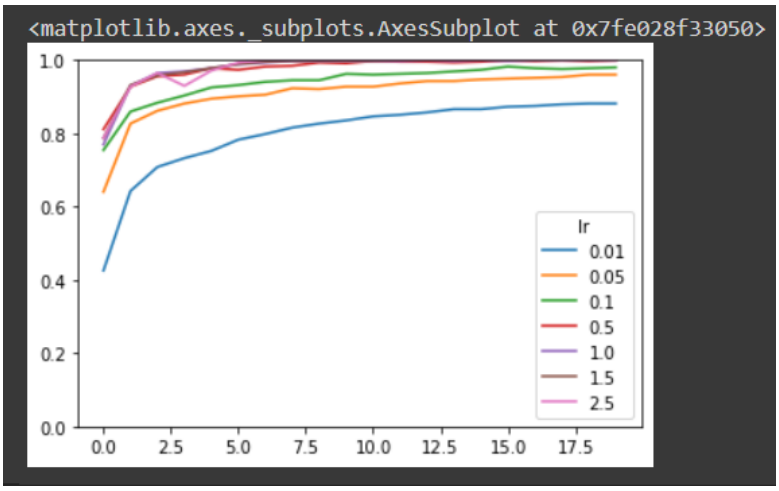
lr	0.01	0.05	0.10	0.50	1.00	1.50	2.50
0	0.425439	0.640351	0.754386	0.811404	0.769737	0.787281	0.787281
1	0.642544	0.826754	0.859649	0.927632	0.929825	0.932018	0.927632
2	0.708333	0.861842	0.883772	0.956140	0.964912	0.958333	0.964912
3	0.732456	0.881579	0.903509	0.960526	0.969298	0.967105	0.929825
4	0.752193	0.894737	0.925439	0.978070	0.978070	0.978070	0.971491
5	0.782895	0.901316	0.932018	0.973684	0.989035	0.989035	0.993421
6	0.798246	0.905702	0.940789	0.982456	0.993421	0.995614	1.000000
7	0.815789	0.923246	0.945175	0.984649	0.997807	0.997807	1.000000
8	0.826754	0.921053	0.945175	0.993421	0.997807	0.997807	1.000000
9	0.835526	0.927632	0.962719	0.991228	0.997807	0.997807	1.000000
10	0.846491	0.927632	0.960526	0.997807	0.997807	0.997807	1.000000
11	0.850877	0.936404	0.962719	0.995614	0.997807	1.000000	1.000000

Nos ayudaremos en el análisis con la evolución del accuracy de manera gráfica.

```
#Recordemos que mientras más pequeño sea el "accuracy" en cada iteración significa que los aciertos se están maximizando
dfAccuracy.plot(ylim=(0,1))
```

Figura 57

Gráfico de la evolución del accuracy.



En el grafico nos muestras que a partir de 0.5 el ratio de aprendizaje es similar, por lo cual reafirmamos la elección de este valor como el óptimo para el rango de aprendizaje.

4.4.2. Calibración del tamaño de batch

Para poder mejorar la funcionalidad del servidor, buscaremos optimizar nuestra data, para esto realizaremos un agrupamiento, a este agrupamiento se llama batch, esto lo realizamos para optimizar la infraestructura que utilizamos para el procesamiento de información, para esto se analizó e hizo pruebas con diversos valores, como se muestra a continuación:

Primero volveremos a realizar los pasos anteriores de la carga de data, y separaremos la data en data de entrenamiento y de prueba.

Figura 58

Código de separación de data.

```
[55] #Importamos la librería para dividir los datos
     from sklearn.model_selection import train_test_split

     #Dividimos los datos en datos de entrenamiento (x_train, y_train) y datos de validación (x_test, y_test)
     x_train, x_test, y_train, y_test = train_test_split(valor_x, y_categorico, test_size = 0.2)
```

Segundo, procederemos a definir un array con diferentes valores de tamaño de batch, con números que sean potencia de 2.

Figura 59
Valores para el batch.

```
array_batch = [2, 4, 8, 16, 32 , 64]

#Creamos un array en donde guardaremos las descripciones de cada entrenamiento
array_descripciones = []
```

Tercero, procederemos a iterar los diversos valores del batch en el modelo anteriormente definido, con el valor de rango de entrenamiento de 0.5

Figura 60
Código de calibración del batch.

```
#Iteramos cada calibración
for batch in array_batch:
  model = Sequential()
  model.add(Dense(3, input_shape=(109,), activation='softmax'))

  model.compile(
    loss='categorical_crossentropy',
    optimizer=SGD(lr=0.5),
    metrics=['accuracy']
  )
    #Entrenamos la neurona
    #En "batch_size" definimos el valor
  descripcion_de_entrenamiento = model.fit(x_train, y_train, epochs = 10, batch_size=batch, verbose=0)

    #Convertimos la descripción del entrenamiento en un dataframe
  df = pd.DataFrame(
      descripcion_de_entrenamiento.history,
      index = descripcion_de_entrenamiento.epoch
  )

  #Agregamos el dataframe a la lista de descripciones de entrenamiento
  array_descripciones.append(df)
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: UserWarning: The `lr` argumer
  super(SGD, self).__init__(name, **kwargs)
```

Vamos a llevar a un dataframe la evolución de los resultados y crearemos 2 dataframe el de loss y el de accuracy, para esto utilizaremos la siguiente funcion.

Figura 61
Código de resultados en array.

```
#Vamos a crear una función utilitaria para obtener el dataframe con los "accuracy" y "loss"
def obtener_df_loss_accuracy(array_calibracion, descripcion_calibracion, array_descripciones):
  #Fusionaremos uno tras otro para crear un único registro
  #Recordemos que al usar "axis = 1" estamos fusionandolas filas
  dfDescripcion = pd.concat(array_descripciones, axis = 1)
  #Obtenemos el primer elemento del array de dataframes para obtener los títulos de las columnas
  columnas_loss_accuracy = array_descripciones[0].columns

  #Vamos a crear nuevas columnas a nuestro dataframe
  #Primero colocaremos las columnas que tendrán el nombre de cada tb usado, estos valores están en "array_batch"
  nuevas_columnas = pd.MultiIndex.from_product(
    [array_calibracion, columnas_loss_accuracy],
    names=[descripcion_calibracion, 'metricas']
  )

  #Colocamos las nuevas columnas a nuestro dataframe
  dfDescripcion.columns = nuevas_columnas
  dfDescripcion

  #Vamos a crear el dataframe de "loss"
  dfLoss = dfDescripcion.xs('loss', level = 'metricas', axis = 1)

  #Vamos a crear el dataframe de "accuracy"
  dfAccuracy = dfDescripcion.xs('accuracy', level = 'metricas', axis = 1)
  dfAccuracy

  #Devolvemos los dataframes
  return dfLoss, dfAccuracy
```

Llamaremos a la función

```
[74] #Usamos la función
     dfLoss, dfAccuracy = obtener_df_loss_accuracy(array_batch, 'batch', array_descripciones)
```

Veremos el dataframe con los valores Loss.

Figura 62
Valores de loss del batch.

```
dfLoss
```

batch	2	4	8	16	32	64
0	0.603957	0.399776	0.395405	0.418534	0.523496	0.641732
1	0.245672	0.156609	0.166740	0.207052	0.249931	0.314221
2	0.174400	0.112539	0.127477	0.159893	0.201092	0.258316
3	0.059958	0.091283	0.097077	0.133376	0.172477	0.238444
4	0.038246	0.058748	0.080334	0.124062	0.157330	0.201506
5	0.032512	0.048778	0.068563	0.106534	0.138781	0.184021
6	0.017657	0.037417	0.061445	0.095583	0.128888	0.174813
7	0.008819	0.029613	0.056216	0.086607	0.118919	0.176237
8	0.007676	0.026718	0.047611	0.078988	0.114221	0.156002
9	0.006713	0.021808	0.045391	0.075073	0.109263	0.141869

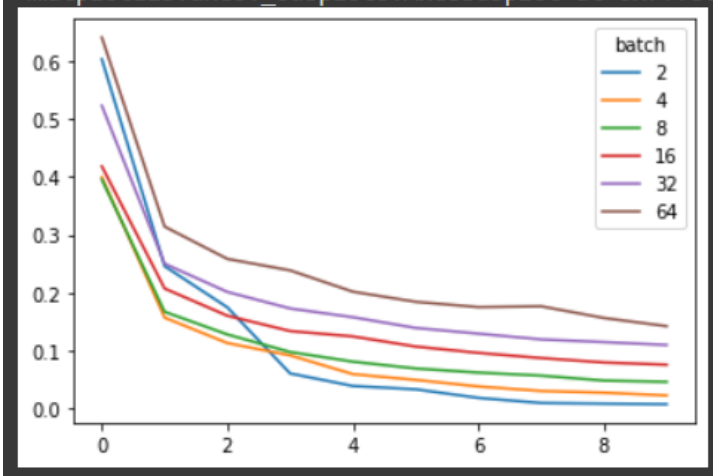
Nos ayudaremos en el análisis con la evolución del porcentaje de error de manera gráfica.

Figura 63

Gráfico de la evolución del loss del batch.

```
dfLoss.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff8c3341210>
```



Vemos que el menor error es para 2, pero vemos que tiene más variación, en cambio al ver la tendencia del valor 4 vemos que tiene una evolución más uniforme y óptima.

Procedemos a ver los valores del accuracy.

Figura 64

Valores de accuracy del batch.

```
dfAccuracy
```

batch	2	4	8	16	32	64
0	0.820175	0.831140	0.817982	0.813596	0.778509	0.717105
1	0.921053	0.925439	0.927632	0.918860	0.907895	0.892544
2	0.938596	0.953947	0.960526	0.940789	0.940789	0.901316
3	0.973684	0.969298	0.964912	0.956140	0.947368	0.907895
4	0.982456	0.978070	0.980263	0.964912	0.951754	0.923246
5	0.991228	0.982456	0.993421	0.971491	0.967105	0.938596
6	0.997807	0.997807	0.993421	0.980263	0.964912	0.951754
7	1.000000	0.997807	0.993421	0.980263	0.975877	0.945175
8	1.000000	1.000000	0.997807	0.991228	0.971491	0.947368
9	1.000000	1.000000	0.997807	0.980263	0.975877	0.962719

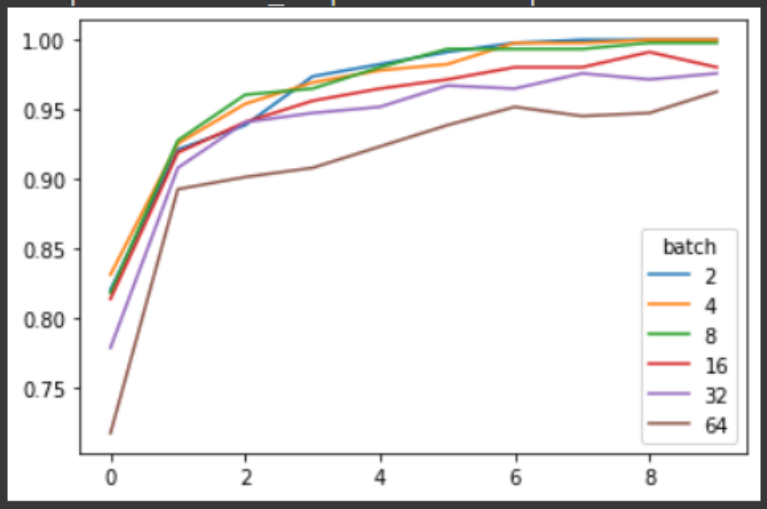
Nos ayudaremos en el análisis con la evolución del accuracy de manera gráfica.

Figura 65

Gráfico de la evolución del accuracy del batch.

```
[72] dfAccuracy.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff8bddcad90>
```



Aquí también vemos que los valores óptimos son 2, 4, 8, pero el que tiene una mayor uniformidad es el 4, por lo cual ese será nuestro valor de batch.

4.4.3. Calibración del optimizador

Para poder aplicar correctamente un modelo de redes neuronales, necesitamos definir el optimizador a utilizar, existen diversos algoritmos que implementan gradiente descendiente, para esto se analizó e hizo pruebas con diversos valores, como se muestra a continuación: Primero realizaremos los pasos anteriores hasta la división de la data en data de prueba y data de entrenamiento.

Figura 66

Código de separación de data.

```
#Importamos la librería para dividir los datos
from sklearn.model_selection import train_test_split

#Dividimos los datos en datos de entrenamiento (x_train, y_train) y datos de validación (x_test, y_test)
x_train, x_test, y_train, y_test = train_test_split(valor_x, y_categorico, test_size = 0.2)
```

Procederemos a importar los diversos optimizadores

Figura 67

Importación de librerías de optimizadores.

```
#Importamos las librerías de Keras
from keras.models import Sequential
from keras.layers import Dense, Activation
#Importamos los optimizadores
from tensorflow.keras.optimizers import SGD, Adam, Adagrad, RMSprop
```

Definimos los diversos optimizadores a probar

Figura 68

Definición de valores de optimizadores.

```
[77] #Probaremos diferentes valores para el optimizador
     array_optimizadores = [
     'SGD(lr=0.5)',
     'SGD(lr=0.5, momentum=0.3)',
     'SGD(lr=0.5, momentum=0.3, nesterov=True)',
     'Adam(lr=0.5)',
     'Adagrad(lr=0.5)',
     'RMSprop(lr=0.5)'
     ]
```

Definimos el array para almacenar las descripciones

```
#Creamos un array en donde guardaremos las descripciones de cada entrenamiento
array_descripciones = []
```

Realizamos la función con el bucle de iteraciones para los diversos valores del optimizador, con los valores ya definidos del ratio de aprendizaje y el valor del batch.

Figura 69
Calibración de optimizadores.

```
#Iteramos cada calibración
for optimizador in array_optimizadores:
  model = Sequential()
  model.add(Dense(3, input_shape=(109,), activation='softmax'))

  #En "lr" ya hemos calibrado su valor, sabemos que el valor de 0.5 es el indicado
  model.compile(
    loss='categorical_crossentropy',
    optimizer=eval(optimizador),
    metrics=['accuracy']
  )

  #Entrenamos la neurona
  #En "batch_size" colocamos 4, ya que fue lo que encontramos en la calibración anterior
  descripcion_de_entrenamiento = model.fit(x_train, y_train, epochs = 10, batch_size=4, verbose=0)

  #Convertimos la descripción del entrenamiento en un dataframe
  df = pd.DataFrame(
      descripcion_de_entrenamiento.history,
      index = descripcion_de_entrenamiento.epoch
  )

  #Agregamos el dataframe a la lista de descripciones de entrenamiento
  array_descripciones.append(df)
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: UserWarning: The `lr`
  super(SGD, self).__init__(name, **kwargs)
```

Llamaremos a la función para almacenar los resultados en el dataframe que nos muestre por separado los valores del error y de accuracy.

Figura 70
Código para dataframe de resultados de optimizadores.

```
#Vamos a crear una función utilitaria para obtener el dataframe con los "accuracy" y "loss"
def obtener_df_loss_accuracy(array_calibracion, descripcion_calibracion, array_descripciones):
  dfDescripcion = pd.concat(array_descripciones, axis = 1)
  columnas_loss_accuracy = array_descripciones[0].columns

  #Vamos a crear nuevas columnas a nuestro dataframe
  nuevas_columnas = pd.MultiIndex.from_product(
    [array_calibracion, columnas_loss_accuracy],
    names=[descripcion_calibracion, 'metricas']
  )

  #Colocamos las nuevas columnas a nuestro dataframe
  dfDescripcion.columns = nuevas_columnas
  dfDescripcion

  #Vamos a crear el dataframe de "loss"
  dfLoss = dfDescripcion.xs('loss', level = 'metricas', axis = 1)

  #Vamos a crear el dataframe de "accuracy"
  dfAccuracy = dfDescripcion.xs('accuracy', level = 'metricas', axis = 1)
  dfAccuracy

  #Devolvemos los dataframes
  return dfLoss, dfAccuracy
```

Llamaremos a la función y al dataframe con los resultados del Loss.

Figura 71

Valores de error de optimizador.

```
#Usamos la función
dfLoss, dfAccuracy = obtener_df_loss_accuracy(array_optimizadores, 'optimizador', array_descripciones)
```

```
#Obtenemos la evolución de los "loss"
dfLoss
```

optimizador	SGD(lr=0.5)	SGD(lr=0.5, momentum=0.3)	SGD(lr=0.5, momentum=0.3, nesterov=True)	Adam(lr=0.5)	Adagrad(lr=0.5)	RMSprop(lr=0.5)
0	0.400626	0.410355	0.480105	2.206053	0.482711	2.178655
1	0.152774	0.147820	0.197023	0.783276	0.124376	0.983915
2	0.096375	0.081605	0.115031	0.689616	0.051152	0.918046
3	0.073358	0.054974	0.062661	0.575228	0.040077	0.655488
4	0.056121	0.043357	0.040901	1.065122	0.028305	0.363731
5	0.043302	0.034474	0.029208	0.816756	0.025229	0.368226
6	0.039331	0.032411	0.018897	0.147138	0.021051	0.303165
7	0.031510	0.022143	0.016385	0.121059	0.018940	0.207429
8	0.028352	0.018065	0.014306	0.221816	0.017523	0.247898
9	0.023243	0.014128	0.013438	0.260643	0.016204	0.099251

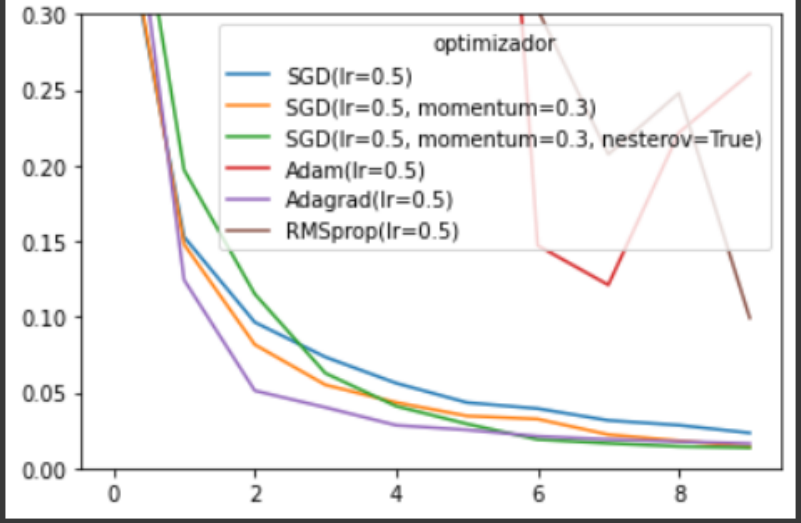
Nos ayudaremos en el análisis con la evolución del porcentaje de error de manera gráfica.

Figura 72

Gráfico de la evolución del loss del optimizador.

```
dfLoss.plot(ylim=(0,0.3))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff8bd858b50>
```



Debemos buscar el error más pequeño, en este caso es el Adragrad, procederemos a analizar los valores del accuracy.

Figura 73

Valores de accuracy de optimizador.

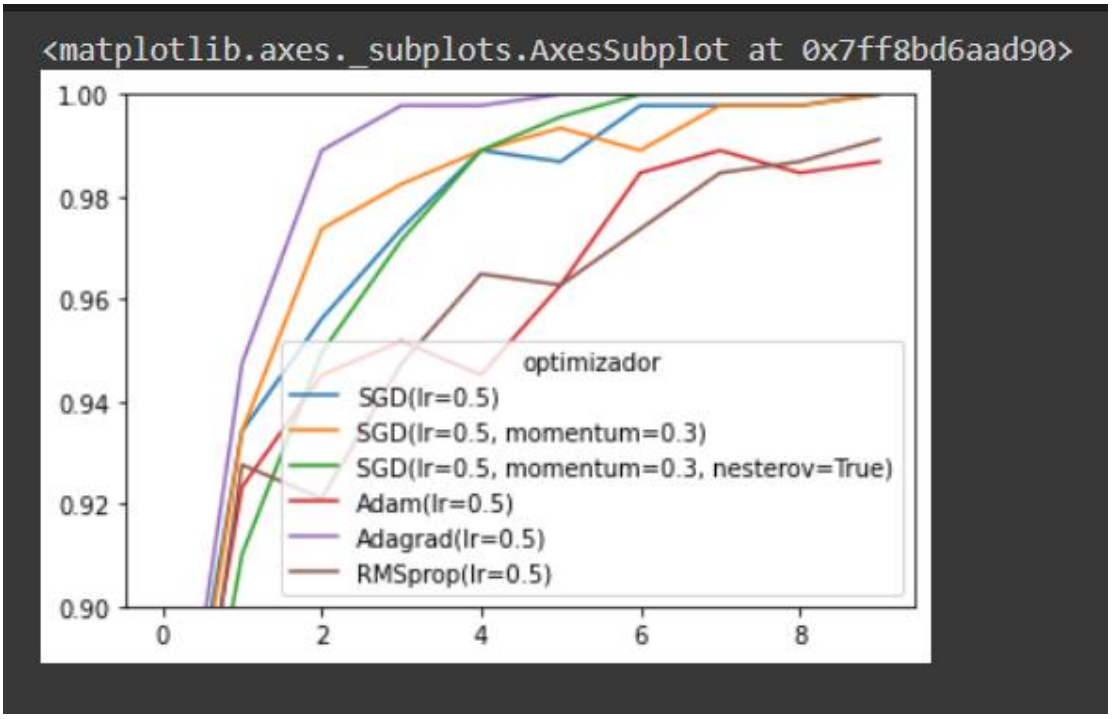
```
#Obtenemos la evolución de los "accuracy"
dfAccuracy
```

optimizador	SGD(lr=0.5)	SGD(lr=0.5, momentum=0.3)	SGD(lr=0.5, momentum=0.3, nesterov=True)	Adam(lr=0.5)	Adagrad(lr=0.5)	RMSprop(lr=0.5)
0	0.844298	0.835526	0.824561	0.833333	0.842105	0.822368
1	0.934211	0.934211	0.910088	0.923246	0.947368	0.927632
2	0.956140	0.973684	0.949561	0.945175	0.989035	0.921053
3	0.973684	0.982456	0.971491	0.951754	0.997807	0.947368
4	0.989035	0.989035	0.989035	0.945175	0.997807	0.964912
5	0.986842	0.993421	0.995614	0.962719	1.000000	0.962719
6	0.997807	0.989035	1.000000	0.984649	1.000000	0.973684
7	0.997807	0.997807	1.000000	0.989035	1.000000	0.984649
8	0.997807	0.997807	1.000000	0.984649	1.000000	0.986842
9	1.000000	1.000000	1.000000	0.986842	1.000000	0.991228

Vemos su grafica de valores

Figura 74

Gráfico de la evolución del accuracy del optimizador.



Vemos que el optimizador Adragrad es el más óptimo para aplicar al modelo.

4.5. EVALUACION DEL MODELO DE REDES NEURONALES

Para elegir el modelo de redes neuronales a utilizar, debemos identificar la capacidad de predicción del modelo, por esto hemos realizado la comparación de diferentes algoritmos con los datos de entrenamiento y prueba, estos resultados se muestran a continuación:

4.5.1. Modelo con 5 capas

Al igual que en las diferentes calibraciones realizadas anteriormente, aquí veremos el desempeño del modelo del tipo red multicapa, con 5 capas, este modelo tiene una capa de entrada con 25 neutrones, 3 capas ocultas con 26 neutrones y una capa de salida con 3 neutrones, a continuación, veremos los resultados de este entrenamiento.

Figura 75

Código de modelo I.

```
#Definimos el modelo
model = Sequential()
model.add(Dense(25, input_shape=(109,), activation='tanh'))
model.add(Dense(13, activation='tanh'))
model.add(Dense(8, activation='tanh'))
model.add(Dense(5, activation='tanh'))
model.add(Dense(3, activation='softmax'))
```

Como resumen vemos el desempeño del modelo en las diferentes capas, aquí se ve cómo evolucionan los parámetros, el número de parámetros en las diferentes capas son el número de nodos de la capa anterior multiplicado por el número de nodos de la capa actual, tenemos que en total aquí se procesa 3263 parámetros.

Figura 76

Descripción del modelo I.

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 25)                2750

 dense_1 (Dense)             (None, 13)                338

 dense_2 (Dense)             (None, 8)                 112

 dense_3 (Dense)             (None, 5)                 45

 dense_4 (Dense)             (None, 3)                 18

=================================================================
Total params: 3,263
Trainable params: 3,263
Non-trainable params: 0
```

Compilaremos todos los modelos con los datos anteriormente calibrados.

```
#Compilamos el modelo con el datos calibrados
model.compile(
  loss='categorical_crossentropy',
  optimizer=Adagrad(lr=0.5),
  metrics=['accuracy']
)
```

Para aprecia los valores de predicción que se tiene del modelo, exportamos los resultados en un dataframe que nos permitirá ver de forma gráfica los valores de error y predicción máximos para el modelo.

Figura 77

Valores de entrenamiento del modelo I.

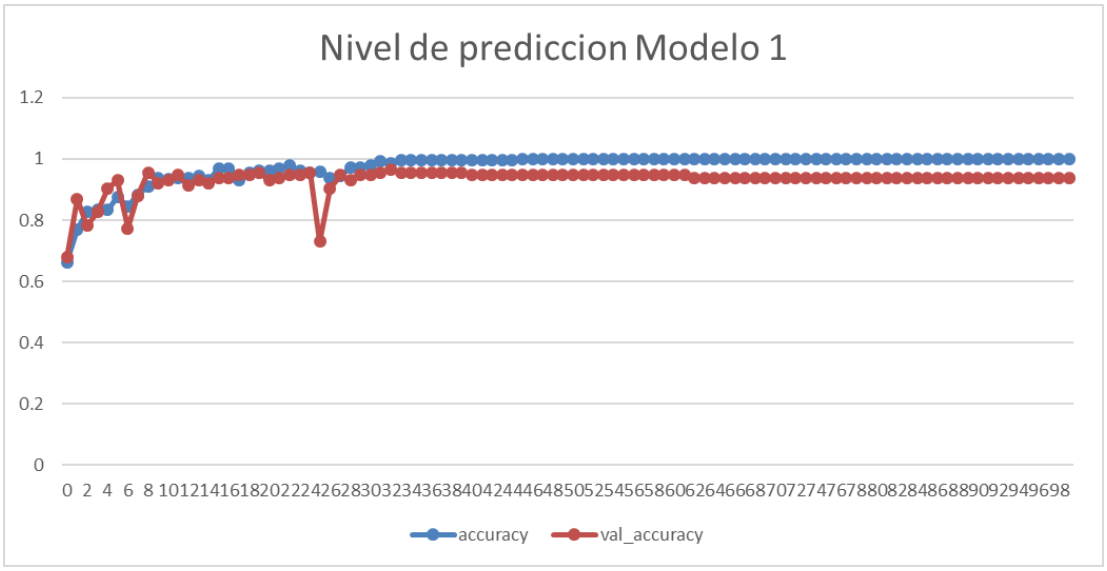
```
#Colocamos los datos en un dataframe de pandas
#El dataframe tendrá dos variables: accuracy y loss, divididos en datos de entrenamiento y prueba
dfDescripcion = pd.DataFrame(
    descripcion_de_entrenamiento.history,
    index = descripcion_de_entrenamiento.epoch
)

dfDescripcion
```

	loss	accuracy	val_loss	val_accuracy
0	0.763238	0.662281	0.635203	0.678261
1	0.521139	0.769737	0.298863	0.869565
2	0.432041	0.826754	0.559617	0.782609
3	0.346712	0.835526	0.362251	0.826087
4	0.320545	0.833333	0.220243	0.904348
...	...	...	...	...
95	0.000508	1.000000	0.325538	0.939130
96	0.000500	1.000000	0.326547	0.939130
97	0.000493	1.000000	0.327537	0.939130
98	0.000486	1.000000	0.328525	0.939130
99	0.000480	1.000000	0.329503	0.939130

Primero veremos el resultado de la predicción, en la iteración 31 se llega al máximo nivel de predicción con la data de prueba con un 95.61% y con una predicción de entrenamiento de 99.12%.

Figura 78

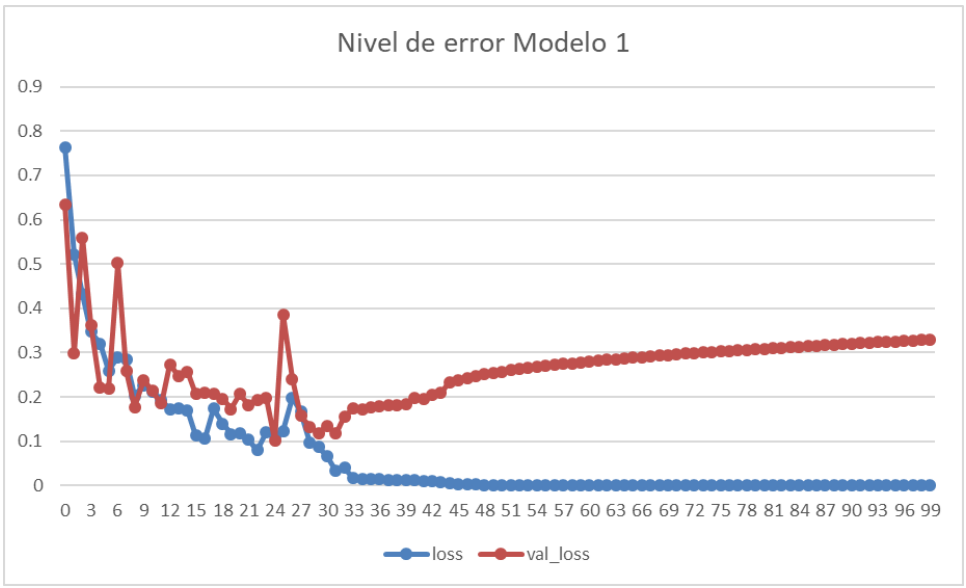
Gráfico de nivel de predicción del modelo 1.



Segundo visualizamos el resultado del error, en la iteración 24 se llega al mínimo nivel de error con la data de prueba con un 10.19 % y con un error de entrenamiento de 11.13 %.

Figura 79

Gráfico de nivel de error del modelo 1.



4.5.2. Modelo con 5 capas II

Aquí veremos el desempeño del modelo del tipo red multicapa, con 5 capas versión II, este modelo tiene una capa de entrada con 50 neutrones, 3 capas ocultas con 40 neutrones y una capa de salida con 3 neutrones, a continuación, veremos los resultados de este entrenamiento.

Figura 80

Código de modelo II.

```
#Definimos el modelo
model = Sequential()
model.add(Dense(50, input_shape=(109,), activation='tanh'))
model.add(Dense(25, activation='tanh'))
model.add(Dense(10, activation='tanh'))
model.add(Dense(5, activation='tanh'))
model.add(Dense(3, activation='softmax'))
```

Como resumen del modelo vemos el desempeño del modelo en las diferentes capas, tenemos que en total procesa 7108 parámetros.

Figura 81

Descripción del modelo II.

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                5500

 dense_1 (Dense)             (None, 25)                1275

 dense_2 (Dense)             (None, 10)                260

 dense_3 (Dense)             (None, 5)                 55

 dense_4 (Dense)             (None, 3)                 18

=================================================================
Total params: 7,108
Trainable params: 7,108
Non-trainable params: 0
_________________________________________________________________
```

Para apreciar los valores de predicción que se tiene del modelo, exportamos los resultados en un dataframe.

Figura 82

Valores de entrenamiento del modelo II.

```
dfDescripcion = pd.DataFrame(
    descripcion_de_entrenamiento.history,
    index = descripcion_de_entrenamiento.epoch
)

dfDescripcion
```

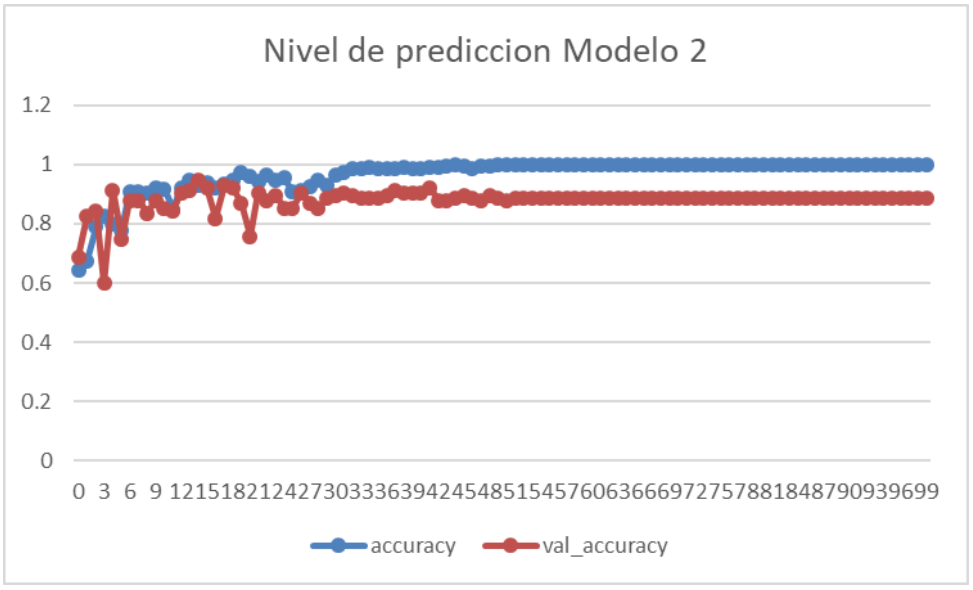
	loss	accuracy	val_loss	val_accuracy
0	0.814716	0.644737	0.639059	0.686957
1	0.616694	0.675439	0.391795	0.826087
2	0.459848	0.789474	0.391388	0.843478
3	0.426750	0.824561	0.856776	0.600000
4	0.455075	0.798246	0.352636	0.913043
...	...	...	...	...
95	0.001630	1.000000	0.667595	0.886957
96	0.001604	1.000000	0.669238	0.886957
97	0.001578	1.000000	0.670840	0.886957
98	0.001554	1.000000	0.672206	0.886957
99	0.001530	1.000000	0.673774	0.886957

100 rows × 4 columns

Primero veremos el resultado de la predicción, en la iteración 14 se llega al máximo nivel de predicción con la data de prueba con un 94.78% y con una predicción de entrenamiento de 92.76 %.

Figura 83

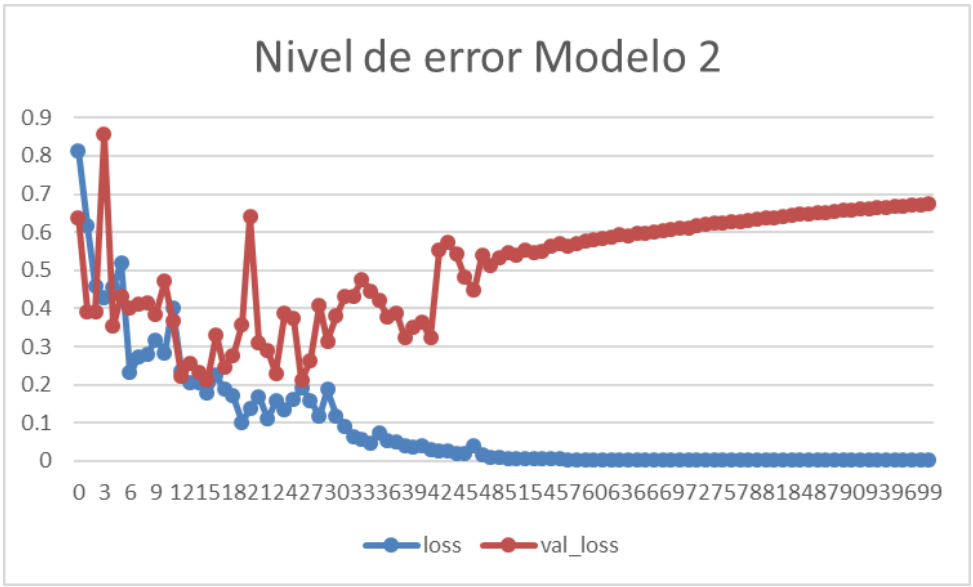
Gráfico de nivel de predicción del modelo 2.



Segundo visualizamos el resultado del error, en la iteración 26 se llega al mínimo nivel de error con la data de prueba con un 21.09 % y con un error de entrenamiento de 19.32 %.

Figura 84

Gráfico de nivel de error del modelo 2.



4.5.3. Modelo con 6 capas

Aquí veremos el desempeño del modelo del tipo red multicapa, con 6 capas, este modelo tiene una capa de entrada con 50 neutrones, 4 capas ocultas con 51 neutrones y una capa de salida con 3 neutrones, a continuación, veremos los resultados de este entrenamiento.

Figura 85

Código de modelo III.

```
#Definimos el modelo
model = Sequential()
model.add(Dense(50, input_shape=(109,), activation='tanh'))
model.add(Dense(25, activation='tanh'))
model.add(Dense(13, activation='tanh'))
model.add(Dense(8, activation='tanh'))
model.add(Dense(5, activation='tanh'))
model.add(Dense(3, activation='softmax'))
```

Como resumen del modelo vemos el desempeño del modelo en las diferentes capas, tenemos que en total procesa 7288 parámetros.

Figura 86

Descripción del modelo III.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 50)                5500

 dense_6 (Dense)             (None, 25)                1275

 dense_7 (Dense)             (None, 13)                338

 dense_8 (Dense)             (None, 8)                 112

 dense_9 (Dense)             (None, 5)                 45

 dense_10 (Dense)            (None, 3)                 18

=================================================================
Total params: 7,288
Trainable params: 7,288
Non-trainable params: 0
_________________________________________________________________
```

Para apreciar los valores de predicción que se tiene del modelo, exportamos los resultados en un dataframe.

Figura 87

Valores de entrenamiento del modelo III.

```
dfDescripcion = pd.DataFrame(
    descripcion_de_entrenamiento.history,
    index = descripcion_de_entrenamiento.epoch
)

dfDescripcion
```

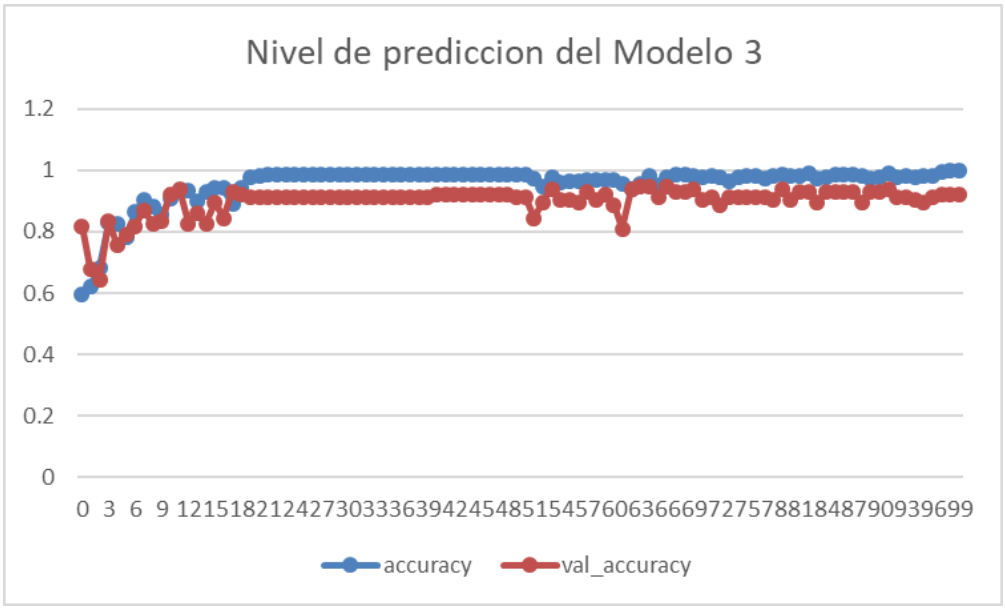
	loss	accuracy	val_loss	val_accuracy
0	0.957784	0.596491	0.663094	0.817391
1	0.782692	0.622807	0.607934	0.678261
2	0.654839	0.684211	0.646761	0.643478
3	0.511240	0.828947	0.464489	0.834783
4	0.436101	0.826754	0.622681	0.756522
...	...	...	...	...
95	0.049545	0.982456	0.338209	0.895652
96	0.041885	0.980263	0.408459	0.913043
97	0.028187	0.993421	0.409262	0.921739
98	0.019490	0.997807	0.384496	0.921739
99	0.016108	0.997807	0.388606	0.921739

100 rows × 4 columns

Primero veremos el resultado de la predicción, en la iteración 64 se llega al máximo nivel de predicción con la data de prueba con un 94.78% y con una predicción de entrenamiento de 98.24 %.

Figura 88

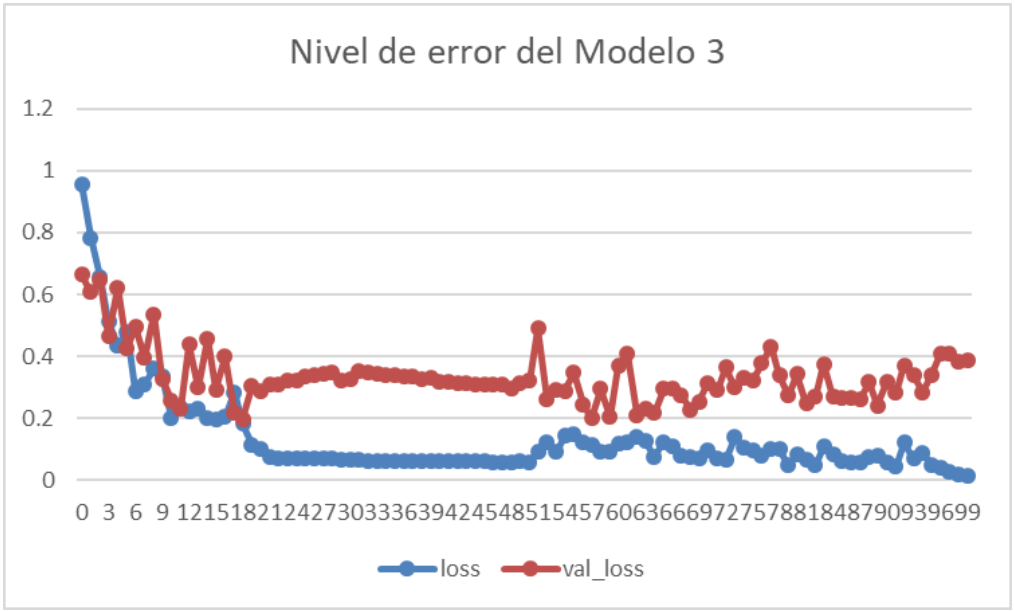
Gráfico de nivel de predicción del modelo 3.



Segundo visualizamos el resultado del error, en la iteración 26 se llega al mínimo nivel de error con la data de prueba con un 21.09 % y con un error de entrenamiento de 19.32 %.

Figura 89

Gráfico de nivel de error del modelo 3.



4.5.4. Modelo con 5 capas III

Aquí veremos el desempeño del modelo del tipo red multicapa, con 5 capas versión III, este modelo tiene una capa de entrada con 30 neutrones, 3 capas ocultas con 35 neutrones y una capa de salida con 3 neutrones, a continuación, veremos los resultados de este entrenamiento.

Figura 90

Código de modelo IV.

```
#Definimos el modelo
model = Sequential()
model.add(Dense(30, input_shape=(109,), activation='tanh'))
model.add(Dense(20, activation='tanh'))
model.add(Dense(10, activation='tanh'))
model.add(Dense(5, activation='tanh'))
model.add(Dense(3, activation='softmax'))
```

Como resumen del modelo vemos el desempeño del modelo en las diferentes capas, tenemos que en total procesa 4203 parámetros.

Figura 91

Descripción del modelo IV.

```
Model: "sequential_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_91 (Dense)            (None, 30)                3300

 dense_92 (Dense)            (None, 20)                620

 dense_93 (Dense)            (None, 10)                210

 dense_94 (Dense)            (None, 5)                 55

 dense_95 (Dense)            (None, 3)                 18

=================================================================
Total params: 4,203
Trainable params: 4,203
Non-trainable params: 0
_________________________________________________________________
```

Para apreciar los valores de predicción que se tiene del modelo, exportamos los resultados en un dataframe.

Figura 92

Valores de entrenamiento del modelo IV.

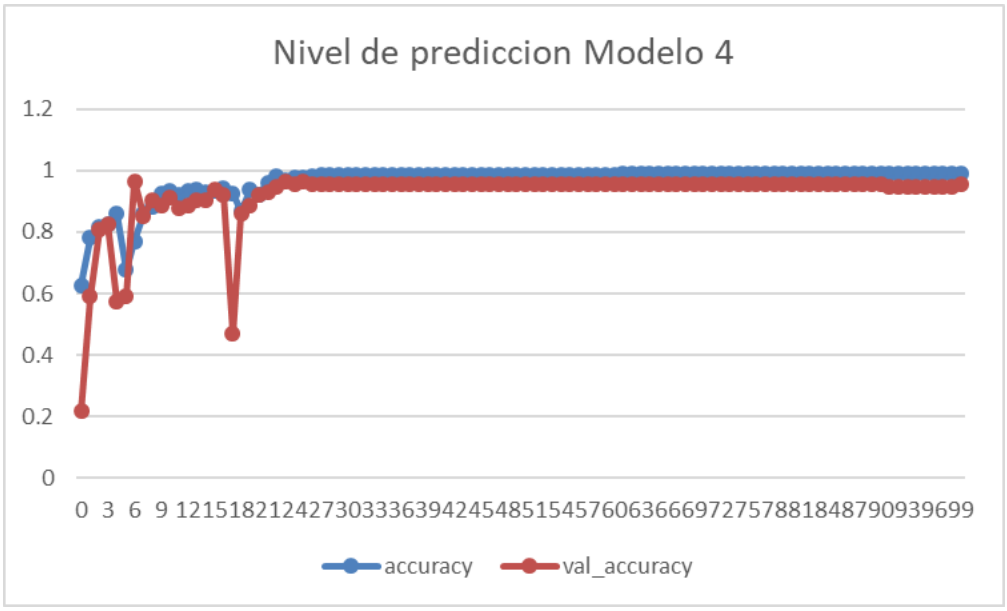
	loss	accuracy	val_loss	val_accuracy
0	0.803773	0.627193	0.771824	0.217391
1	0.557496	0.780702	0.994056	0.591304
2	0.441752	0.817982	0.678291	0.808696
3	0.409847	0.824561	0.381297	0.826087
4	0.335931	0.859649	0.685016	0.573913
...	...	...	...	...
95	0.034899	0.989035	0.244256	0.947826
96	0.034024	0.989035	0.242448	0.947826
97	0.033468	0.989035	0.238393	0.947826
98	0.033087	0.989035	0.236010	0.947826
99	0.032505	0.989035	0.232900	0.956522

100 rows × 4 columns

Primero veremos el resultado de la predicción, en la iteración 25 se llega al máximo nivel de predicción con la data de prueba con un 96.52% y con una predicción de entrenamiento de 97.80 %.

Figura 93

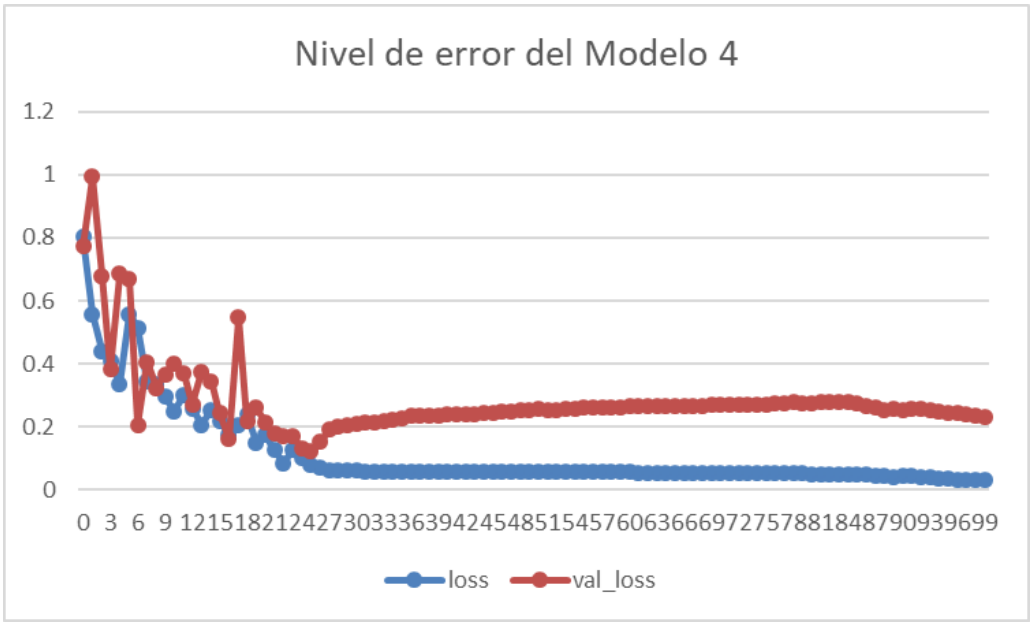
Gráfico de nivel de predicción del modelo 4.



Segundo visualizamos el resultado del error, en la iteración 25 se llega al mínimo nivel de error con la data de prueba con un 12.45 % y con un error de entrenamiento de 8.00 %.

Figura 94

Gráfico de nivel de error del modelo 4.



Los resultados completos de las iteraciones de los modelos presentados están en el anexo C.

El modelo 4 es el modelo elegido para ser utilizado en la investigación, esto por el buen nivel de resultados tanto en el nivel de predicción como en el nivel de error.

4.6. ARQUITECTURA DE RED NEURONAL CON CLASIFICACION MULTICLASE

La arquitectura o topología de la red define el número de capas y el número de neuronas que esta tiene en cada capa, además también se ve cómo interactúan estas entre sí.

La arquitectura de la red neuronal que usamos está compuesta por una capa de entrada, tres capas ocultas y una capa de salida.

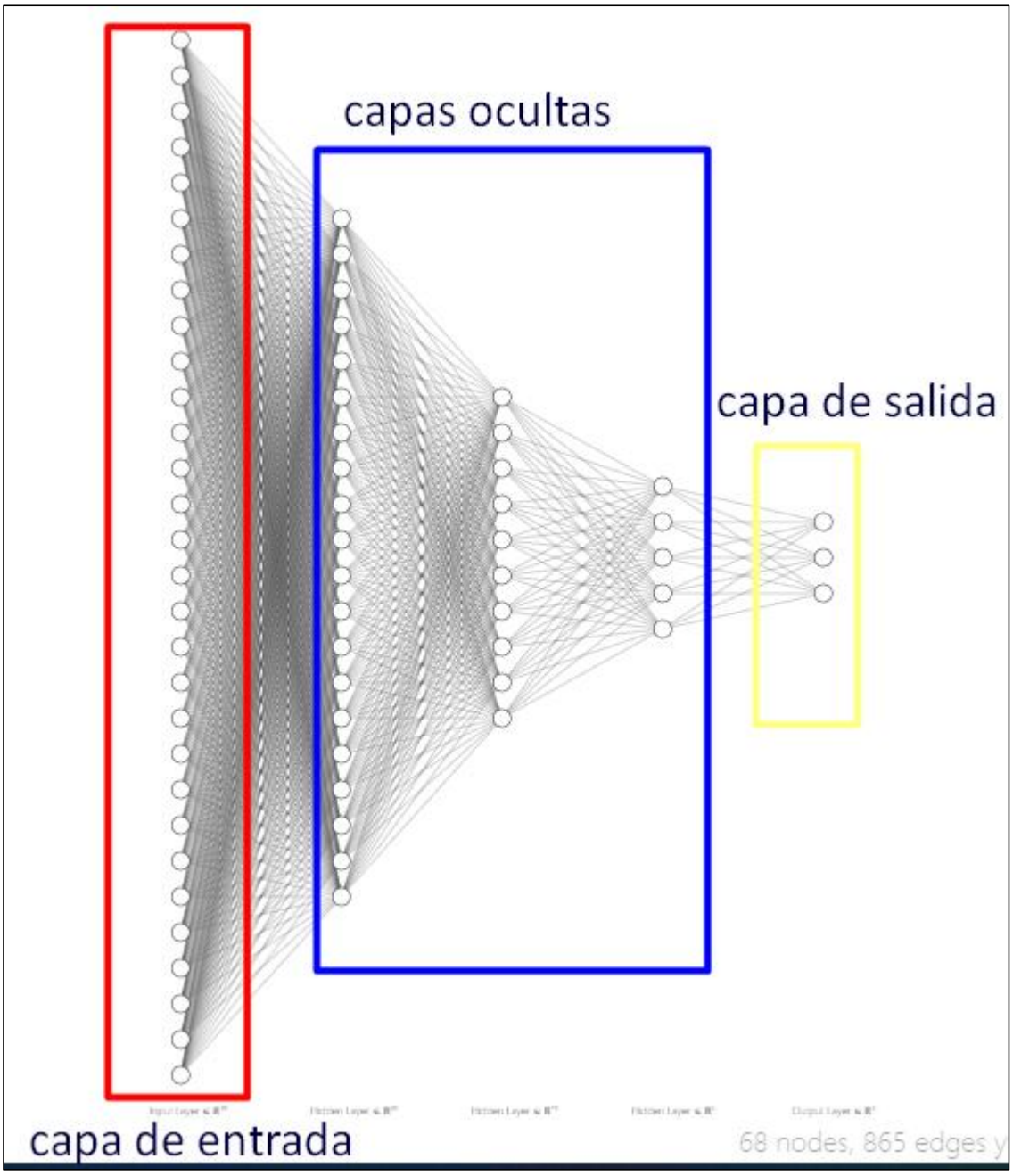
La primera capa está compuesta por 30 neutrones que reciben directamente los datos de las variables de los features.

Las tres capas ocultas reciben los datos y los procesan, generando una salida, utilizan para esto una función matemática, llamada función de activación, para el caso de nuestro modelo de red neuronal se utilizó la función de activación Tangente hiperbólica (Tanh).

La última capa o capa de salida está compuesta por 3 neutrones, esto por el número de valores del label que es el tipo de estrés a determinar, esta capa tiene una función de activación softmax, la arquitectura de la red neuronal se vería como en la figura 46

Figura 95

Arquitectura del modelo final de redes neuronales profundas.



4.7. VALIDACION DEL MODELO DE RED NEURONAL

Una vez ya definidos los valores del rango de aprendizaje, tamaño de batch, optimizador y evaluar los mejores valores para el modelo, podemos proceder a entrenar al algoritmo.

Primero realizaremos la carga de data, y la separación de datos en entrenamiento y validación, definimos un 80% de los datos para el entrenamiento, y ponemos un 20% de los datos para las pruebas.

Figura 96

Código de separación de data.

```
#Dividimos los datos en datos de entrenamiento (x_train, y_train) y datos de validación (x_test, y_test)
x_train, x_test, y_train, y_test = train_test_split(x, y_categorico, test_size = 0.2)
```

Ejecutaremos nuestro modelo de redes neuronales profundas, ya anteriormente validado, definimos el input_shape con un valor de 109, que es el valor de columnas con los diversos valores de las features.

Figura 97

Código del modelo final.

```
model = Sequential()
model.add(Dense(30, input_shape=(109,), activation='tanh'))
model.add(Dense(20, activation='tanh'))
model.add(Dense(10, activation='tanh'))
model.add(Dense(5, activation='tanh'))
model.add(Dense(3, activation='softmax'))
```

Procedemos a compilar el modelo con el optimizador calibrado, óptimo para nuestros datos.

Figura 98

Código de optimizador calibrado.

```
#Compilamos el modelo con el optimizador calibrado
model.compile(
  loss='categorical_crossentropy',
  optimizer=Adagrad(lr=0.5),
  metrics=['accuracy']
)
```

Entrenamos al modelo con la data de entrenamiento, con 30 iteraciones y con el batch óptimo igual a cuatro, validado ya para nuestro tipo de data.

Figura 99

Iteraciones de entrenamiento del modelo final.

```
#Entrenamos el modelo enviándole los features (x_train) y labels (y_train) de entrenamiento, con el batch calibrado
model.fit(x_train, y_train, epochs=30, batch_size=4)
```

```
Epoch 1/30
114/114 [==============================] - 0s 990us/step - loss: 0.8565 - accuracy: 0.6689
Epoch 2/30
114/114 [==============================] - 0s 1ms/step - loss: 0.7194 - accuracy: 0.6579
Epoch 3/30
114/114 [==============================] - 0s 1ms/step - loss: 0.5224 - accuracy: 0.7719
Epoch 4/30
114/114 [==============================] - 0s 969us/step - loss: 0.4284 - accuracy: 0.7873
Epoch 5/30
114/114 [==============================] - 0s 1ms/step - loss: 0.3665 - accuracy: 0.8794
Epoch 6/30
114/114 [==============================] - 0s 1ms/step - loss: 0.2787 - accuracy: 0.8947
Epoch 7/30
114/114 [==============================] - 0s 1ms/step - loss: 0.2272 - accuracy: 0.9035
Epoch 8/30
114/114 [==============================] - 0s 1ms/step - loss: 0.2537 - accuracy: 0.9013
Epoch 9/30
114/114 [==============================] - 0s 1ms/step - loss: 0.3177 - accuracy: 0.8882
Epoch 10/30
114/114 [==============================] - 0s 1ms/step - loss: 0.2441 - accuracy: 0.9035
Epoch 11/30
114/114 [==============================] - 0s 1ms/step - loss: 0.1575 - accuracy: 0.9364
Epoch 12/30
114/114 [==============================] - 0s 1ms/step - loss: 0.1515 - accuracy: 0.9452
Epoch 13/30
114/114 [==============================] - 0s 1ms/step - loss: 0.1294 - accuracy: 0.9518
Epoch 14/30
114/114 [==============================] - 0s 1ms/step - loss: 0.1015 - accuracy: 0.9605
Epoch 18/30
114/114 [==============================] - 0s 1ms/step - loss: 0.1057 - accuracy: 0.9583
Epoch 19/30
114/114 [==============================] - 0s 1ms/step - loss: 0.1349 - accuracy: 0.9496
Epoch 20/30
114/114 [==============================] - 0s 1ms/step - loss: 0.1062 - accuracy: 0.9649
Epoch 21/30
114/114 [==============================] - 0s 1ms/step - loss: 0.0726 - accuracy: 0.9781
Epoch 22/30
114/114 [==============================] - 0s 989us/step - loss: 0.0463 - accuracy: 0.9890
Epoch 23/30
114/114 [==============================] - 0s 1ms/step - loss: 0.0380 - accuracy: 0.9912
Epoch 24/30
114/114 [==============================] - 0s 1ms/step - loss: 0.0431 - accuracy: 0.9912
Epoch 25/30
114/114 [==============================] - 0s 1ms/step - loss: 0.0569 - accuracy: 0.9803
Epoch 26/30
114/114 [==============================] - 0s 1ms/step - loss: 0.0570 - accuracy: 0.9737
Epoch 27/30
114/114 [==============================] - 0s 1ms/step - loss: 0.0422 - accuracy: 0.9846
Epoch 28/30
114/114 [==============================] - 0s 1ms/step - loss: 0.0617 - accuracy: 0.9781
Epoch 29/30
114/114 [==============================] - 0s 1ms/step - loss: 0.0453 - accuracy: 0.9846
Epoch 30/30
114/114 [==============================] - 0s 1ms/step - loss: 0.0302 - accuracy: 0.9912
<keras.callbacks.History at 0x7fc58af7f790>
```

Vemos que el valor máximo de predicción que llega es del 99.12% con un 3 % de error, siendo estos valores óptimos.

Para evaluar el modelo, utilizaremos los datos de testing para ver el porcentaje de aciertos del modelo, lo veremos a través de la matriz de valores, donde nos mostrara el nivel de probabilidad para cada categoría.

Figura 100

Array de predicción.

```
y_prediccion = model.predict(x_test)
y_prediccion.round(3)
```

```
array([[0.001, 0.999, 0.   ],
       [0.001, 0.999, 0.   ],
       [0.001, 0.999, 0.   ],
       [0.001, 0.999, 0.   ],
       [0.001, 0.999, 0.   ],
       [0.001, 0.999, 0.   ],
       [0.995, 0.005, 0.   ],
       [0.995, 0.005, 0.   ],
       [0.001, 0.999, 0.   ],
       [0.001, 0.017, 0.982],
       [0.001, 0.999, 0.   ],
       [0.001, 0.017, 0.982],
       [0.001, 0.87 , 0.13 ],
       [0.001, 0.999, 0.   ],
       [0.995, 0.005, 0.   ],
       [0.001, 0.999, 0.   ],
```

Para ver el valor de la categoría elegiremos el valor máximo de la predicción.

Figura 101

Array de valor de predicción.

```
y_prediccion_categorico = np.argmax(y_prediccion, axis = 1)
y_prediccion_categorico
```

```
array([1, 1, 1, 1, 1, 1, 0, 0, 1, 2, 1, 2, 1, 1, 0, 1, 0, 2, 2, 0, 2, 0,
       0, 1, 1, 2, 1, 2, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
       1, 2, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 2,
       0, 1, 1, 1, 0, 1, 2, 0, 1, 1, 1, 0, 1, 1, 2, 1, 1, 1, 0, 0, 0, 1,
       1, 1, 1, 2, 0, 0, 1, 1, 2, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       0, 1, 1, 0, 1])
```

Como se tiene los valores de y_test procederemos a llamarlos.

Figura 102

Valores de Y reales.

```
#obtenemos los Y reales para testing
y_test
```

```
array([[0., 1., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 1., 0.],
```

Categorizamos los valores de y_test para poder compararlos con los valores con los valores obtenidos en y_prediccion.

Figura 103

Valores de Y de prueba.

```
#Categorizamos la data de test
y_test_categorico = np.argmax(y_test, axis = 1)
y_test_categorico
```

```
array([1, 1, 1, 1, 1, 1, 1, 0, 1, 2, 1, 2, 2, 1, 0, 1, 0, 2, 2, 0, 2, 0,
       0, 1, 2, 2, 1, 2, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
       1, 2, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 2,
       0, 1, 1, 1, 0, 1, 2, 0, 1, 0, 1, 0, 1, 1, 2, 1, 1, 1, 0, 0, 0, 1,
       1, 1, 1, 2, 0, 0, 1, 1, 2, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 1])
```

La matriz de confusión nos mostrara el nivel de acierto según cada categoría.

```
#Imprimimos la matriz de confusión
confusion_matrix(y_test_categorico, y_prediccion_categorico)
```

```
array([[34,  2,  0],
       [ 2, 62,  0],
       [ 0,  2, 13]])
```

Llevaremos esta matriz a un dataframe que nos haga más fácil su análisis.

Figura 104

Valores de matriz de confusión.

```
#Imprimos la matriz de confusión
pintar_matriz_de_confusion(y_test_categorico, y_prediccion_categorico, ['Estres_simple', 'Estres_moderado'
```

	PREDICCION Estres_simple	PREDICCION Estres_moderado	PREDICCION Estres_severo
Estres_simple	34	2	0
Estres_moderado	2	62	0
Estres_severo	0	2	13

Analizando la matriz vemos que tiene un acierto del 93.1 % con un rango de error de 6,9 %, vemos que para la categoría de estrés simple detecto correctamente 34 casos y se equivocó en 2, en la categoría de estrés moderado detecto correctamente 62 casos y se equivocó en 2 confundiéndolos con estrés simple y por último en la categoría de estrés severo detecto correctamente 13 casos y se equivocó en 2 confundiéndolos con estrés moderado.

Finalmente, a partir de los resultados obtenidos al probar el modelo de redes neuronales profundas realizado para determinar el nivel de estrés en estudiantes universitarios, de un total de 576 muestras, se obtuvo que el modelo tiene un 96.52 % de predicción con un promedio de error de 6 %.

Procederemos a guardar el modelo, para poder así empezar a utilizarlo.

```
#Almacenamos el modelo entrenado
model.save('/content/drive/MyDrive/Data/Modelo_Estres.h5')
```

Almacenaremos también la metada que necesitamos.

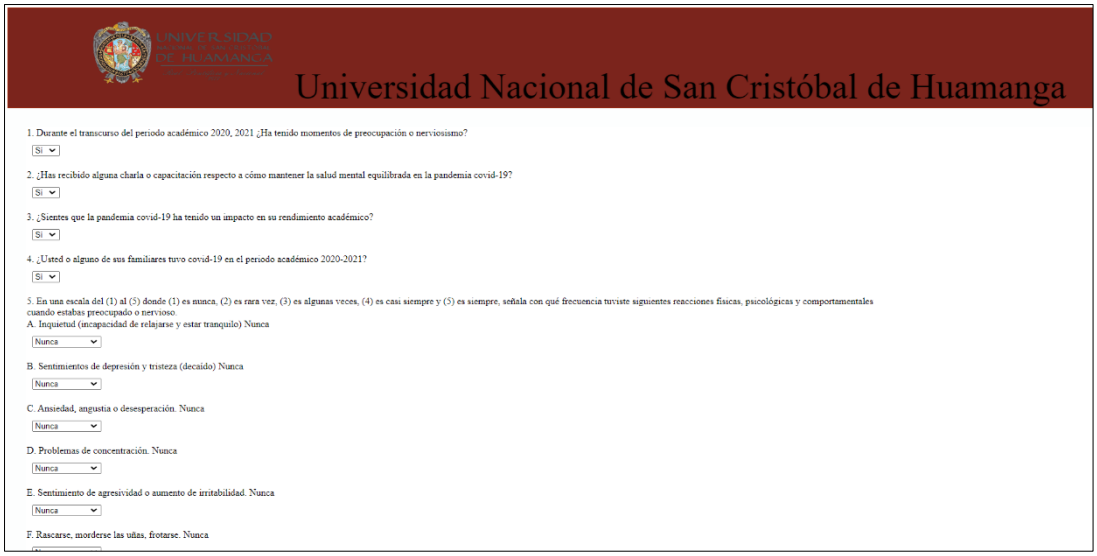
```
#Almacenamos el archivo
np.savez('/content/drive/MyDrive/Data/metadata', dataset_numpy)
```

Obtenido estos archivos, podemos proceder a aplicar nuestro modelo, esto se puede realizar tanto para predicciones individuales como para predicciones a grandes cantidades

de registro, para validar esto se implementó una página web temporal que podría integrarse en la página de la UNSCH. En esta página se responde a un cuestionario y ya el alumno podrá saber el nivel de estrés que tiene.

Figura 105

Página Web donde esta implementado el modelo de la red neuronal profunda.



CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

De acuerdo a la investigación realizada concluimos que, para realizar el entrenamiento de modelos con redes neuronales profundas, se realizaron diferentes pruebas, al calibrar cada uno de sus elementos se realizó la comparación de resultados, para elegir el más óptimo y así lograr un mayor rango de predicción, es ahí donde se validó el nivel de acierto de la investigación.

Al comenzar una investigación donde se desea aplicar redes neuronales, se debe tomar en cuenta el tipo de variables de las features y labels, ya que existen diferentes tipos de algoritmos que están direccionados al tipo de variable que se tiene.

Para diseñar la arquitectura de la red neuronal profunda, se tuvo en cuenta que el número de neutrones de la capa de salida debe ser el mismo que el número de variables del label, en cambio el número de neutrones en las capas ocultas y de entrada se deben ir probando a manera de prueba y error.

Para realizar el diagnostico de estrés estudiantil con redes neuronales profundas, ha sido necesario crear variables dummy por el valor categórico de las diferentes variables de los features, esto fue necesario para que la red funcionara correctamente.

En cuanto al nivel de predicción alcanzado por el modelo, este es variable, pero en todas las pruebas realizadas a este, su nivel de predicción no baja del 95,5%, mientras que el error máximo al cual llego este modelo es de 7%, lo cual nos señala que es un modelo optimo que puede productivizarse.

El proceso de diagnosticar el nivel de estrés de un estudiante universitario, es tardado, pero con la aplicación del modelo, este proceso se hace de forma inmediata.

Se puede mejorar el resultado de la investigación, esto quiere decir ampliar el nivel de predicción, pero para esto se necesitaría más data de entrenamiento para el modelo y volver a hacer la calibración de los elementos de la red neuronal.

5.2. RECOMENDACIONES

Se recomienda que antes de empezar cualquier entrenamiento de modelos de redes neuronales, se valide la calidad de la data, así como el tipo de carga de cada variable, esto con el fin de asegurar el correcto entrenamiento de la red neuronal.

Esta investigación utilizo como plataforma de desarrollo el Google Colab, se recomienda que, si se desea llevar este proyecto a mayor escala, ósea implementar un modelo con mayor rango de entrenamiento, se necesitara más data, por lo cual se recomienda usar una plataforma cloud de paga, la cual hará que el entrenamiento del modelo demore menos tiempo de ejecución.

Se recomienda realizar el calibrado de cada uno de los elementos de la red neuronal cada 3 meses, para asegurar el correcto funcionamiento de este, este modelo está basado en un entorno covid-19, el cual actualmente vivimos, pero este entorno puede variar, por lo cual, al realizar la calibración, se debe a volver a realizar la feature engineering, en la cual se hace una evaluación de las variables a estudiar.

Se recomienda que, para hacer un seguimiento al nivel de salud mental del alumnado de la UNSCH, el área de bienestar estudiantil podría implementar la realización de esta encuesta desde una página web, con los datos obtenidas de esta encuesta se podría lanzar campañas de salud mental por escuelas o individuales.

Como se mencionó en el capítulo anterior, la web creada para el uso del modelo es temporal, por eso se recomienda anexarla a una página de la UNSCH ya existente, si se desea utilizar.

REFERENCIAS BIBLIOGRAFICAS

Aggarwal, C. (2018). *Neural Networks and Deep Learning.* Primera edición. Springer International Publishing AG.

Andrade, R., Bartocci, L., da Silva, I., dos Reis Alves, S., y Hernane, D. (2017). *Artificial Neural Networks, A Practical Course.* Primera edición. Springer International Publishing Switzerland.

American Psychological Association. (2010). *Los distintos tipos de Estrés*. https://www.apa.org/topics/stress/tipos

Anguiano, E. (2018). *Redes neuronales profundas en energías renovables*. Trabajo fin de grado. Universidad Autónoma de Madrid. Madrid, España.

Barraza, A. (2006). Un modelo conceptual para el estudio del estrés académico. *Revista Electrónica de Psicología Iztacala Vol. 9 No. 3*.

Barraza, A. (2018). *INVENTARIO SISCO SV-21. Inventario Sistémico Cognoscitivista para el estudio del estrés académico. Segunda versión de 21 ítems.* Primera edición. Editorial ECORFAN - México S.C

Bernal, César A. (2010). *Metodología de la Investigación*. Tercera edición. Pearson Educación: Colombia.

Bengio Y, et al. (2016). *Deep learning*. Primera edición, Cambridge, MA : MIT Press

Beysolow, T. (2017). *Introduction to Deep Learning Using R*. Primera edición. Springer Science+Business Media.

Bramer, M. (2016). *Principles of Data Mining*. (3era ed.). Springer. https://doi.org/10.1007/978-1-4471-7307-6-1

Bong, K. (2018). *El estrés y los síntomas somáticos*. Springer. https://doi.org/10.1007/978-3-030-02783-4.

Buduma, N. (2010). *Fundamentals of Deep Learning Designing Next Generation Artificial Intelligence Algorithms*. Primera edición. O'Reilly Media, Inc.

Davy, C., Arno, D., Meysman y Mohamed A. (2016). *Introducing Data Science*. Manning. https://livebook.manning.com/book/introducing-data-science/chapter-2/4

Diario El Comercio (26 mayo, 2020). Diario El Comercio. Del campus a la pantalla: ¿cuál es el impacto del COVID-19 en las universidades del Perú? Recuperado de https://elcomercio.pe/lima/sucesos/del-campus-a-la-pantalla-cual-es-el-impacto-del-covid-19-en-las-universidades-del-peru-coronavirus-sunedu-minedu-noticia/

Dumbill, E., Liddy, E. D., Stanton, J., Mueller, K., & Farnham, S. (2013). *Educating the Next Generation of Data Scientists*. *Big Data*, *1*(1), 21-27. https://doi.org/10.1089/big.2013.1510

Ertel, W. (2016). *Introduction to Artificial Intelligence*. (2da Ed.). Springer. https://doi.org/10.1007/978-3-319-58487-4

Estrada, E., Mamani, M., Gallegos, N., Mamani, H. y Zuloaga, M. (2021). Estrés académico en estudiantes universitarios peruanos en tiempos de la pandemia del COVID-19. *Revista AVFT*, 1-6. http://doi.org/10.5281/zenodo.4675923

Google research, *Google Colaboratory. https://research.google.com/colaboratory/intl/es/faq.html#:~:text=Colaboratory%2C%20o%20%22Colab%22%20para,an%C3%A1lisis%20de%20datos%20y%20educaci%C3%B3n.*

Hernández Sampieri, R., Fernández, C. y Baptista, P. (2014). *Metodología de la investigación* (6ta Ed.). México, D.F., México: McGraw Hill Interamericana.

Huamán, R. (2020). *Estrés académico y miedo a contraer coronavirus en universitarios del primer y último año de la carrera de psicología de una universidad*

privada lima – sur. Tesis de pregrado. Universidad Autónoma del Perú. Lima, Perú. Recuperado de http://repositorio.autonoma.edu.pe/handle/AUTONOMA/1254

IBM, (2015). *Delivery Process: ASUM-DM.* Recuperado de http://i2t.icesi.edu.co/ASUM-DM_External/cognos.external.asum-DM_Teaser

Instituto Nacional de Estadística e Informática. (2021). Informe Técnico: Estadísticas de las Tecnologías de Información y Comunicación en los Hogares. Nº 02- junio 2021. Gobierno de Perú.

Ketkar, N. (2017). *Introduction to Keras. In: Deep Learning with Python*. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-2766-4_7

Kroenke K., Spitzer R. y Williams J. (2001). *The PHQ-9: validity of a brief depression severity measure*. J Gen Intern Med. https://doi.org/10.1046/j.1525-1497.2001.016009606.x.

Ley N.° 30220. Ley Universitaria. (8 de julio de2014). Normas Legales Actualizadas. Diario Oficial El Peruano.

Lee, K (2014). *Python programming fundamentals.* Segunda edición. Springer London Heidelberg New York Dordrecht.

Lineamientos para el cuidado integral de la salud mental en las universidades. LI-001-01-MINEDU. 08 de agosto de 2019 (Perú).

Lineamientos para el cuidado integral de la salud mental en las universidades. RV-277-2020-MINEDU. 22 de diciembre de 2020 (Perú).

López, E. (2019). Introducción al Deep Learning. Argentina. IAAR. Recuperado de https://iaarbook.github.io/deeplearning/

Lovibond, S.H. y Lovibond, P.F. (1995). *Manual for the Depression Anxiety Stress Scales*. (2da Ed.). Austria. Sydney: Psychology Foundation.

Mehta, C. y Patel, N. (1995). *Exact logistic regression: Theory and examples.* Statistics in Medicine, 14(19), 2143–2160.

Nakatamo, Pat (2017). *Neural Networks and Deep Learning: Deep Learning explained to your granny.* Primera edición. Editorial CreateSpace Independent Publishing Platform

Ñaupas, Alba. (16 de febrero de 2021). *Depresión, estrés y ansiedad en el dormitorio: el precio de estudiar en tiempos de pandemia*. SomosPeriodismo. https://somosperiodismo.com/depresion-estres-y-ansiedad-en-el-dormitorio-el-precio-de-estudiar-en-tiempos-de-pandemia/

Organización Panamericana de la Salud. (2020). Directrices de laboratorio para la detección y el diagnóstico de la infección por el virus responsable de la COVID-19. https://iris.paho.org/handle/10665.2/52471

Política de aseguramiento de la calidad de la educación superior universitaria. DS-016-2015-MINEDU. 25 de setiembre de 2015 (Perú).

Qamar, U. y Summair, M.(2020). *Data Science Concepts and Techniques with Applications*. Springer. https://doi.org/10.1007/978-981-15-6133-7

Sanner, M. (1975). *PYTHON: A PROGRAMMING LANGUAGE FOR SOFTWARE INTEGRATION AND DEVELOPMENT*. The Scripps Research Institute.

Vásquez, R., Bongianino, C. y Sosisky, L. (2006). *La tecnología educativa y la educación a distancia.* XVII Jornadas Universitarias de Contabilidad. Universidad Nacional de Entre Ríos. Concepción del Uruguay, Argentina.

ANEXO A

CUESTIONARIO DE ESTRÉS ACADÉMICO EN EL PERIODO DE COVID-19

CUESTIONARIO DE ESTRÉS ACADÉMICO EN EL PERIODO DE COVID-19

El presente cuestionario tiene como objetivo reconocer las características del estrés en los estudiantes universitarios de pre-grado durante el periodo de clases del 2020 y 2021, periodo covid-19. La sinceridad con que respondan a las preguntas será de gran utilidad para la investigación. La información que se proporcione será totalmente confidencial y solo se manejarán resultados globales. La respuesta a este cuestionario es voluntaria por lo que usted está en su derecho de contestarlo o no contestarlo.

¿A qué universidad pertenece?

……..……

¿En qué ciclo o semestre se encuentra actualmente?

……..……

¿En qué semana de clases se encuentra actualmente?

……..……

Durante el transcurso del periodo académico 2020, 2021 ¿Ha tenido momentos de preocupación o nerviosismo?

Si

No

En una escala del (1) al (5), donde (1) es nunca, (2) es rara vez, (3) es algunas veces, (4) es casi siempre y (5) es siempre, señala con qué frecuencia tuviste siguientes reacciones físicas, psicológicas y comportamentales cuando estabas preocupado o nervioso.

REACCIONES	(1) nunca	(2) rara vez	(3) algunas veces	(4) casi siempre	(5) siempre
Inquietud (incapacidad de relajarse y estar tranquilo)					
Sentimientos de depresión y tristeza (decaído)					
Ansiedad, angustia o desesperación.					
Problemas de concentración					
Sentimiento de agresividad o aumento de irritabilidad					
Rascarse, morderse las uñas, frotarse, etc.					
Conflictos o tendencia a polemizar o discutir					
Aislamiento de los demás					
Desgano para realizar las labores educativas					
Trastornos en el sueño (insomnio o pesadillas)					
Fatiga crónica (cansancio permanente)					
Dolores de cabeza o migrañas					
Problemas de digestión, dolor abdominal o diarrea					
Cambios en los horarios de alimentación					
Cambios en los horarios de sueño					
Sobrecarga de trabajos educativos					
No entender los temas que se abordan en la clase					
Somnolencia o mayor necesidad de dormir					

En una escala del (1) al (5) donde (1) es nunca, (2) es rara vez, (3) es algunas veces, (4) es casi siempre y (5) es siempre, señala como te sientes respecto al covid-19.

REACCIONES	(1) Nunca	(2) Rara vez	(3) Algunas veces	(4) Casi siempre	(5) Siempre
Tengo miedo al coronavirus(covid-19)					
Mis manos se ponen húmedas cuando pienso en el coronavirus(covid-19)					
Tengo miedo de perder mi vida a causa del coronavirus(covid-19)					
Cuando veo noticias e historias sobre el coronavirus(covid-19) en redes sociales me pongo nervioso o ansioso.					
Mi corazón se acelera cuando pienso en contagiarme del coronavirus(covid-19).					

¿Has recibido alguna capacitación respecto a cómo mantener la salud mental equilibrada en la pandemia covid-19?

a) Si
b) No

¿Sientes que la pandemia covid-19 ha tenido un impacto en su rendimiento académico?

a) Si
b) No

Si su respuesta fue si, para usted fue un impacto:

a) Positivo
b) Negativo

¿Usted o alguno de sus familiares tuvo covid-19 en el periodo académico 2020-2021?

a) Si
b) No

¿Si su respuesta fue si, Durante el periodo de enfermedad o posterior sintió que las reacciones indicadas en la pregunta número 5 se presentaron con mayor frecuencia?

a) Si
b) No

Durante el periodo de enfermedad o posterior pudo realizar sus labores académicas con normalidad

a) Si
b) No

ANEXO B

DOCUMENTOS PARA VALIDACION DE INSTRUMENTOS

CARTA DE PRESENTACION

Licenciada:

Sharon Villacorta Cabrera

Presente

Asunto: **VALIDACION DE INSTRUMENTOS A TRAVES DE JUICIO DE EXPERTO.**

Es grato dirigirme a usted para expresarle mis saludos y así mismo, hacer de su conocimiento que requiero validar el instrumento con el cual recogeré la información necesaria para poder desarrollar mi investigación, con la cual optaré el Título de Ingeniero de Sistemas.

El título de mi investigación es: **"ESTRÉS ESTUDIANTIL UNIVERSITARIO BASADO EN REDES NEURONALES PROFUNDAS, EN PANDEMIA COVID-19, 2021"** y siendo imprescindible contar con la aprobación de expertos especializados para poder aplicar el instrumento que lleva por título: **"CUESTIONARIO DE ESTRÉS ACADÉMICO EN EL PERIODO DE COVID-19",** he considerado conveniente recurrir a usted, ante su connotada experiencia en docencia de educación superior y en temas de salud con especialidad en psicología.

Los documentos de validación, que le hago llegar contiene:

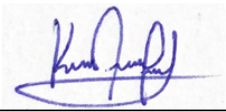
1. Carta de presentación.

2. Constancia de Validación de Instrumento.

3. Cuestionario de estrés académico en el periodo de covid-19

Expresándole mi consideración me despido de usted, no sin antes agradecerle por la atención a la presente.

Atentamente.



Kelly Patricia De la Cruz Oriundo

DNI:47231597

CONSTANCIA DE VALIDACION DEL INSTRUMENTO

CONSTANCIA DE VALIDACIÓN

Yo ...Sharon Sandy Villacorta Cabrera....., con D.N.I.: ...47454626... de profesión .....Psicologa..... Ejerciendo actualmente como ...Psicologa - Asimilado Vat..... en la institución: ...Policlinico Policial - Ayacucho....., hago constar que he revisado, con fines de validación el instrumento de recolección de datos, diseñado por la investigadora Bach. Kelly Patricia De la Cruz Oriundo, y luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones:

	Deficiente	Aceptable	Excelente
Congruencia ítem-dimensión			✓
Amplitud de contenidos			✓
Redacción de los ítems			✓
Ortografía			✓
Presentación			✓

Anotaciones:

Ninguna

Nombre: Sharon Sandy

Código colegiatura: 25015

GS - 403869
Sharon Sandy VILLACORTA CABRERA
CAP. SPNP
JEFE DEL SERVICIO DE PSICOLOGÍA
POLPOL AYACUCHO

Firma del validador

ANEXO C

Tabla 10

Valores predicción y error de los modelos según iteración.

	PORCENTAJE DE PREDICCION								PORCENTAJE DE ERROR							
	Modelo 1		Modelo 2		Modelo 3		Modelo 4		Modelo 1		Modelo 2		Modelo 3		Modelo 4	
N°	accuracy	val_accu	accuracy	val_accu	accuracy	val_accu	accuracy	val_accu	loss	val_loss	loss	val_loss	loss	val_loss	loss	val_loss
0	0.662281	0.678261	0.644737	0.686957	0.596491	0.817391	0.627193	0.217391	0.763238	0.635203	0.814716	0.639059	0.957784	0.663094	0.803773	0.771824
1	0.769737	0.869565	0.675439	0.826087	0.622807	0.678261	0.780702	0.591304	0.521139	0.298863	0.616694	0.391795	0.782692	0.607934	0.557496	0.994056
2	0.826754	0.782609	0.789474	0.843478	0.684211	0.643478	0.817982	0.808696	0.432041	0.559617	0.459848	0.391388	0.654839	0.646761	0.441752	0.678291
3	0.835526	0.826087	0.824561	0.6	0.828947	0.834783	0.824561	0.826087	0.346712	0.362251	0.42675	0.856776	0.51124	0.464489	0.409847	0.381297
4	0.833333	0.904348	0.798246	0.913043	0.826754	0.756522	0.859649	0.573913	0.320545	0.220243	0.455075	0.352636	0.436101	0.622681	0.335931	0.685016
5	0.877193	0.930435	0.778509	0.747826	0.782895	0.791304	0.677632	0.591304	0.259531	0.217693	0.520085	0.432673	0.480378	0.424719	0.558863	0.667607
6	0.844298	0.773913	0.910088	0.878261	0.864035	0.817391	0.769737	0.965217	0.289206	0.502107	0.232203	0.402781	0.289554	0.497851	0.513666	0.206909
7	0.883772	0.878261	0.907895	0.878261	0.905702	0.869565	0.859649	0.852174	0.283406	0.259414	0.273647	0.412936	0.309571	0.395493	0.343347	0.405996
8	0.910088	0.956522	0.905702	0.834783	0.881579	0.826087	0.881579	0.904348	0.203383	0.177631	0.278996	0.415147	0.360926	0.53685	0.334055	0.324289
9	0.936404	0.921739	0.921053	0.878261	0.855263	0.834783	0.925439	0.886957	0.226025	0.238034	0.317961	0.38599	0.3369	0.327256	0.294864	0.368001
10	0.934211	0.930435	0.916667	0.852174	0.910088	0.921739	0.932018	0.913043	0.212557	0.213505	0.283557	0.472084	0.201446	0.259029	0.248889	0.399607
11	0.936404	0.947826	0.848684	0.843478	0.932018	0.93913	0.921053	0.878261	0.192844	0.185188	0.402005	0.36872	0.237726	0.233531	0.301749	0.370006
12	0.938596	0.913043	0.921053	0.904348	0.932018	0.826087	0.934211	0.886957	0.172089	0.273371	0.235413	0.222175	0.221532	0.437776	0.259448	0.271182
13	0.942982	0.930435	0.947368	0.913043	0.899123	0.86087	0.936404	0.904348	0.174256	0.247708	0.20552	0.255796	0.229489	0.300096	0.205985	0.376281
14	0.929825	0.921739	0.927632	0.947826	0.929825	0.826087	0.929825	0.904348	0.170154	0.257451	0.205705	0.233254	0.202317	0.457751	0.255018	0.345299
15	0.969298	0.93913	0.936404	0.921739	0.942982	0.895652	0.934211	0.93913	0.112866	0.20703	0.177668	0.211281	0.197109	0.292838	0.218531	0.24268
16	0.969298	0.93913	0.921053	0.817391	0.940789	0.843478	0.942982	0.921739	0.106941	0.208368	0.225264	0.330978	0.203695	0.399631	0.17587	0.162309
17	0.932018	0.947826	0.934211	0.930435	0.890351	0.930435	0.925439	0.469565	0.173603	0.206235	0.188172	0.244663	0.285116	0.21883	0.206704	0.546407
18	0.953947	0.947826	0.947368	0.921739	0.942982	0.921739	0.868421	0.86087	0.139915	0.196257	0.170949	0.275973	0.182511	0.197581	0.238633	0.218398
19	0.962719	0.956522	0.971491	0.869565	0.97807	0.913043	0.936404	0.886957	0.115829	0.172487	0.101543	0.358551	0.115755	0.304371	0.149767	0.261314
20	0.960526	0.930435	0.960526	0.756522	0.980263	0.913043	0.921053	0.921739	0.117411	0.206465	0.136999	0.642406	0.102142	0.288197	0.173867	0.212901
21	0.969298	0.93913	0.940789	0.904348	0.986842	0.913043	0.960526	0.930435	0.103784	0.182252	0.168942	0.311091	0.074	0.307387	0.127016	0.177554
22	0.97807	0.947826	0.964912	0.878261	0.986842	0.913043	0.982456	0.947826	0.07928	0.192887	0.109495	0.289346	0.07158	0.311358	0.085183	0.171421
23	0.960526	0.947826	0.947368	0.895652	0.986842	0.913043	0.969298	0.965217	0.120683	0.197579	0.156965	0.229774	0.070964	0.32399	0.129591	0.169666
24	0.953947	0.956522	0.953947	0.852174	0.986842	0.913043	0.975877	0.956522	0.111368	0.101908	0.134569	0.386612	0.069988	0.323459	0.10143	0.132157
25	0.958333	0.730435	0.910088	0.852174	0.986842	0.913043	0.97807	0.965217	0.123762	0.385695	0.161299	0.373264	0.069513	0.335915	0.080002	0.124525
26	0.938596	0.904348	0.912281	0.904348	0.986842	0.913043	0.980263	0.956522	0.198457	0.24044	0.193226	0.210922	0.069333	0.338394	0.072679	0.15257
27	0.945175	0.947826	0.925439	0.869565	0.986842	0.913043	0.986842	0.956522	0.167102	0.157685	0.159557	0.264278	0.069176	0.342187	0.062549	0.191454
28	0.971491	0.930435	0.947368	0.852174	0.986842	0.913043	0.986842	0.956522	0.096205	0.132071	0.116722	0.408738	0.069579	0.346578	0.060649	0.199696
29	0.973684	0.947826	0.929825	0.886957	0.986842	0.913043	0.986842	0.956522	0.086621	0.116791	0.187867	0.315231	0.064925	0.320592	0.060998	0.205172
30	0.980263	0.947826	0.962719	0.895652	0.986842	0.913043	0.986842	0.956522	0.066608	0.13473	0.119126	0.380167	0.067953	0.325315	0.060258	0.209956
31	0.991228	0.956522	0.973684	0.904348	0.986842	0.913043	0.986842	0.956522	0.03312	0.117673	0.090871	0.431922	0.065707	0.3544	0.060161	0.213039
32	0.984649	0.965217	0.984649	0.895652	0.986842	0.913043	0.986842	0.956522	0.040098	0.156287	0.063188	0.432468	0.064119	0.347876	0.059705	0.21525
33	0.997807	0.956522	0.984649	0.886957	0.986842	0.913043	0.986842	0.956522	0.016962	0.173805	0.05678	0.474851	0.063652	0.344019	0.05979	0.218556

	PORCENTAJE DE PREDICCION								PORCENTAJE DE ERROR							
	Modelo 1		Modelo 2		Modelo 3		Modelo 4		Modelo 1		Modelo 2		Modelo 3		Modelo 4	
N°	accuracy	val_accu	accuracy	val_accu	accuracy	val_accu	accuracy	val_accu	loss	val_loss	loss	val_loss	loss	val_loss	loss	val_loss
34	0.997807	0.956522	0.991228	0.886957	0.986842	0.913043	0.986842	0.956522	0.014682	0.172512	0.04726	0.445612	0.063183	0.339746	0.059799	0.223152
35	0.997807	0.956522	0.984649	0.886957	0.986842	0.913043	0.986842	0.956522	0.014132	0.17632	0.07345	0.420384	0.063326	0.338063	0.059357	0.228577
36	0.997807	0.956522	0.984649	0.895652	0.986842	0.913043	0.986842	0.956522	0.013718	0.178647	0.053583	0.378785	0.063041	0.336272	0.059586	0.234173
37	0.997807	0.956522	0.986842	0.913043	0.986842	0.913043	0.986842	0.956522	0.013343	0.180876	0.050561	0.386818	0.062891	0.333703	0.059672	0.2341
38	0.997807	0.956522	0.991228	0.904348	0.986842	0.913043	0.986842	0.956522	0.013179	0.18226	0.040812	0.325078	0.062826	0.328883	0.059365	0.235155
39	0.997807	0.956522	0.986842	0.904348	0.986842	0.913043	0.986842	0.956522	0.012638	0.183828	0.038253	0.350819	0.062549	0.330492	0.059203	0.235598
40	0.997807	0.947826	0.984649	0.904348	0.986842	0.921739	0.986842	0.956522	0.012261	0.196731	0.039182	0.363851	0.062031	0.318766	0.059319	0.238043
41	0.997807	0.947826	0.989035	0.921739	0.986842	0.921739	0.986842	0.956522	0.011092	0.194907	0.03125	0.322553	0.062179	0.318836	0.059133	0.240833
42	0.995614	0.947826	0.989035	0.878261	0.986842	0.921739	0.986842	0.956522	0.009519	0.204129	0.026828	0.554945	0.061725	0.314432	0.058937	0.241471
43	0.997807	0.947826	0.993421	0.878261	0.986842	0.921739	0.986842	0.956522	0.00815	0.208889	0.02498	0.572138	0.061044	0.312265	0.059084	0.241938
44	0.997807	0.947826	0.997807	0.886957	0.986842	0.921739	0.986842	0.956522	0.006434	0.233222	0.019636	0.542564	0.060786	0.310047	0.059228	0.244114
45	1	0.947826	0.995614	0.895652	0.986842	0.921739	0.986842	0.956522	0.002217	0.238426	0.020733	0.480797	0.060298	0.308581	0.05917	0.246072
46	1	0.947826	0.986842	0.886957	0.986842	0.921739	0.986842	0.956522	0.002045	0.242971	0.03923	0.447091	0.059732	0.311656	0.059027	0.247815
47	1	0.947826	0.995614	0.878261	0.986842	0.921739	0.986842	0.956522	0.001906	0.246908	0.018008	0.539549	0.059117	0.309968	0.059193	0.249999
48	1	0.947826	0.995614	0.895652	0.984649	0.921739	0.986842	0.956522	0.001789	0.250569	0.010566	0.51132	0.059199	0.294548	0.058914	0.253816
49	1	0.947826	1	0.886957	0.986842	0.913043	0.986842	0.956522	0.001688	0.25397	0.008154	0.534386	0.060536	0.315914	0.059071	0.253265
50	1	0.947826	0.997807	0.878261	0.986842	0.913043	0.986842	0.956522	0.001599	0.256985	0.007112	0.546561	0.056705	0.321172	0.05892	0.255398
51	1	0.947826	0.997807	0.886957	0.971491	0.843478	0.986842	0.956522	0.001521	0.259871	0.00654	0.538648	0.09416	0.491526	0.058958	0.25504
52	1	0.947826	1	0.886957	0.947368	0.895652	0.986842	0.956522	0.00145	0.262471	0.005962	0.553214	0.125126	0.260133	0.058699	0.254522
53	1	0.947826	1	0.886957	0.97807	0.93913	0.986842	0.956522	0.001387	0.264989	0.005689	0.546185	0.091537	0.293832	0.058994	0.258321
54	1	0.947826	1	0.886957	0.958333	0.904348	0.986842	0.956522	0.001329	0.267435	0.005291	0.549997	0.142652	0.28903	0.059339	0.258214
55	1	0.947826	1	0.886957	0.964912	0.904348	0.986842	0.956522	0.001276	0.269785	0.004906	0.564202	0.147146	0.347357	0.058836	0.260389
56	1	0.947826	1	0.886957	0.964912	0.895652	0.986842	0.956522	0.001228	0.27197	0.004696	0.56903	0.121891	0.246589	0.058766	0.259936
57	1	0.947826	1	0.886957	0.967105	0.930435	0.986842	0.956522	0.001183	0.274035	0.004451	0.562578	0.113163	0.199699	0.058679	0.263829
58	1	0.947826	1	0.886957	0.969298	0.904348	0.986842	0.956522	0.001142	0.276069	0.004215	0.571511	0.093965	0.294961	0.058823	0.262791
59	1	0.947826	1	0.886957	0.969298	0.921739	0.986842	0.956522	0.001104	0.278028	0.004078	0.575464	0.092566	0.203529	0.058788	0.263451
60	1	0.947826	1	0.886957	0.969298	0.886957	0.986842	0.956522	0.001068	0.279937	0.003908	0.579102	0.118518	0.37109	0.058485	0.265479
61	1	0.947826	1	0.886957	0.953947	0.808696	0.989035	0.956522	0.001034	0.28174	0.003753	0.582726	0.122897	0.407477	0.052856	0.264182
62	1	0.93913	1	0.886957	0.936404	0.93913	0.989035	0.956522	0.001003	0.283492	0.00361	0.586692	0.142113	0.210405	0.052797	0.264776
63	1	0.93913	1	0.886957	0.953947	0.947826	0.989035	0.956522	0.000974	0.285213	0.003472	0.59233	0.125818	0.231453	0.053103	0.264109
64	1	0.93913	1	0.886957	0.982456	0.947826	0.989035	0.956522	0.000946	0.286887	0.00335	0.591018	0.076231	0.216736	0.052694	0.264753
65	1	0.93913	1	0.886957	0.951754	0.913043	0.989035	0.956522	0.00092	0.288496	0.003238	0.597615	0.124392	0.296423	0.052679	0.265621
66	1	0.93913	1	0.886957	0.97807	0.947826	0.989035	0.956522	0.000895	0.290065	0.003135	0.598031	0.108071	0.29855	0.052664	0.265902
67	1	0.93913	1	0.886957	0.984649	0.930435	0.989035	0.956522	0.000872	0.291605	0.00304	0.599769	0.07885	0.276934	0.052642	0.266724

	PORCENTAJE DE PREDICCION								PORCENTAJE DE ERROR							
	Modelo 1		Modelo 2		Modelo 3		Modelo 4		Modelo 1		Modelo 2		Modelo 3		Modelo 4	
N°	accuracy	val_accu	accuracy	val_accu	accuracy	val_accu	accuracy	val_accu	loss	val_loss	loss	val_loss	loss	val_loss	loss	val_loss
68	1	0.93913	1	0.886957	0.984649	0.930435	0.989035	0.956522	0.00085	0.29311	0.002953	0.603786	0.075055	0.226316	0.052592	0.267844
69	1	0.93913	1	0.886957	0.982456	0.93913	0.989035	0.956522	0.000829	0.294583	0.002866	0.606759	0.069897	0.254321	0.052477	0.269243
70	1	0.93913	1	0.886957	0.975877	0.904348	0.989035	0.956522	0.000809	0.296018	0.002774	0.610242	0.094951	0.315944	0.05301	0.269466
71	1	0.93913	1	0.886957	0.982456	0.913043	0.989035	0.956522	0.00079	0.29743	0.002703	0.612031	0.069802	0.290937	0.052553	0.269769
72	1	0.93913	1	0.886957	0.97807	0.886957	0.989035	0.956522	0.000772	0.298812	0.002634	0.616788	0.06497	0.364651	0.052509	0.272085
73	1	0.93913	1	0.886957	0.962719	0.913043	0.989035	0.956522	0.000755	0.300181	0.002565	0.619726	0.138725	0.299986	0.052518	0.27123
74	1	0.93913	1	0.886957	0.97807	0.913043	0.989035	0.956522	0.000738	0.301498	0.002499	0.62281	0.104333	0.329363	0.052415	0.272111
75	1	0.93913	1	0.886957	0.980263	0.913043	0.989035	0.956522	0.000722	0.302811	0.002435	0.624165	0.095072	0.322387	0.052501	0.27235
76	1	0.93913	1	0.886957	0.980263	0.913043	0.989035	0.956522	0.000707	0.304106	0.002379	0.627461	0.078115	0.377139	0.052364	0.273154
77	1	0.93913	1	0.886957	0.971491	0.913043	0.989035	0.956522	0.000693	0.305377	0.002321	0.628975	0.101486	0.432412	0.052269	0.276162
78	1	0.93913	1	0.886957	0.982456	0.904348	0.989035	0.956522	0.000679	0.306615	0.002269	0.631647	0.100028	0.338062	0.051977	0.278717
79	1	0.93913	1	0.886957	0.984649	0.93913	0.989035	0.956522	0.000666	0.30785	0.002217	0.634994	0.048374	0.274853	0.051707	0.275647
80	1	0.93913	1	0.886957	0.982456	0.904348	0.989035	0.956522	0.000653	0.309059	0.002168	0.636679	0.082516	0.342541	0.051322	0.273202
81	1	0.93913	1	0.886957	0.982456	0.930435	0.989035	0.956522	0.000641	0.310256	0.002122	0.63926	0.065165	0.248398	0.051099	0.278574
82	1	0.93913	1	0.886957	0.989035	0.930435	0.989035	0.956522	0.000629	0.311434	0.002077	0.641509	0.048963	0.268624	0.050515	0.277463
83	1	0.93913	1	0.886957	0.971491	0.895652	0.989035	0.956522	0.000618	0.312594	0.002034	0.644104	0.108788	0.375837	0.050343	0.277515
84	1	0.93913	1	0.886957	0.975877	0.930435	0.989035	0.956522	0.000607	0.313751	0.001993	0.646676	0.082247	0.268509	0.049704	0.280216
85	1	0.93913	1	0.886957	0.986842	0.930435	0.989035	0.956522	0.000596	0.314881	0.001953	0.647885	0.061362	0.267401	0.050556	0.275971
86	1	0.93913	1	0.886957	0.986842	0.930435	0.989035	0.956522	0.000586	0.315999	0.001916	0.650115	0.059039	0.267717	0.049049	0.268264
87	1	0.93913	1	0.886957	0.986842	0.930435	0.989035	0.956522	0.000576	0.317107	0.001879	0.652451	0.05978	0.260863	0.047021	0.263483
88	1	0.93913	1	0.886957	0.980263	0.895652	0.989035	0.956522	0.000566	0.318201	0.001844	0.654246	0.075087	0.317148	0.045834	0.255253
89	1	0.93913	1	0.886957	0.971491	0.930435	0.989035	0.956522	0.000557	0.319283	0.00181	0.656464	0.077609	0.241945	0.042256	0.257471
90	1	0.93913	1	0.886957	0.97807	0.930435	0.989035	0.956522	0.000548	0.320356	0.001778	0.658513	0.058446	0.317956	0.044789	0.253004
91	1	0.93913	1	0.886957	0.991228	0.93913	0.989035	0.947826	0.00054	0.321417	0.001746	0.659934	0.043853	0.283038	0.044585	0.256328
92	1	0.93913	1	0.886957	0.975877	0.913043	0.989035	0.947826	0.000531	0.322465	0.001716	0.66218	0.122816	0.37225	0.040968	0.257109
93	1	0.93913	1	0.886957	0.980263	0.913043	0.989035	0.947826	0.000523	0.3235	0.001687	0.664071	0.070776	0.339482	0.039317	0.251247
94	1	0.93913	1	0.886957	0.975877	0.904348	0.989035	0.947826	0.000515	0.324527	0.001668	0.665297	0.089049	0.284144	0.037237	0.248064
95	1	0.93913	1	0.886957	0.982456	0.895652	0.989035	0.947826	0.000508	0.325538	0.00163	0.667595	0.049545	0.338209	0.034899	0.244256
96	1	0.93913	1	0.886957	0.980263	0.913043	0.989035	0.947826	0.0005	0.326547	0.001604	0.669238	0.041885	0.408459	0.034024	0.242448
97	1	0.93913	1	0.886957	0.993421	0.921739	0.989035	0.947826	0.000493	0.327537	0.001578	0.67084	0.028187	0.409262	0.033468	0.238393
98	1	0.93913	1	0.886957	0.997807	0.921739	0.989035	0.947826	0.000486	0.328525	0.001554	0.672206	0.01949	0.384496	0.033087	0.23601
99	1	0.93913	1	0.886957	0.997807	0.921739	0.989035	0.956522	0.00048	0.329503	0.00153	0.673774	0.016108	0.388606	0.032505	0.2329

ACTA DE SUSTENTACIÓN DE TESIS

ACTA N° 065-2022-FIMGC

En la ciudad de Ayacucho, en cumplimiento a la **RESOLUCIÓN DECANAL N° 243-2022-FIMGC-D**, siendo el cinco días del mes de setiembre del 2022, a horas 9:30 a.m.; se reunieron los jurados del acto de sustentación, en el Auditórium virtual google meet del Campus Universitario de la Universidad Nacional de San Cristóbal de Huamanga.

Siendo el Jurado de la sustentación de tesis compuesto por el presidente el **Dr. Ing. Efraín Elías PORRAS FLORES,** Jurado - Asesor el **Mg. Ing. Hubner JANAMPA PATILLA,** el Jurado **Mg. Ing. Celia Edith MARTÍNEZ CÓRDOVA,** el Jurado **Dr. Ing. Manuel Avelino LAGOS BARZOLA,** secretario del proceso el **Mg. Ing. Eloy VILA HUAMÁN,** con el objetivo de recepcionar la sustentación de la tesis denominada **"ESTRÉS ESTUDIANTIL UNIVERSITARIO BASADO EN REDES NEURONALES PROFUNDAS, EN PANDEMIA COVID-19, 2021",** sustentado por el Srta. **Kelly Patricia DE LA CRUZ ORIUNDO,** Bachiller en **Ingeniería de Sistemas**.

El Jurado luego de haber recepcionado la sustentación de la tesis y realizado las preguntas, el sustentante al haber dado respuesta a las preguntas, y el Jurado haber deliberado; califica con la nota aprobatoria de **14 (CATORCE).**

En fe de lo cual, se firma la presente acta, por los miembros integrantes del proceso de sustentación.

UNIVERSIDAD NACIONAL DE SAN CRISTÓBAL DE HUAMANGA
FACULTAD DE INGENIERÍA DE MINAS GEOLOGÍA Y CIVIL
Dr. Efraín Elías Porras Flores
DECANO

Firmado digitalmente por Dr. Efraìn Elìas Porras Flores
Fecha: 2022.09.06 10:56:06 -09'00'

Dr. Ing. Efraín Elías PORRAS FLORES
presidente

Mg. Ing. Hubner JANAMPA PATILLA
Jurado Asesor

Mg. Ing. Celia Edith MARTÍNEZ CÓRDOVA
Jurado

Dr. Ing. Manuel Avelino LAGOS BARZOLA
Jurado

Mg. Ing. Eloy VILA HUAMÁN
Secretario del Proceso

c.c.:
Bach. Kelly Patricia DE LA CRUZ ORIUNDO
Jurados (4)
Archivo

CONSTANCIA DE ORIGINALIDAD DE TRABAJO DE INVESTIGACIÓN

CONSTANCIA N° 062-2022-FIMGC

El que suscribe; responsable verificador de originalidad de trabajos de tesis de pregrado en segunda instancia para las **Escuelas Profesionales** de la **Facultad de Ingeniería de Minas, Geología y Civil**; en cumplimiento a la Resolución de Consejo Universitario N° 039-2021-UNSCH-CU, Reglamento de Originalidad de Trabajos de Investigación de la UNSCH y Resolución Decanal N° 158-2021-FIMGC-UNSCH-D, deja constancia que Sr./Srta.

Apellidos y Nombres	:	DE LA CRUZ ORIUNDO, Kelly Patricia
Escuela Profesional	:	INGENIERÍA DE SISTEMAS
Título de la Tesis	:	"ESTRÉS ESTUDIANTIL UNIVERSITARIO BASADO EN REDES NEURONALES PROFUNDAS, EN PANDEMIA COVID- 19, 2021"
Evaluación de la Originalidad	:	**21 %** Índice de Similitud
Identificador de la entrega	:	1889854627

Por tanto, según los Artículos 12, 13 y 17 del Reglamento de Originalidad de Trabajos de Investigación, es **PROCEDENTE** otorgar la **Constancia de Originalidad** para los fines que crea conveniente.

Ayacucho, 31 de agosto del 2022

Firmado digitalmente por LEZAMA CUELLAR CHRISTIAN

Mg. Ing. Christian LEZAMA CUELLAR
Verificador de Originalidad de Trabajos de Tesis de Pregrado de la FIMGC

(X) Con depósito para Sustentación y Tramite de Titulo

FACULTAD DE INGENIERÍA
DE MINAS, GEOLOGÍA Y CIVIL
Av. Independencia S/N
Ciudad Universitaria
Central Tel 066 312510
Anexo 151

“ESTRÉS ESTUDIANTIL UNIVERSITARIO BASADO EN REDES NEURONALES PROFUNDAS, EN PANDEMIA COVID-19, 2021”

por Kelly Patricia De La Cruz Oriundo

Fecha de entrega: 31-ago-2022 12:09a.m. (UTC-0500)
Identificador de la entrega: 1889854627
Nombre del archivo: Tesis_Kelly_Patricia_De_la_Cruz_Oriundo_EPIS.pdf (4.23M)
Total de palabras: 27829
Total de caracteres: 157938

“ESTRÉS ESTUDIANTIL UNIVERSITARIO BASADO EN REDES NEURONALES PROFUNDAS, EN PANDEMIA COVID-19, 2021”

INFORME DE ORIGINALIDAD

21%	21%	3%	13%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	hdl.handle.net Fuente de Internet	2%
2	redie.mx Fuente de Internet	2%
3	www.cali.gov.co Fuente de Internet	2%
4	research.google.com Fuente de Internet	1%
5	repositorio.ug.edu.ec Fuente de Internet	1%
6	Submitted to UNIBA Trabajo del estudiante	1%
7	www.researchgate.net Fuente de Internet	1%
8	repositorio.unasam.edu.pe Fuente de Internet	1%
9	github.com Fuente de Internet	

		1 %
10	www.uniamazonia.edu.co Fuente de Internet	1 %
11	repositorio.unsch.edu.pe Fuente de Internet	1 %
12	www.gob.pe Fuente de Internet	1 %
13	Submitted to Universidad Nacional de San Cristóbal de Huamanga Trabajo del estudiante	1 %
14	documentop.com Fuente de Internet	<1 %
15	repositorio.autonomadeica.edu.pe Fuente de Internet	<1 %
16	Submitted to Universitat Politècnica de València Trabajo del estudiante	<1 %
17	psicohanoi.wixsite.com Fuente de Internet	<1 %
18	somosperiodismo.com Fuente de Internet	<1 %
19	qdoc.tips Fuente de Internet	<1 %

20	www.psicologiacientifica.com Fuente de Internet	<1 %
21	abctexto.blogspot.com Fuente de Internet	<1 %
22	repositorio.unsa.edu.pe Fuente de Internet	<1 %
23	ciateq.repositorioinstitucional.mx Fuente de Internet	<1 %
24	iris.paho.org Fuente de Internet	<1 %
25	bibdigital.epn.edu.ec Fuente de Internet	<1 %
26	reunir.unir.net Fuente de Internet	<1 %
27	sitp.pichincha.gob.ec Fuente de Internet	<1 %
28	www.slideshare.net Fuente de Internet	<1 %
29	Submitted to Universidad Pedagogica y Tecnologica de Colombia Trabajo del estudiante	<1 %
30	www.coursehero.com Fuente de Internet	<1 %
31	Submitted to Universidad Autonoma del Peru	

	Trabajo del estudiante	<1 %
32	repositorio.tec.mx Fuente de Internet	<1 %
33	docs.google.com Fuente de Internet	<1 %
34	iasistemasinteligencia.blogspot.com Fuente de Internet	<1 %
35	tangara.uis.edu.co Fuente de Internet	<1 %
36	torres.ai Fuente de Internet	<1 %
37	www.laculturaesmaravillosa.com Fuente de Internet	<1 %
38	Submitted to Escuela Politecnica Nacional Trabajo del estudiante	<1 %
39	repositorio.ucv.edu.pe Fuente de Internet	<1 %
40	repositorio.autonoma.edu.pe Fuente de Internet	<1 %
41	vsip.info Fuente de Internet	<1 %
42	www.tdx.cat Fuente de Internet	<1 %

43 Submitted to Universidad de Cádiz
Trabajo del estudiante

<1 %

Excluir citas	Activo	Excluir coincidencias	< 30 words
Excluir bibliografía	Activo		