



UNIVERSIDAD ESAN
FACULTAD DE INGENIERÍA
INGENIERÍA DE TECNOLOGÍAS DE INFORMACIÓN Y SISTEMAS

Predicción del estado de financiamiento de proyectos de tecnología en web de crowdfunding
Kickstarter mediante modelo(s) de Aprendizaje Profundo

Tesis para optar el Título de Ingeniero de Tecnologías de Información y Sistemas que
presenta:

Alonso Augusto Puente Ríos
Asesor: Marks Arturo Calderón Niquin

Lima, 25 de febrero de 2021

Esta tesis denominada:

PREDICCIÓN DEL ESTADO DE FINANCIAMIENTO DE PROYECTOS DE
TECNOLOGÍA EN WEB DE CROWDFUNDING KICKSTARTER MEDIANTE
MODELO(S) DE APRENDIZAJE PROFUNDO

ha sido aprobada.

.....
(Jurado Presidente)

.....
(Jurado)

.....
(Jurado)

Universidad ESAN
2021

PREDICCIÓN DEL ESTADO DE FINANCIAMIENTO DE PROYECTOS DE
TECNOLOGÍA EN WEB DE CROWDFUNDING KICKSTARTER MEDIANTE
MODELO(S) DE APRENDIZAJE PROFUNDO

Agradecimiento y dedicatoria

Durante la inducción en la empresa en la cual realicé mis segundas prácticas pre-profesionales, se realizaron varias actividades, entre ellas, una que me marcó positivamente. Esta consistía en comparar los tiempos de llegada de un punto a otro de una persona corriendo. Se caracterizó porque quien asumió el reto tuvo presente en su mente las personas y las razones por las cuales todos los días lucha y son su principal fuente de motivación.

Por ello, quiero dedicar este gran esfuerzo personal de trabajo de tesis a quienes siempre han estado a mi lado en los mejores y peores momentos, aquellos críticos en que definen el destino. Mi amada hermana Clarisabel, mis queridos padres Augusto e Isabel, mi familia en especial mis abuelos; y mis pocos, pero verdaderos y leales amigos de la universidad, colegio y trabajo. Todos ellos han sido y son cada uno, piedra fundamental en el desarrollo de mi ser como persona y profesional, así como también seres con los cuales siempre comparto gratos momentos. Su presencia en mi vida no ha sido una suerte más sino parte de mi destino. Asimismo, luchar por mis sueños y mi país, y pensar cada día en solidificar su planificación me motivan emocionalmente hasta en aquellos momentos en que parece haber imposibles.

Quiero concluir esta sección, muy especial para mí, agradeciendo también a mi alma máter, la Universidad Esan, y al Programa Nacional de Becas (Pronabec) por hacer que estos 5 años entre el 2015 y 2019 sean mágicos y muy fructíferos. Tuve la oportunidad no solo de incrementar y potenciar mis conocimientos en distintas áreas académicas sino también de aprender de excelentes profesionales como mis profesores, conocer grandes amigos dentro y fuera de su campus (desde el primer ciclo como cachimbo hasta el último ciclo, en el CADE Universitario 2019, estudiantes de diferentes universidades y otras partes del Perú), ponerme a prueba en el exterior (en el II Congreso Internacional de Investigación en Colombia) y formar parte de la gran familia UE.

Por todos ellos, simplemente gracias.

Índice general

Índice de Figuras	9
Índice de Tablas	14
Índice de Ecuaciones	15
1 PLANTEAMIENTO DEL PROBLEMA	19
1.1 Descripción de la Realidad Problemática	19
1.2 Formulación del Problema	22
1.2.1 Problema General	22
1.2.2 Problemas Específicos	23
1.3 Objetivos de la Investigación	23
1.3.1 Objetivo General	23
1.3.2 Objetivos Específicos	23
1.4 Justificación de la Investigación	24
1.4.1 Teórica	24
1.4.2 Práctica	24
1.4.3 Metodológica	24
1.5 Delimitación del Estudio	24
1.5.1 Espacial	24
1.5.2 Temporal	24
1.5.3 Conceptual	25
1.6 Hipótesis	25
1.6.1 Hipótesis General	25
1.6.2 Hipótesis Específicas	25
1.6.3 Matriz de Consistencia	25
2 MARCO TEÓRICO	26
2.1 Antecedentes de la investigación	26
2.1.1 Primer antecedente: «KickPredict: Predicting Kickstarter Success» (K. Chen, Jones, Kim & Schlamp, 2013)	26

2.1.2 Segundo antecedente: «The Language that Gets People to Give: Phrases that Predict Success on Kickstarter» (Mitra & Gilbert, 2014)	27
2.1.3 Tercer antecedente: «Money Talks: A Predictive Model on Crowdfunding Success Using Project Description» (M. Zhou, Zhang, Wang, Du, Qiao & Fan, 2015)	29
2.1.4 Cuarto antecedente: «Will your Project get the Green light? Predicting the success of crowdfunding campaigns» (S.-Y. Chen, Chen, Chen, Yang & Lin, 2015)	30
2.1.5 Quinto antecedente: «Predicting Success in Equity Crowdfunding» (Beckwith, 2016)	31
2.1.6 Sexto antecedente: «Project Success Prediction in Crowdfunding Environments» (Y. Li, Rakesh & Reddy, 2016)	33
2.1.7 Séptimo antecedente: «The Determinants of Crowdfunding Success: A Semantic Text Analytics Approach» (H. Yuan, Lau & Xu, 2016)	34
2.1.8 Octavo antecedente: «Using Language to Predict Kickstarter Success» (Sawhney, Tran & Tuason, 2016)	35
2.1.9 Noveno antecedente: «Effect of Social Media Connectivity on Success of Crowdfunding Campaigns» (Kaur & Gera, 2017)	36
2.1.10 Décimo antecedente: «Supervised Learning Model For Kickstarter Campaigns With R Mining» (R. S. Kamath & Kamat, 2018)	38
2.1.11 Undécimo antecedente: «Prediction of Crowdfunding Project Success with Deep Learning» (P.-F. Yu, Huang, Yang, Liu, Li & Tsai, 2018) . .	39
2.1.12 Duodécimo antecedente: «Content-based Success Prediction of Crowdfunding Campaigns: A Deep Learning Approach» (S. Lee, Lee & Kim, 2018)	40
2.1.13 Decimotercer antecedente: «Estimating the Days to Success of Campaigns in Crowdfunding: A Deep Survival Perspective» (Jin, Zhao, Chen, Liu & Ge, 2019)	42
2.1.14 Decimocuarto antecedente: «Success Prediction on Crowdfunding with Multimodal Deep Learning» (Cheng, Tan, Hou & Wei, 2019)	43
2.1.15 Decimoquinto antecedente: «Finding the Keywords Affecting the Success of Crowdfunding Projects» (L.-S. Chen & Shen, 2019)	44
2.1.16 Decimosexto antecedente: «Deploying Natural Language Processing to Extract Key Product Features of Crowdfunding Campaigns: The Case of 3D Printing Technologies on Kickstarter» (Chaichi & Anderson, 2019)	46

2.1.17	Decimoséptimo antecedente: «Topic Predictions and Optimized Recommendation Mechanism Based on Integrated Topic Modeling and Deep Neural Networks in Crowdfunding Platforms» (Shafqat & Byun, 2019)	47
2.2	Bases Teóricas	49
2.2.1	Inteligencia Artificial	49
2.2.2	Aprendizaje Automático	50
2.2.3	Aprendizaje Profundo	55
2.2.4	Modelo Predictivo	56
2.2.5	Minería de Datos	56
2.2.6	Metodologías de Minería de Datos	58
2.2.7	Técnicas de Minería de Datos	62
2.2.8	Procesamiento del Lenguaje Natural	82
2.3	Marco Conceptual	93
2.3.1	Crowdfunding	93
2.3.2	Kickstarter	94
2.3.3	Proyecto	95
2.3.4	Campaña	95
3	METODOLOGÍA DE LA INVESTIGACIÓN	97
3.1	Diseño de la investigación	97
3.1.1	Enfoque de la investigación	97
3.1.2	Alcance de la investigación	97
3.1.3	Tipo de la investigación	98
3.1.4	Descripción del prototipo de investigación	98
3.2	Población y muestra	99
3.2.1	Población	99
3.2.2	Muestra	99
3.2.3	Unidad de análisis	100
3.3	Operacionalización de Variables	100
3.4	Instrumentos de medida	101
3.5	Técnicas de recolección de datos	106
3.6	Técnicas para el procesamiento y análisis de la información	107
3.6.1	Comprensión del negocio	108
3.6.2	Comprensión de los datos	110
3.6.3	Preparación de los datos	110
3.6.4	Modelamiento	111
3.6.5	Evaluación	112

3.6.6 Despliegue	113
3.7 Cronograma de actividades y presupuesto	114
4 DESARROLLO DEL EXPERIMENTO	116
4.1 Construcción de los conjuntos finales de datos	116
4.1.1 Metainformación	116
4.1.2 Descripción	120
4.1.3 Comentarios	123
4.2 Análisis Exploratorios de los datos	125
4.2.1 Metainformación	126
4.2.2 Descripción	131
4.2.3 Comentarios	132
4.3 Pre-procesamiento de los conjuntos de datos	135
4.3.1 Metainformación	135
4.3.2 Descripción	136
4.3.3 Comentarios	137
4.4 Creación de los modelos predictivos	139
4.4.1 Metainformación	139
4.4.2 Descripción	141
4.4.3 Comentarios	144
4.4.4 Modelo apilado	146
5 ANÁLISIS Y DISCUSIÓN DE RESULTADOS	148
5.1 Metainformación	148
5.2 Descripción	150
5.3 Comentarios	153
5.4 Modelo apilado	155
5.5 Comparación de resultados	157
5.6 Demostración del modelo final	158
6 CONCLUSIONES Y RECOMENDACIONES	160
6.1 Conclusiones	160
6.2 Recomendaciones	162
BIBLIOGRAFÍA	163
Anexo	172
Anexo A Árbol de Problemas	173

Anexo B Árbol de Objetivos **174**

Anexo C Matriz de Consistencia **175**

Índice de Figuras

1.1 Resultados y ratios obtenidos en la encuesta por GEM y ESAN. Fuente: Redacción Gestión, 2018	20
1.2 Ratio de éxito de proyectos en Kickstarter desde 2009 hasta 2019 (Febrero). Fuente: The Hustle, 2019	22
2.1 Ejemplo de algoritmos de regresión y clasificación. Fuente: Zambrano, 2018 . .	51
2.2 Algoritmo de K Vecinos más cercanos con pesos ponderados. Fuente: Sancho Caparrini, 2018	52
2.3 Funcionamiento del algoritmo de K medias. Fuente: Sancho Caparrini, 2018 . .	54
2.4 Componentes del Aprendizaje por Refuerzo. Fuente: Sutton y Barto, 2018 . . .	54
2.5 Diferencia entre Aprendizaje Automático y Aprendizaje Profundo. Fuente: House of Bots, 2018	55
2.6 Diagrama de los seis pasos básicos. Fuente: Microsoft, 2019	57
2.7 Fases de la metodología CRISP-DM. Fuente: Braulio Gil y Curto Díaz, 2015 . .	58
2.8 Fases de la metodología SEMMA. Fuente: Braulio Gil y Curto Díaz, 2015 . .	59
2.9 Fases de la metodología KDD. Fuente: Braulio Gil y Curto Díaz, 2015	60
2.10 Modelo para representar una neurona propuesto por McCulloch y Pitts (1943). Fuente: Russell y Norvig, 2004	62
2.11 Nodos con funciones de activación umbral en forma de puertas lógicas. Fuente: Russell y Norvig, 2004	64
2.12 Función de activación sigmoide. Fuente: Dorofki y col., 2012	64
2.13 Ilustración del algoritmo gradiente descendiente. Fuente: Sancho Caparrini, 2017	66
2.14 Actualización de pesos W con el algoritmo. Fuente: IArtificial.net, 2019b . . .	66
2.15 Capa oculta simple MLP con propagación hacia atrás. Fuente: IArtificial.net, 2019b	67
2.16 Redes neuronales de ejemplo. Fuente: A Not So Random Walk, 2019	68
2.17 Función de activación tangente hiperbólica. Fuente: Dorofki y col., 2012 . . .	69
2.18 Función de activación puramente lineal. Fuente: Dorofki y col., 2012	70
2.19 Función de activación ReLU. Fuente: Machine Learning for Artists, 2019 . . .	70
2.20 Ejemplo de perceptrón simple. Fuente: Calvo, 2017	71

2.21 Ejemplo de perceptrón multicapa. Fuente: Calvo, 2017	71
2.22 Ejemplo de red neuronal convolucional. Fuente: Calvo, 2017	72
2.23 Modelo Neocognitron de Fukushima (1980). Fuente: F.-F. Li y col., 2019	73
2.24 Modelo LeNet-5 de LeCun (1998). Fuente: F.-F. Li y col., 2019	73
2.25 Ejecución de la convolución en una entrada. Fuente: López Briega, 2016	74
2.26 Generación de una nueva imagen a partir de filtros. Fuente: F.-F. Li y col., 2019	74
2.27 Secuencia de varias capas convolucionales. Fuente: F.-F. Li y col., 2019	75
2.28 Extracción de características a partir de convoluciones. Fuente: F.-F. Li y col., 2019	75
2.29 Ejemplo de matriz de imagen de entrada y un filtro. Fuente: Prabhu, 2018	75
2.30 Dimensiones de una entrada y un filtro. Fuente: F.-F. Li y col., 2019	76
2.31 Paso de 2 píxeles por parte de un filtro. Fuente: Prabhu, 2018	77
2.32 Aplanado de matrices luego de agrupar la capa. Fuente: Prabhu, 2018	77
2.33 Arquitectura completa de una CNN. Fuente: Prabhu, 2018	78
2.34 Ejemplo de red neuronal recurrente. Fuente: Calvo, 2018a	78
2.35 Hiperplano con dos clases separadas por una distancia m. Fuente: Betancourt, 2005	79
2.36 Ejemplo de separación de 2 clases. Fuente: Betancourt, 2005	80
2.37 Aplicación de un kernel para transformar el espacio de los datos. Fuente: Betancourt, 2005	80
2.38 Ejemplo del algoritmo de árbol de decisión. Fuente: Russell y Norvig, 2004	81
2.39 Arquitectura de modelo CNN con 2 canales para una oración de ejemplo. Fuente: Kim, 2014	83
2.40 Diferencias entre convoluciones según su dimensión. Fuente: MissingLink.ai, s.f.	84
2.41 Arquitectura de modelo CNN para clasificación de oraciones. Fuente: Zhang y Wallace, 2017	84
2.42 Arquitectura de una LSTM. Fuente: X. Yuan y col., 2019	86
2.43 Representación de arquitectura BiRNN de la palabra <i>jumped</i> en la oración. Fuente: Goldberg, 2017	87
2.44 Comparación entre arquitecturas LSTM y GRU. Fuente: Phi, 2018	87
2.45 Arquitectura de un modelo Seq2seq. Fuente: Kostadinov, 2019	88
2.46 Arquitectura de un modelo multimodal de imágenes y texto. Fuente: Nishida y Nakayama, 2015	88
2.47 Ejemplo de funcionamiento de una capa de incrustación. Fuente: Chengwei, 2018	90
2.48 Incrustaciones de palabras por Word2Vec. Fuente: Bujokas, 2020	90
2.49 Incrustaciones de palabras por GloVe. Fuente: Pennington y col., 2014b	91
2.50 Ejemplo de funcionamiento de bolsa de palabras. Fuente: V. Zhou, 2019	92

3.1	Marco de trabajo del prototipo final. Fuente: Elaboración propia	99
3.2	Descripción de resultados de modelo descriptivo de ejemplo. Fuente: González, 2019	104
3.3	Comparación de tres resultados de la curva AUC en el modelo. Fuente: Molina Arias y Ochoa Sangrador, 2017	105
3.4	Flujograma de la recolección de conjuntos finales de datos. Fuente: Elaboración propia.	106
3.5	Cronograma de actividades de la investigación. Fuente: Elaboración propia	114
4.1	Vista (agosto 2019) del website Web Robots. Fuente: Web Robots, 2019	117
4.2	Tamaño de conjunto de datos al corte de Julio 2019. Fuente: Elaboración propia.	117
4.3	Tamaño de conjunto de datos filtrado del corte de Julio 2019. Fuente: Elaboración propia.	118
4.4	Flujo de trabajo de la data final en Alteryx Designer. Fuente: Elaboración propia.	119
4.5	Visualización del archivo de metainformación subido a Kaggle. Fuente: Elaboración propia.	120
4.6	Función del algoritmo web scraping de la descripción de proyectos. Fuente: Elaboración propia.	122
4.7	Visualización del archivo de descripción subido a Kaggle. Fuente: Elaboración propia.	122
4.8	Función del algoritmo web scraping de los comentarios. Fuente: Elaboración propia.	123
4.9	Visualización del archivo de comentarios subido a Kaggle. Fuente: Elaboración propia.	124
4.10	Instancias lanzadas en paralelo para la extracción de comentarios. Fuente: Elaboración propia.	124
4.11	Distribución de proyectos tecnológicos según su estado. Fuente: Elaboración propia.	125
4.12	Evolución de cantidad de proyectos tecnológicos por año. Fuente: Elaboración propia.	125
4.13	Evolución de proyectos tecnológicos, por su estado y año. Fuente: Elaboración propia.	126
4.14	Distribución de categorías de tecnología. Fuente: Elaboración propia.	127
4.15	Distribución de países. Fuente: Elaboración propia.	127
4.16	Distribución de divisa del monto patrocinado. Fuente: Elaboración propia	127
4.17	Diagrama de caja y bigote de patrocinadores. Fuente: Elaboración propia.	128
4.18	Diagrama de caja y bigote de meta. Fuente: Elaboración propia.	128

4.19 Diagrama de caja y bigote de monto patrocinado. Fuente: Elaboración propia.	129
4.20 Diagrama de caja y bigote de duración. Fuente: Elaboración propia.	129
4.21 Matriz de correlaciones entre variables independientes. Fuente: Elaboración propia.	130
4.22 Gráfico de dispersión de correlaciones entre variables independientes. Fuente: Elaboración propia.	130
4.23 Gráfico alterno de dispersión de correlaciones entre variables independientes. Fuente: Elaboración propia.	131
4.24 Aperturas de proyectos por presencia de comentarios y estado de financiamiento. Fuente: Elaboración propia.	132
4.25 Nube de palabras de descripciones. Fuente: Elaboración propia.	132
4.26 Aperturas de proyectos por estado de financiamiento y presencia de comentarios. Fuente: Elaboración propia.	133
4.27 Aperturas de proyectos por presencia de comentarios y estado de financiamiento. Fuente: Elaboración propia.	133
4.28 Distribución de comentarios en proyectos exitosos y fracasados. Fuente: Elaboración propia.	134
4.29 Nube de palabras de comentarios más frecuentes. Fuente: Elaboración propia. .	134
4.30 Matriz de correlaciones entre variables independientes considerando adicionales. Fuente: Elaboración propia.	135
4.31 Flujograma de limpieza de conjunto de datos de descripciones. Fuente: Elaboración propia.	137
4.32 Nube de palabras de descripciones posterior al pre-procesamiento. Fuente: Elaboración propia.	138
4.33 Nube de palabras de comentarios posterior al pre-procesamiento. Fuente: Elaboración propia.	138
4.34 Arquitectura de modelo MLP para la metadata. Fuente: Elaboración propia. . .	140
4.35 Función de tokenización de palabras de descripciones. Fuente: Elaboración propia.	141
4.36 Función de codificación de palabras y relleno de arreglos. Fuente: Elaboración propia.	141
4.37 Elaboración de matriz de incrustaciones de palabras. Fuente: Elaboración propia.	142
4.38 Arquitectura de modelo CNN para las descripciones. Fuente: Elaboración propia.	143
4.39 Arquitectura de modelo RNN para los comentarios. Fuente: Elaboración propia.	144
4.40 Arquitectura del modelo apilado final. Fuente: Elaboración propia.	146

5.1 Exactitud y pérdida respectivamente de los subconjuntos de entrenamiento y validación para el modelo MLP de metadata con 100 épocas. Fuente: Elaboración propia.	149
5.2 Matriz de confusión para el modelo de metadata. Fuente: Elaboración propia.	149
5.3 Área bajo la curva ROC de modelo de metadata. Fuente: Elaboración propia.	150
5.4 Exactitud y pérdida respectivamente de los subconjuntos de entrenamiento y validación para el modelo CNN de descripciones con 100 épocas. Fuente: Elaboración propia.	151
5.5 Matriz de confusión para el modelo de descripciones. Fuente: Elaboración propia.	151
5.6 Área bajo la curva ROC de modelo de descripciones. Fuente: Elaboración propia.	152
5.7 Exactitud y pérdida respectivamente de los subconjuntos de entrenamiento y validación para el modelo RNN de comentarios con 50 épocas. Fuente: Elaboración propia.	153
5.8 Matriz de confusión para el modelo de comentarios. Fuente: Elaboración propia.	153
5.9 Área bajo la curva de modelo de comentarios. Fuente: Elaboración propia.	154
5.10 Exactitud y pérdida respectivamente de los subconjuntos de entrenamiento y validación para el modelo apilado con 200 épocas. Fuente: Elaboración propia.	155
5.11 Matriz de confusión para el modelo apilado. Fuente: Elaboración propia.	156
5.12 Área bajo la curva de modelo apilado. Fuente: Elaboración propia.	156
5.13 Flujo gráfico del sistema piloto. Fuente: Elaboración propia.	158
5.14 Proyecto usado para la demostración. Captura de pantalla: 15/02/21. Fuente: Kickstarter, 2021	158
5.15 Variables extraídas del proyecto. Fuente: Elaboración propia.	159
5.16 Resultado de predicción de The Hydra para el proyecto consultado. Fuente: Elaboración propia.	159

Índice de Tablas

2.1 Cuadro comparativo entre características de las tres metodologías. Fuente: Shafique y Qaiser, 2014	61
3.1 Diccionario de datos del conjunto final entrenado. Fuente: Elaboración propia. .	100
3.2 Matriz de confusión. Fuente: Izco, 2018	102
3.3 Presupuesto de los costos personales del autor. Fuente: Elaboración propia. . .	115
3.4 Presupuesto de los costos de las herramientas para el proyecto. Fuente: Elaboración propia.	115
4.1 Diccionario de datos del dataset de metainformación generado. Fuente: Elaboración propia.	121
4.2 Combinatorias posibles de variables de metadata. Fuente: Elaboración propia. .	136
5.1 Informe de clasificación para el modelo de metadata. Fuente: Elaboración propia.	150
5.2 Informe de clasificación para el modelo de descripciones. Fuente: Elaboración propia.	152
5.3 Informe de clasificación para el modelo de comentarios. Fuente: Elaboración propia.	154
5.4 Informe de clasificación para el modelo apilado. Fuente: Elaboración propia. .	155
5.5 Comparación de resultados de modelos propuestos con antecedentes. Fuente: Elaboración propia.	157

Índice de Ecuaciones

2.1 Cálculo de los pesos para el algoritmo K-NN mediante ponderación de sus distancias. Fuente: Sancho Caparrini, 2018	52
2.2 Fórmula alternativa del algoritmo K-NN mediante sumatoria de pesos. Fuente: Sancho Caparrini, 2018	52
2.3 Fórmula del algoritmo k-means. Fuente: Sancho Caparrini, 2018	53
2.4 Fórmula del cálculo del valor de un nodo i. Fuente: Russell y Norvig, 2004	63
2.5 Fórmula de una función de activación g para la salida del nodo. Fuente: Russell y Norvig, 2004	63
2.6 Fórmula de la función de activación sigmoide. Fuente: Dorofki y col., 2012	64
2.7 Fórmula de función de coste de una regresión logística. Fuente: Pardo, s.f.	65
2.8 Actualización de pesos W mediante gradiente descendiente. Fuente: IArtificial.net, 2019b	66
2.9 Fórmula del algoritmo de propagación hacia atrás. Fuente: Bertona, 2005	67
2.10 Cálculo del error cometido en una red neuronal. Fuente: Viera Balanta, 2013	68
2.11 Actualización de pesos mediante propagación hacia atrás. Fuente: Viera Balanta, 2013	68
2.12 Cálculo de errores de nodos usando pesos actualizados. Fuente: Viera Balanta, 2013	69
2.13 Fórmula de la función de activación tangente hiperbólica. Fuente: Dorofki y col., 2012	69
2.14 Fórmula de la función de activación puramente lineal. Fuente: Dorofki y col., 2012	70
2.15 Fórmula de la función de activación ReLU. Fuente: Calvo, 2018b	70
2.16 Fórmula matemática de la convolución. Fuente: Figueroa M., s.f.	73
2.17 Cálculo del volumen del mapa de activación. Fuente: Prabhu, 2018	75
2.18 Cálculo del tamaño de la imagen reducida. Fuente: F.-F. Li y col., 2019	76
2.19 Cálculo del tamaño de la imagen reducida con bordes rellenos con ceros. Fuente: F.-F. Li y col., 2019	77
2.20 Ecuación del hiperplano para clasificar dos clases. Fuente: Betancourt, 2005	80

2.21 Fórmula para determinar un estado en una S-RNN. Fuente: Brownlee, 2017a	85
2.22 Fórmula de TF-IDF. Fuente: Hamdaoui, 2019	92
3.1 Fórmula para calcular la exactitud. Fuente: Kohavi y Provost, 1998	103
3.2 Fórmula para calcular la precisión. Fuente: Kohavi y Provost, 1998	103
3.3 Fórmula para calcular la sensibilidad. Fuente: Kohavi y Provost, 1998	105
3.4 Fórmula para calcular el puntaje F1. Fuente: Kohavi y Provost, 1998	105

Resumen

Conocer el destino del financiamiento de proyectos siempre ha sido el principal deseo de todos los emprendedores que los promocionan en Internet, en especial, de la categoría de tecnología por ser los que presentan las ratios más bajas de éxito debido a sus altas metas que buscan alcanzar. El presente trabajo de investigación se basó en construir un modelo predictivo cuyo objetivo es la de estimar el éxito o fracaso de financiamiento de proyectos tecnológicos en la plataforma de crowdfunding Kickstarter durante su campaña a partir de su metainformación, descripción y comentarios de los patrocinadores. Para ello, se creó un Modelo Apilado Integrado de modelos de Aprendizaje Profundo para cada modalidad, denominada The Hydra. Luego de analizar los resultados del modelo final con las métricas mencionadas en la literatura, se concluyó que el modelo propuesto presentó mejor rendimiento que la mayoría de los antecedentes debido a la mejora de su capacidad colectiva por encima de cada performance individual que lo compone. Como trabajo a futuro, el sistema piloto que incluye al modelo será insertada en una plataforma web, se afinarán más hiperparámetros y se considerarán más variables potenciales.

Palabras claves: metainformación, descripción, comentarios, Aprendizaje Profundo, Modelo Apilado, Perceptrón Multicapa (MLP), Red Neuronal Convolutacional (CNN), Red Neuronal Recurrente (RNN).

Knowing funding projects destiny has always been the main desire of all entrepreneurs who promote them on the Internet, especially in the technology category because they have the lowest success rates due to their high goals. The present research work was based on building a predictive model whose objective is to estimate the success or failure of technological projects funding in the Kickstarter crowdfunding platform during its campaign based on its metadata, description and backers comments. For this, an Integrated Stacking Model based on Deep Learning models for each modal created, called The Hydra. After analyzing the final model results with metrics mentioned in the literature, it was concluded that the proposed model presented better performance than most of the antecedents due to the improvement of its collective capacity over each individual performance that composes it. As work in the future, the pilot system that includes the model will be inserted into a web platform, more hyperparameters will be refined and more potential variables will be considered.

Keywords: project metadata, project description, project comments, Deep Learning, Stacked Model, Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN).

Introducción

Por muchos años, en especial en las dos últimas décadas, diversos proyectos emprendedores han sido lanzados en distintas plataformas web, buscando un objetivo compartido por todos: ser financiados en un determinado plazo para hacer realidad estas ideas. Entre fracasos y éxitos, han surgido nuevas tendencias, así como nuevos enfoques de estudios de estos casos para encontrar la clave que descifre las variables de éxito.

El presente trabajo de investigación se basó formular un modelo ensamblado robusto que determine el estado final de un proyecto, agregando un nuevo enfoque: basarse solamente en proyectos de tecnología, la segunda categoría con más baja probabilidad de éxito al final de una campaña. En estudios previos, los modelos planteados resultaron bastante aceptables debido a que el resto de categorías de proyectos balancearon la inequidad existente en las dos clases del estado.

El reto principal fue el de construir modelos predictivos que consideren 3 características importantes de un proyecto: la metainformación, descripción de la campaña, y los comentarios acerca de esta hechos por los patrocinadores.

Para ello, se recolectó un total de 27,251 proyectos tecnológicos en Kickstarter entre los períodos 2009-2019. Algunos proyectos provenientes de países fuera del territorio de los Estados Unidos y en distintos idiomas fueron considerados dentro de esta cantidad ya que no afectó al rendimiento general como en casos particulares de algunos estudios previos.

La principal motivación de este trabajo fue la de aportar una herramienta de ayuda para los emprendedores que les permita estimar el estado final del financiamiento de su proyecto de tecnología, es decir, éxito o fracaso, con un nivel confiable de probabilidad de éxito del mismo durante el transcurso de su campaña, permitiendo además servir de soporte en la toma de decisiones de cara a lograr su principal objetivo.

Capítulo 1

PLANTEAMIENTO DEL PROBLEMA

1.1. Descripción de la Realidad Problemática

El emprendimiento hoy en día es una realidad en todo el mundo. Desde crear productos nuevos hasta crear nuevas formas de hacer las cosas, todo gracias a ideas nacidas a partir de querer satisfacer nuestras propias necesidades.

Nuestro país no es ajeno a ello. El 50.6% de la población entre 18 y 64 años tiene la expectativa de iniciar un emprendimiento dentro de los tres próximos años de acuerdo al último reporte de Global Entrepreneurship Monitor (GEM) 2014. El 62.3% de la población entre ese rango de edad, además, tiende a ser más optimista en su percepción de oportunidades. Asimismo, según informa la Cámara de Comercio de Lima, la iniciativa emprendedora responde más a la identificación de una oportunidad de negocio que a una falta de oportunidad de empleo ([Redacción Gestión, 2015](#)). Sin embargo, en un estudio más reciente basado en una encuesta realizada a residentes peruanos entre junio y julio del 2017 desarrollada por el equipo GEM Perú y ESAN a 2080 personas entre el mismo rango de edad, el 24.6% de emprendimientos se encuentra en fase temprana, es decir, representa una dificultad para el emprendedor peruano llegar a etapas más avanzadas como un emprendimiento establecido (negocios con más de 3.5 años, que representan solo el 7.4% para Perú), ubicando así a nuestro país en la posición 25 de 54 economías a nivel mundial ([Redacción Gestión, 2018](#)). En la Figura 1.1 se aprecian algunas ratios del estudio.

Estos resultados desfavorables tienen como base el ecosistema poco beneficioso para los emprendimientos que permitan su establecimiento en el entorno nacional, con condiciones asociadas al acceso de financiamiento, políticas gubernamentales que alienen la implementación de Innovación y Desarrollo en las empresas, acceso a infraestructura física y asesoría

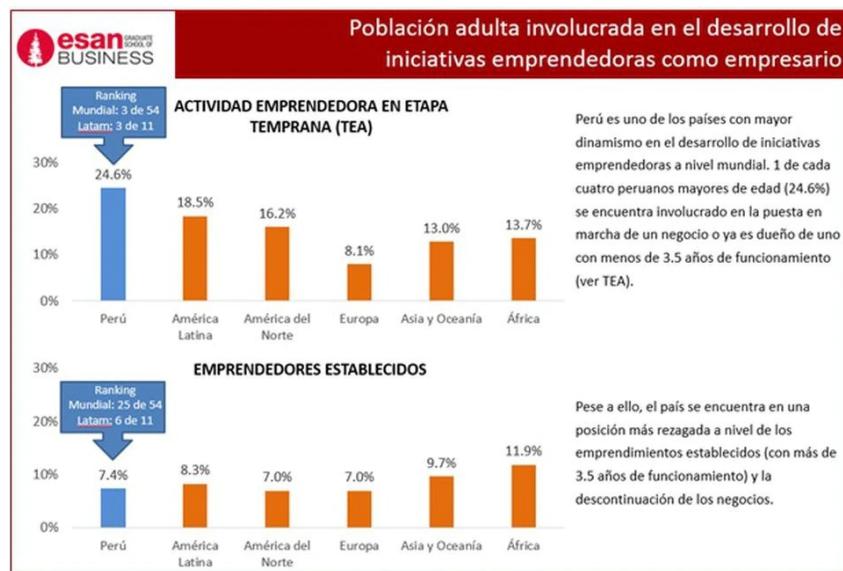


Figura 1.1: Resultados y ratios obtenidos en la encuesta por GEM y ESAN. Fuente: [Redacción Gestión, 2018](#)

a nivel comercial y profesional, como sostiene el investigador del equipo GEM Perú Carlos Guerrero ([Redacción Gestión, 2018](#)). La Asociación de Emprendedores de Perú (ASEP) afirma, asimismo, que en la región solo se invierte el 1.5 % del PIB en actividades de ciencia, tecnología e innovación, y las limitaciones son dadas por barreras burocráticas ejercidas por el Gobierno y el sector privado ([Asociación de Emprendedores de Perú, 2018](#)). En adición a esto, otras razones que representan barreras para emprender son la falta de conocimientos en la iniciación de un negocio, su tramitación, la fuente de financiamiento del proyecto o búsqueda de inversionistas, la cultura, la falta de fomento de emprendimiento y la falta de una red de contactos ([Sandoval, s.f.](#)).

Ante estas limitaciones, en la actualidad muchos emprendedores se ven forzados a mostrar sus proyectos al público en la Internet con el fin de captar personas interesadas en ayudarlos en el financiamiento de estos. Por ello, se han creado plataformas web con el fin de permitir la interacción entre los proyectos publicados en un determinado tiempo, el cual puede variar entre 30 y 120 días, y la comunidad en general que desee colaborar con una cantidad de dinero para su financiamiento. El sitio web solo servirá para mostrar los proyectos presentados a detalle por los creadores y la promoción de estos al público. La idea es que, al término de este plazo de tiempo, el proyecto sea financiado y se logre convertir en una realidad. A esta práctica se le conoce como crowdfunding ([Universo Crowdfunding, s.f.](#)).

En Latinoamérica, son muy pocos los países los que se incorporan en el crowdfunding, entre los principales se destacan Chile, México, Argentina y Brasil. Sin embargo, el modelo funciona distinto a países de Norteamérica y Europa debido a la cultura diferente y resistencia

a su implementación por la poca confianza en el éxito de los proyectos. En los países mencionados, se encontró que los proyectos audiovisuales, a pesar de encontrarse en desventaja que en sus pares europeos y norteamericanos, se estrenaron 4,135 títulos extranjeros en la región iberoamericana frente a 791 propios, de los cuales 162 fueron obras brasileñas y 131, argentinas, en el año 2015. Estas estadísticas se apoyan con el incremento de aperturas de salas de cines en nuestra región en las dos últimas décadas. Los factores principales: el apoyo de gobiernos latinoamericanos, la masificación de las comunicaciones digitales y por ende, la participación activa de usuarios en redes sociales y el papel protagónico que toma el crowdfunding para apoyar estos proyectos (22,179 proyectos financiados con éxito en Kickstarter con más de 1 millón de dólares de recaudación en el 2017) ([López-Golán y col., 2017](#)). Asimismo, en los últimos años se decidió seguir una manera muy similar a los modelos de Estados Unidos, basados en la creación de campañas de un emprendedor para obtener fondos para sus ideas con la moneda norteamericana pero limitados a las leyes económicas de cada país ([Solidaridad Latina, s.f.](#)).

En el caso de nuestro país, el crowdfunding sí existe y es apoyada por universidades y organizaciones sin fines de lucro, cuyo objetivo es generar empresas comerciales que aumenten la calidad de vida en la comunidad, además de convertir a las personas en socios financieros ([Fernández-Bedoya y col., 2020](#)). Esta actividad es parte de uno de los 4 grupos básicos de economía colaborativa: el financiamiento colaborativo ([Stokes y col., 2014](#)).

Entre los sitios web más conocidos de crowdfunding están Kickstarter e Indiegogo. Kickstarter, desde su inicio en 2009, es una plataforma de financiamiento de proyectos creativos de todo tipo, los cuales incluyen películas, juegos, música, arte, diseño y tecnología. Actualmente, se han registrado más de 162 mil proyectos realizados, 16 millones de contribuyentes y 4,3 miles de millones de dólares fondeados ([Kickstarter, s.f.-a](#)). La plataforma utiliza un modelo de financiamiento llamado “todo o nada”, el cual consiste en que si un proyecto no alcanza su meta de financiamiento en un determinado plazo de tiempo, no se realiza ninguna transacción de fondos ([Kickstarter, s.f.-d](#)). Si bien los patrocinadores apoyan estos proyectos por motivos personales y distintos para hacerlos realidad, ellos no obtienen la propiedad o los ingresos de los proyectos que financian, sino que los creadores conservan la totalidad de su trabajo ([Kickstarter, s.f.-f](#)).

Para los proyectos tecnológicos, en contraste, la ratio de éxito es uno de los más bajos de las categorías existentes (20%) solo por delante de Artesanía y Periodismo, como se aprecia en la Figura 1.2.

Ya existen estudios previos para predecir la probabilidad de éxito de financiamiento para este tipo de proyectos utilizando técnicas de Aprendizaje Automático. Sin embargo, la mayoría de los modelos predictivos propuestos no arrojan resultados con exactitud muy alta

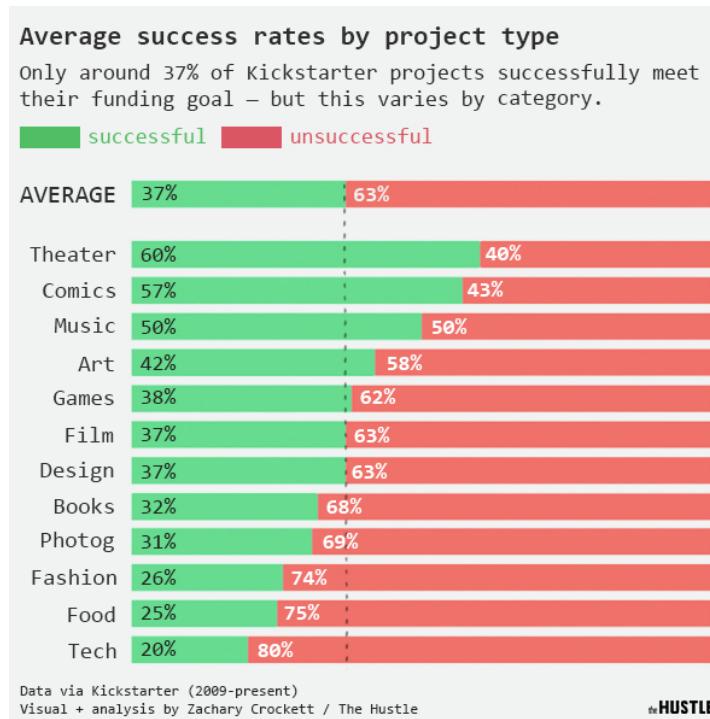


Figura 1.2: Ratio de éxito de proyectos en Kickstarter desde 2009 hasta 2019 (Febrero). Fuente: [The Hustle, 2019](#)

ya que su rango varía entre 60 y 70%. Esto conlleva a generar imprecisión para pronosticar confiablemente el éxito de financiamiento de estos proyectos de tecnología. Para el presente trabajo de tesis, se creó un modelo predictivo alimentado de datos históricos de la plataforma para estimar el estado final de financiamiento de un proyecto aleatorio, así como su probabilidad de éxito.

1.2. Formulación del Problema

Para la formulación de los problemas de la presente investigación, se elaboró un «árbol de problemas» (véase Anexo A).

1.2.1. Problema General

Bajo rendimiento de modelos de Aprendizaje Profundo según métricas de evaluación de Machine Learning para predecir estado de financiamiento de proyectos de tecnología.

1.2.2. Problemas Específicos

- Presencia de data desbalanceada.
- Carencia de datos en variables de algunos proyectos.
- Información irrelevante considerada para entrenamiento de modelos.
- Parámetros no ajustados de modelos.
- Sobreajuste de aprendizaje de modelos y clasificación incorrecta de clases del estado final de financiamiento (exitoso o fracasado).
- Falta de herramientas predictivas para ayudar a creadores de proyectos en la toma de decisiones.

1.3. Objetivos de la Investigación

Para la formulación de los objetivos de la presente investigación, se elaboró un «árbol de objetivos» (véase Anexo B)

1.3.1. Objetivo General

Construir modelo de Aprendizaje Profundo para predecir éxito de financiamiento de proyectos de tecnología.

1.3.2. Objetivos Específicos

- Utilizar técnicas de Machine Learning para trabajar con data desbalanceada.
- Imputar datos faltantes o incompletos de proyectos.
- Definir información relevante para entrenamiento de modelos.
- Ajustar parámetros de modelos.
- Evitar sobreajuste de aprendizaje del modelo y clasificación incorrecta de clases del estado final de financiamiento (exitoso o fracasado).
- Ofrecer herramienta analítica y predictiva a creadores de proyectos para ayudar en la toma de decisiones.

1.4. Justificación de la Investigación

1.4.1. Teórica

Esta investigación se basa en crear un modelo de Aprendizaje Automático que sea aplicable a proyectos de tecnología de la plataforma Kickstarter por presentar bajas performances en antecedentes.

1.4.2. Práctica

Al culminar la investigación, se ofrecerá un modelo predictivo confiable que ayude a los emprendedores en la toma de decisiones respecto a sus proyectos a partir del insight obtenido de los resultados que deriven a la manipulación de los datos de entrada.

1.4.3. Metodológica

Se creará un modelo predictivo a partir de las variables finales seleccionadas, previa limpieza de datos. Luego, será entrenado y evaluado por las métricas correspondientes. Finalmente, se lanzará una versión de prueba que reciba datos de entrada para predecir la viabilidad de un proyecto de tecnología.

1.5. Delimitación del Estudio

1.5.1. Espacial

Para la presente investigación, se considerará el territorio de los Estados Unidos ya que tanto la campaña del proyecto a servir para la investigación como los datos fuentes de proyectos relacionados financiados previamente, que servirán para la elaboración del modelo predictivo, se encuentran en dicho país.

1.5.2. Temporal

El periodo de tiempo abarcará desde el año 2009, fecha en el cual se tiene registrado los primeros conjuntos de datos de proyectos en Kickstarter hasta el mes de agosto del año 2019,

últimos registros descargados hasta el inicio del presente trabajo.

1.5.3. Conceptual

La presente investigación consistirá en la implementación de un modelo predictivo del estado de financiamiento de un proyecto tecnológico en Kickstarter basado en técnicas y conceptos de Aprendizaje Automático, previamente evaluando cuál de todas las existentes genera un mejor desempeño para su uso y análisis de resultados.

1.6. Hipótesis

1.6.1. Hipótesis General

El modelo propuesto de Aprendizaje Profundo predice el estado de financiamiento de proyectos de tecnología.

1.6.2. Hipótesis Específicas

- Las técnicas de Machine Learning ayudan a los modelos a trabajar con data desbalanceada.
- Los datos faltantes o incompletos de proyectos son imputados.
- Se considera solo información relevante para entrenamiento de modelos.
- Los parámetros de los modelos están ajustados.
- Se evita el sobreajuste de aprendizaje del modelo y éste clasifica correctamente las clases del estado final de financiamiento (exitoso o fracasado).
- Se ofrece herramienta analítica y predictiva para ayudar a los creadores de proyectos en la toma de decisiones.

1.6.3. Matriz de Consistencia

A continuación se presenta la matriz de consistencia elaborada para la presente investigación (véase Anexo C).

Capítulo 2

MARCO TEÓRICO

2.1. Antecedentes de la investigación

En esta sección se presentarán diversos trabajos de investigación basados en la predicción de éxito o fracaso de campañas en Kickstarter o plataformas similares y el análisis de estas utilizando conjunto de datos de la propia plataforma o almacenadas en otros repositorios, con sus variables respectivas. En la mayoría de casos, consideraron variables básicas que se obtienen del repositorio de las plataformas de crowdfunding, en otros casos consideraron variables cualitativas basadas en texto y descripción de proyectos, y en otros antecedentes usaron nuevas técnicas poco convencionales como el Aprendizaje Profundo e híbridos de modelos para obtener los mejores resultados posibles.

2.1.1. Primer antecedente: «KickPredict: Predicting Kickstarter Success» (K. Chen, Jones, Kim & Schlamp, 2013)

K. Chen y col. realizaron la publicación «KickPredict: Predicting Kickstarter Success» para el Departamento de Ciencias Computacionales y Matemáticas del Instituto de Tecnología de California. Este fue titulado *KickPredict: Predicting Kickstarter Success* la cual traducida al español significa «KickPredict: Predicción del éxito de Kickstarter».

2.1.1.1. Planteamiento del Problema y objetivo

Ante la cantidad considerable de proyectos en Kickstarter que fracasan en finanziarse debido a los datos colocados por sus creadores, los autores desarrollaron un sistema predic-

tor de estado de financiamiento de un proyecto a partir de la obtención de las variables más importantes, tanto internas como de fuentes externas (YouTube y Twitter).

2.1.1.2. Técnicas empleadas por los autores

Para el sistema propuesto, se utilizaron Máquinas de Vectores de Soporte (SVM) entrenadas con cada combinatoria posible de las variables del conjunto de datos.

2.1.1.3. Metodología empleada por los autores

Se recolectaron 20 mil proyectos completados, es decir, cuyo estado de financiamiento fue exitoso o fracasado. De esta cantidad, el 95 % fue destinado al subconjunto de entrenamiento. Después, se utilizó un algoritmo clasificador binario estándar para identificar las variables más importantes. Por último, se desarrolló una aplicación en Android y una extensión en Google Chrome para usar en proyectos nuevos de Kickstarter.

2.1.1.4. Resultados obtenidos

Las 4 variables más importantes encontradas del algoritmo clasificador fueron el número de proyectos patrocinados por el creador, número de proyectos creados por el creador, si presenta o no video, monto de la meta del proyecto. Evaluando el modelo con la métrica de exactitud, se alcanzó el valor de 0.90 al 40 % de transcurrida la duración de la campaña.

2.1.2. Segundo antecedente: «The Language that Gets People to Give: Phrases that Predict Success on Kickstarter» ([Mitra & Gilbert, 2014](#))

Mitra y Gilbert realizaron un resumen de la conferencia «The 17th ACM conference on Computer supported cooperative work & social computing (CSCW '14)» del 15 al 19 de febrero del 2014 en Baltimore, Estados Unidos. Este fue titulado «The Language that Gets People to Give: Phrases that Predict Success on Kickstarter» la cual traducida al español significa «El lenguaje que hace que la gente dé: Frases que predicen el éxito en Kickstarter».

2.1.2.1. Planteamiento del Problema y objetivo

Existencia de poco conocimiento sobre los factores que impulsan a financiar proyectos a pesar del crecimiento en los últimos años de plataformas de financiamiento colectivo como Kickstarter.

Los autores elaboraron, ante ello, un modelo para predecir el éxito de financiamiento en proyectos crowdfunding a partir de frases textuales.

2.1.2.2. Técnicas empleadas por los autores

Se diseñó un modelo de Regresión logística penalizada para proteger contra la colinealidad y escasez que prevalecen en el conjunto de datos de frases.

2.1.2.3. Metodología empleada por los autores

Se recolectaron 45,815 proyectos de Kickstarter del 2012. A continuación, se realizó limpieza de datos eliminando campañas inconclusas y con esto se procedió a extraer la información textual (descripción y recompensas del proyecto). El siguiente paso fue la generación del cuerpo de más de 9 millones de frases únicas, de las cuales quedaron 20,391 frases que aparecen al menos 50 veces en todos los proyectos. Finalmente, se creó el modelo predictivo para encontrar las frases que determinen que un proyecto será financiado y aquellas que determinen lo contrario.

2.1.2.4. Resultados obtenidos

De las 59 variables de control que no son texto, 29 que tenían «valores aceptables» tenían pesos positivos. 15 variables fueron consideradas para proyectos que serán financiados, mientras que 14 para los que no. Se obtuvo un Top 100 de frases que tenían pesos positivos (proyectos que sí serán financiados), así como también para aquellas con pesos negativos (proyectos que no serán financiados). El ratio de error del modelo propuesto (variables de control más frases) fue de 2.24.

2.1.3. Tercer antecedente: «Money Talks: A Predictive Model on Crowdfunding Success Using Project Description» ([M. Zhou, Zhang, Wang, Du, Qiao & Fan, 2015](#))

M. Zhou y col. realizaron un resumen de la conferencia «Twenty-first Americas Conference on Information Systems» en Puerto Rico en el año 2015. Este fue titulado «Money Talks: A Predictive Model on Crowdfunding Success Using Project Description» la cual traducida al español significa «Money Talks: un modelo predictivo sobre el éxito del crowdfunding utilizando la descripción del proyecto».

2.1.3.1. Planteamiento del Problema y objetivo

Las investigaciones existentes de crowdfunding se centran principalmente en las variables básicas del proyecto como la categoría y la meta; sin embargo, son muy pocos estudios basados en el contenido de la información, es decir, en la descripción del mismo.

El objetivo de los autores fue estudiar la influencia y el impacto del uso de descripciones de proyectos en su éxito de financiación para predecirlo.

2.1.3.2. Técnicas empleadas por los autores

Se desarrolló un modelo de Regresión Logística usando variables previamente identificadas (meta del proyecto, duración del proyecto, si el creador tiene conexión a Facebook, número de amigos en Facebook del creador, si el proyecto incluye una imagen, si el proyecto incluye un video, número de recompensas, año de lanzamiento del proyecto, categoría del proyecto) y nuevas introducidas (número de palabras en la descripción del proyecto, legibilidad de descripción del proyecto medido por Índice de niebla Gunning, ratio de positivo y negativo en descripción del proyecto, número de proyectos previamente creados y número de proyectos previamente patrocinados por el autor).

2.1.3.3. Metodología empleada por los autores

Se comenzó con la operacionalización de los constructos, es decir, cómo se consideraría la información recolectada a partir de los antecedentes en la literatura. El siguiente paso fue la recolección de información de más de 154 mil proyectos de Kickstarter entre 2009 y 2014. Y finalmente, se construyó el modelo predictivo y se comparó contra otros modelos de la base y convencionales.

2.1.3.4. Resultados obtenidos

Todos los modelos (base, convencional y propuesto) fueron evaluados por el valor-F probando en validaciones cruzadas de 3, 5 y 10 iteraciones. Bajo esta métrica, el mejor modelo fue el propuesto, alcanzando un valor-F de 71.17 con 10 iteraciones.

2.1.4. Cuarto antecedente: «Will your Project get the Green light? Predicting the success of crowdfunding campaigns» ([S.-Y. Chen, Chen, Chen, Yang & Lin, 2015](#))

S.-Y. Chen y col. realizaron un resumen de la conferencia «Pacific Asia Conference on Information Systems (PACIS) 2015» en New York, Estados Unidos, en el año 2015. Este fue titulado «Will Your Project Get the Green Light? Predicting the Success of Crowdfunding Campaigns» la cual traducida al español significa «¿Tu proyecto obtendrá la luz verde? Prediciendo el éxito de campañas de financiamiento colectivo».

2.1.4.1. Planteamiento del Problema y objetivo

Desde el surgimiento de campañas de financiamiento colectivo en sitios web así como estudios de estos casos, la mayoría de estos presentan problemas en la predicción de éxito de una campaña por distintas casuísticas ya que suelen enfocarse en aquellos concluidos. ¿Será posible desarrollar una técnica efectiva para predecir si una campaña será exitosa o no en diferentes momentos de tiempo de las campañas de crowdfunding?

Para responder esta interrogante, los autores desarrollaron una técnica efectiva para predecir si una campaña de crowdfunding tendrá éxito o fracasará a través de extracción y posterior uso de características estáticas y dinámicas.

2.1.4.2. Técnicas empleadas por los autores

Los autores construyeron una serie de 8 modelos de Bosque Aleatorio para diferentes etapas (puntos de tiempo) de la campaña desde el día 0 hasta el día 7.

2.1.4.3. Metodología empleada por los autores

Se recolectaron más de 4 mil proyectos de Kickstarter obtenido directamente desde el sitio, así como con ayuda de la página web Kickspy, entre el primer y último día de abril del 2014. Una vez pre-procesada la data, se extrajeron las características para la tarea de predicción de objetivos. Luego, estas características se clasificaron en 5 categorías: características intrínsecas, mecanismo financiero, calidad y sentimiento del contenido, interacción social y efecto de progresión, en donde las 3 primeras + interacción social corresponden a un nuevo grupo asignado como «características estáticas», mientras que la última + interacción social se asignó como «características dinámicas». Ambos nuevos conjuntos agrupados finalmente fueron entrenados con la serie de modelos para diferentes etapas.

2.1.4.4. Resultados obtenidos

Se comparó la serie de modelos propuestos considerando todas las características con modelos que solo consideraban algunas, así como también comparado con uno de la literatura. El resultado final de la evaluación medidos por la exactitud determinó que la primera opción mencionada logró la mejor performance, con un valor de 0.8467.

2.1.5. Quinto antecedente: «Predicting Success in Equity Crowdfunding» (Beckwith, 2016)

Beckwith realizó un artículo de investigación el cual fue publicado en la revista «Joseph Wharton Scholars» de la Universidad de Pensilvania en el año 2016. Este fue titulado «Predicting Success in Equity Crowdfunding» la cual traducida al español significa «Prediciendo el éxito en el financiamiento colectivo de acciones».

2.1.5.1. Planteamiento del Problema y objetivo

El financiamiento colectivo o crowdfunding de inversión está haciéndose cada vez más popular. Este concepto permite que los emprendedores ofrezcan algún tipo de producto o servicio como compensación por contribuciones financieras una vez que ya se tengan ventas reales o acuerdos comerciales, a diferencia por ejemplo del crowdfunding de recompensas que ofrece algo a sus patrocinadores desde la concepción del proyecto. Resulta ser muy interesante para los patrocinadores ya que no se necesita contar con un capital muy alto. A cambio, ellos recibirán algo a cambio una vez que el proyecto se realice. Este factor permite su mayor probabilidad de

éxito de financiamiento al momento de darse una campaña de este tipo de crowdfunding ya que los patrocinadores pueden invertir en más de un proyecto a la vez. A partir de este punto nace la siguiente cuestión: “Dados diferentes start-ups con similares características observables, ¿qué motiva a pequeños patrocinadores a invertir en ciertos start-ups y no en otros?”.

Por ello, el objetivo del autor fue determinar la relación entre las características de una compañía determinada y su capacidad para recaudar fondos en la plataforma de financiamiento colectivo de capital AngelList.

2.1.5.2. Técnicas empleadas por los autores

El autor desarrolló un modelo de Regresión logística usando las siguientes variables: si la empresa previamente había recibido o no financiación, si la compañía cuenta con perfil en Twitter, número de veces que la compañía había sido mencionada en alguna publicación, número de veces que la compañía había sido mencionada en TechCrunch, número de personas listadas como co-fundadoras en el perfil de la compañía AngelList, si AngelList lista entre 11 y 50 empleados como tamaño de la empresa, si la compañía está ubicada en San Francisco, si entre los fundadores de AngelList al menos uno de ellos cuenta con un MBA, si entre los fundadores de AngelList al menos uno de ellos realizó sus estudios en alguna de las mejores 20 universidades de Estados Unidos, y el número de cierre del S&P 500 el día anterior al lanzamiento de la campaña de crowdfunding de AngelList. Este modelo se comparó con un Árbol de Decisión CART, un modelo de Naïve Bayes y una Máquina de Vectores de Soporte.

2.1.5.3. Metodología empleada por los autores

Inicialmente, se recolectaron más de 5 mil compañías de AngelList que solicitaron contribuciones financieras. De esta cantidad, se consideraron aquellas con datos completos (2,603 empresas). Luego del pre-procesamiento del conjunto final de datos, se elaboró el modelo y se evaluó el modelo mediante la exactitud, precisión, sensibilidad, puntaje F1, y área bajo la curva (AUC).

2.1.5.4. Resultados obtenidos

Bajo las métricas mencionadas, el modelo propuesto por los autores superó a los otros 3 modelos comparados en cada una, tomando como resultados 0.87 de exactitud, precisión promedio de 0.85, sensibilidad promedio de 0.88, puntaje F1 promedio de 0.86, y área bajo la curva de 0.74.

2.1.6. Sexto antecedente: «Project Success Prediction in Crowdfunding Environments» (Y. Li, Rakesh & Reddy, 2016)

Y. Li y col. realizaron un resumen de la conferencia «The Ninth ACM International Conference on Web Search and Data Mining (WSDM' 16)» en San Francisco, Estados Unidos, en el año 2016. Este fue titulado «Project Success Prediction in Crowdfunding Environments» la cual traducida al español significa «Predicción de éxito de un proyecto en ambientes de financiamiento colectivo».

2.1.6.1. Planteamiento del Problema y objetivo

Durante la última década se han elaborado una serie de modelos de clasificación que permitan pronosticar con cierto nivel de exactitud si algunos proyectos de financiamiento colectivo tendrán éxito o no. Sin embargo, el hecho de estimar si esto ocurrirá durante el plazo dado de la campaña no puede proporcionar una guía adecuada a los patrocinadores que desean invertir en proyectos populares. Es entonces que se cuestiona la posibilidad de predecir el éxito de campañas en sitios web de crowdfunding como Kickstarter considerando características obtenidas durante la operación de la campaña.

Para ello, los autores formularon la predicción del éxito del proyecto como un problema de análisis de supervivencia y aplicar el enfoque de regresión censurada.

2.1.6.2. Técnicas empleadas por los autores

Los autores desarrollaron 2 modelos para evaluar la regresión censurada: 1 modelo de distribución logística y 1 de distribución log-logística. Para comparar con sus modelos, también se elaboraron otros métodos para manejar observaciones censuradas, mencionados en la literatura como Cox, Regresión Tobit, estimación Buckley-James e Índice de Concordancia Boosting (BoostCI).

2.1.6.3. Metodología empleada por los autores

Inicialmente, se recolectaron más de 27 mil proyectos de Kickstarter entre diciembre del 2013 y junio del 2014 para la data estática, a los cuales se removieron los cancelados, suspendidos o con menos de 1 patrocinador y monto prometido de \$100. Asimismo, para la data dinámica se capturaron más de 106 mil tweets de Twitter que contenían el enlace rápido

hacia la página de Kickstarter. Una vez obtenidos los conjuntos de datos, se pre-procesaron para finalmente elaborar los modelos predictivos.

2.1.6.4. Resultados obtenidos

Como resultados de las múltiples pruebas en los modelos, en los cuales los autores experimentaron distintas combinatorias con la data estática y la dinámica, así como agregando y desagregando proyectos fracasados, se obtuvo que el modelo de Regresión log-logística evaluado con el área bajo la curva (AUC) presentó mejor performance que el resto en 3 de las 4 combinaciones, alcanzando su mejor nivel en el conjunto de datos que combina data estática, dinámica, de características de proyectos transcurridos los primeros 3 días y con proyectos fracasados, con un puntaje Survival AUC de 0.9030.

2.1.7. Séptimo antecedente: «The Determinants of Crowdfunding Success: A Semantic Text Analytics Approach» ([H. Yuan, Lau & Xu, 2016](#))

H. Yuan y col. realizaron un artículo de investigación el cual fue publicado en la revista «Decision Support Systems» en el año 2016. Este fue titulado «The Determinants of Crowdfunding Success: A Semantic Text Analytics Approach» la cual traducida al español significa «Los determinantes del éxito del financiamiento colectivo: Un enfoque de analítica semántica de texto».

2.1.7.1. Planteamiento del Problema y objetivo

Actualmente existen diversos estudios de éxito de campañas de financiamiento colectivo, la mayoría usando modelos predictivos en proyectos crowdfunding basados en recompensas y análisis de características poco profundas del lenguaje en descripción de los proyectos (por ejemplo, número de palabras, errores ortográficos, etc) así como en la implementación de métodos de Aprendizaje Automático.

Como propuesta alterna, los autores construyeron un marco de trabajo basado en análisis de texto que pueda extraer semánticas de descripciones textuales de proyectos para predecir sus resultados de recaudación de fondos.

2.1.7.2. Técnicas empleadas por los autores

Se creó un modelo de Bosque Aleatorio para los datos numéricos, y POS Tag Stemming más modelo DC-LDA más características tópicas para los datos textuales. Esta arquitectura se comparó con otros modelos en la literatura de la publicación.

2.1.7.3. Metodología empleada por los autores

Se recolectaron 500 proyectos de 2 webs chinas sobre crowdfunding cada una. Luego, se construyó el framework de análisis textual. A continuación, se desarrolló un modelo DC-LDA para una extracción más efectiva de las características tópicas a partir de las descripciones. Y finalmente, se realizó un análisis empírico para identificar características discriminatorias que influye en el éxito de la recaudación de fondos.

2.1.7.4. Resultados obtenidos

Para los datos numéricos, la mejor performance se obtuvo evaluando el modelo con la sensibilidad para el Random Forest (0.98 en la segunda data); mientras que para los datos textuales, el modelo DC-LDA (al ser comparado con el LDA normal) obtuvo un puntaje F1 de 0.91.

2.1.8. Octavo antecedente: «Using Language to Predict Kickstarter Success» ([Sawhney, Tran & Tuason, 2016](#))

Sawhney y col. realizaron la publicación «Using Language to Predict Kickstarter Success» para el Departamento de Ciencia de la Computación de la Universidad Standford. Este fue titulado *Using Language to Predict Kickstarter Success* la cual traducida al español significa «Uso del lenguaje para predecir el éxito de Kickstarter».

2.1.8.1. Planteamiento del Problema y objetivo

Dada la considerable cantidad de investigaciones y trabajos de predicción de éxito financiamiento de proyectos basados en la evolución de variables estáticas, los autores construyeron un clasificador binario que logre predecir el éxito de una campaña a partir de su contenido, características lingüísticas y metadata.

2.1.8.2. Técnicas empleadas por los autores

Los autores utilizaron el Clasificador de Naive Bayes para las características primarias, Parts-of-Speech Tagging (POS), legibilidad Flesch-Kincaid, y Máquina de Vectores de Soporte (SVM) para los conjuntos de entrenamiento y prueba.

2.1.8.3. Metodología empleada por los autores

Se recolectaron más de 160 mil proyectos de Kickstarter, separando en proporciones de 0.80/0.20 los subconjuntos de entrenamiento y prueba respectivamente. A continuación, recopilaron las características primarias y se realizó un análisis de sentimientos. Luego, asignaron las meta-características para utilizarlos como entradas durante el entrenamiento de los modelos. Finalmente, estos últimos fueron calibrados y se probaron con el subconjunto de prueba para finalmente ser evaluados.

2.1.8.4. Resultados obtenidos

Considerando el modelo solo con las características primarias, el nivel de exactitud alcanzó 0.65. Sin embargo, utilizando el clasificador entrenado con SVM, el valor de la exactitud logró llegar hasta 0.92.

2.1.9. Noveno antecedente: «Effect of Social Media Connectivity on Success of Crowdfunding Campaigns» ([Kaur & Gera, 2017](#))

Kaur y Gera realizaron un artículo de investigación el cual fue publicado en la revista «Procedia Computer Science» en el año 2017. Este fue titulado «Effect of Social Media Connectivity on Success of Crowdfunding Campaigns» la cual traducida al español significa «Efecto de conectividad en medios sociales en éxito de campañas de financiamiento colectivo».

2.1.9.1. Planteamiento del Problema y objetivo

Los proyectos de crowdfunding, así como los proyectos lanzados en sitios dedicados a esto, han crecido exponencialmente en los últimos años. Los medios sociales desempeñan un papel importante en la difusión de palabras sobre una campaña y en la obtención de fondos con éxito. Después de lanzarse una campaña, su éxito se define por la interacción social de los creadores en la plataforma y en las redes sociales. Además, el tamaño de la red social del

creador y su presencia en línea motiva a los patrocinadores a participar y financiar el proyecto. Partiendo de esta premisa, se busca conocer el impacto de la conectividad con las redes sociales y las interacciones sociales en el rendimiento del crowdfunding.

Para ello, los autores plantearon el desarrollo de un modelo predictivo que comprenda las características de la campaña y sumado a eso, la información sobre la conectividad con las diversas redes sociales como Facebook y Twitter.

2.1.9.2. Técnicas empleadas por los autores

Los autores propusieron un modelo de Regresión Logística, el cual compararon con algunos enunciados en sus antecedentes como Naïve Bayes, J48 y Bosques Aleatorios.

2.1.9.3. Metodología empleada por los autores

Se recolectaron 2 tipos de conjuntos de datos: el primero obtenido desde Kickstarter en el mes de Abril del 2014, con más de 4 mil campañas; mientras que el segundo a través del perfil del creador vinculado a redes sociales. El primer conjunto comprendió las variables de categoría, tipo de divisa, monto de la meta, monto prometido, número de recompensas, duración de la campaña en días, y el estado final de financiamiento (exitoso o fracasado). Por el lado del segundo conjunto, además de obtener marcas de hacia qué redes sociales el creador se conecta, mediante el enlace se extrajeron más de 19 mil tweets en el que se promocionaron las campañas gracias a la API de Twitter.

Una vez armados los conjuntos de datos, se creó el modelo propuesto y se comparó con las bases de referencia usando como métrica principal la exactitud. Previo a este paso, todas las variables consideradas se evaluaron en una matriz de correlaciones con el valor de “exitoso” de la variable dependiente.

2.1.9.4. Resultados obtenidos

Culminados los experimentos de la selección de variables en la matriz de correlaciones, así como la ejecución del modelo de Regresión Logística, la exactitud de este alcanzó un valor de 0.767 para un subconjunto de 66 % de entrenamiento y de 34 % de prueba, indicando así que la conectividad a medios sociales e interacciones en ellos influye en la predicción de éxito de la campaña. Asimismo, evaluando también con precisión y sensibilidad promedio, el ratio alcanzó 0.768 y 0.767 respectivamente.

2.1.10. Décimo antecedente: «Supervised Learning Model For Kickstarter Campaigns With R Mining» ([R. S. Kamath & Kamat, 2018](#))

R. S. Kamath y Kamat realizaron un artículo de investigación el cual fue publicado en la revista «International Journal of Information Technology, Modeling and Computing» en el año 2018. Este fue titulado «Supervised Learning Model For Kickstarter Campaigns With R Mining» la cual traducida al español significa «Modelo de aprendizaje supervisado para campañas de Kickstarter con R Mining».

2.1.10.1. Planteamiento del Problema y objetivo

Kickstarter actualmente es la plataforma web basado en crowdfunding más grande existente. Sin embargo, no todas las campañas que se encuentran en su portal son exitosas. Algunos trabajos previos citados en este artículo citan que el lenguaje usado en las campañas alcanza el poder predictivo de 58.86 %, es decir, la descripción semántica de un proyecto ayuda considerablemente en la predicción de éxito debido a que muchos patrocinadores se fijan en la calidad presente en una campaña antes de invertir. Sin embargo, es dejada de lado en muchos trabajos de investigación destinados a la predicción de éxito de financiamiento, así como también las interacciones de un proyecto en redes sociales. Para el tiempo en que este artículo fue publicado, el nivel de exactitud de los modelos predictivos ya existentes bordeaba por el 68 % como el descrito anteriormente

Ante esto, autores desarrollaron un sistema con técnicas de Aprendizaje Automático usando el entorno R aplicada a conjunto de datos de campañas en Kickstarter para clasificar proyectos entrenando diferentes clasificadores en esta data y predecir con un alto nivel de exactitud.

2.1.10.2. Técnicas empleadas por los autores

Los autores elaboraron 5 modelos clasificadores para la investigación: Árbol de decisión, Bosque Aleatorio, Red Neuronal Artificial, Vecino K más cercano, y Naïve Bayes. Se consideraron como variables para estos modelos al nombre del proyecto, URL del proyecto, descripción del proyecto, categoría, sub-categoría, número de patrocinadores, monto total patrocinado, meta de financiación, fecha de lanzamiento del proyecto, duración del proyecto, número de actualizaciones, número de recompensas, si el proyecto cuenta con video, localización del proyecto, y creador del proyecto.

2.1.10.3. Metodología empleada por los autores

En primer lugar, se analizó y definió el problema. A continuación, se recolectaron 120 proyectos de Kickstarter a través de su URL. Luego, las variables categóricas fueron transformadas en numéricas como parte del pre-procesamiento. Esto permitió realizar extracción y exploración del conjunto final mediante estadísticas descriptivas con paquetes de R. Una vez analizado los datos, se procedió a elaborar los modelos clasificadores y finalmente, con la matriz de confusión, se seleccionó el mejor de ellos evaluándolos mediante la exactitud.

2.1.10.4. Resultados obtenidos

Bajo la métrica de exactitud, el mejor de los 5 modelos clasificadores construidos fue la Red Neuronal, con un valor de 0.94.

2.1.11. Undécimo antecedente: «Prediction of Crowdfunding Project Success with Deep Learning» ([P.-F. Yu, Huang, Yang, Liu, Li & Tsai, 2018](#))

P.-F. Yu y col. realizaron un resumen de la conferencia «2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)» del 12 al 14 de octubre del 2018 en Xi'an, China. Este fue titulado «Prediction of Crowdfunding Project Success with Deep Learning», la cual traducida al español significa «Predicción del éxito de proyecto de financiamiento colectivo con Aprendizaje Profundo».

2.1.11.1. Planteamiento del Problema y objetivo

Debido al aumento dramático de la actividad de financiamiento colectivo en los últimos años, en el cual tanto creadores como patrocinadores participan, son más las campañas que buscan alcanzar su objetivo. Sin embargo, según el análisis empírico, solo la tercera parte lo logra. La incógnita que rodea a esta situación es la posibilidad de elaborar un modelo predictivo de éxito en proyectos de Kickstarter utilizando distintas técnicas al Aprendizaje Automático.

Por ello, los autores plantean como objetivo el desarrollo de un modelo que prediga el éxito del proyecto de crowdfunding con Aprendizaje Profundo usando registros históricos de campañas en Kickstarter.

2.1.11.2. Técnicas empleadas por los autores

Los autores elaboraron un modelo basado en un Perceptrón Multicapa (MLP), alimentado por 6 variables visibles de la campaña (metadata). Además, sus resultados se compararon con otros modelos considerados en sus antecedentes como Bosques aleatorios, AdaBoost, Máquina de vectores de soporte, Árboles de decisión, Regresión logística y Naïve Bayes.

2.1.11.3. Metodología empleada por los autores

Se recolectaron más de 378 mil campañas de Kickstarter entre Mayo 2009 y Marzo 2018, recopilados retrospectivamente desde Kaggle. A continuación, realizaron análisis exploratorio de los datos para entender la data y seleccionar variables, y finalmente construyeron un modelo, el cual se comparó sus resultados con los de otros modelos.

2.1.11.4. Resultados obtenidos

Para comparar los resultados del modelo propuesto con los antecedentes, se tomaron en cuenta como métricas la exactitud y el área bajo la curva (AUC). De acuerdo a ellas, para ambos escenarios se lograron mejor performance, con una exactitud de 0.9320 y un área bajo la curva de 0.9323 frente al mejor modelo de los antecedentes, Bosques Aleatorios, que obtuvo como resultados 0.9293 y 0.9272 para las mismas métricas respectivamente.

2.1.12. Duodécimo antecedente: «Content-based Success Prediction of Crowdfunding Campaigns: A Deep Learning Approach» ([S. Lee, Lee & Kim, 2018](#))

S. Lee y col. realizaron un resumen de la conferencia «Companion of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing» del 3 al 7 de noviembre del 2018 en Jersey City, Estados Unidos. Este fue titulado «Content-based Success Prediction of Crowdfunding Campaigns: A Deep Learning Approach» la cual traducida al español significa «Predicción de éxito basada en contenido de campañas de crowdfunding: un enfoque de Aprendizaje Profundo».

2.1.12.1. Planteamiento del Problema y objetivo

El ratio promedio de éxito de financiamiento colectivo ha ido decreciendo en los últimos años a pesar del éxito de plataformas basadas en este tipo de apoyo a proyectos. Asimismo, los desafíos más relevantes fueron descubrir las razones de su éxito y lograr predecirlo.

Los autores plantearon utilizar solo el contenido textual de los proyectos (descripción de campaña, actualizaciones, comentarios y palabras obtenidas del video principal) para predecir el estado de financiamiento de proyectos de Tecnología (la categoría con ratio más bajo de éxito de financiamiento y a la vez, con los mayores montos financiados en la plataforma) a través de la creación de una Red Neuronal Profunda.

2.1.12.2. Técnicas empleadas por los autores

Se diseñó 1 Modelo de Secuencia a Secuencia (Seq2seq) con atención a nivel de oración y 1 Modelo de Atención Jerárquica (HAN).

2.1.12.3. Metodología empleada por los autores

Se recolectaron 216,136 proyectos de crowdfunding en Kickstarter, lanzados entre abril del 2009 y agosto del 2017. De ellos, se filtraron aquellos de la categoría Tecnología (22,982) para luego separar el conjunto de datos final en 0.80 (entrenamiento), 0.10 (validación) y 0.1 (prueba). Posteriormente, se diseñaron los 2 modelos de NLP, fueron entrenados y se compararon los resultados.

2.1.12.4. Resultados obtenidos

Ambos modelos fueron evaluados bajo la métrica de exactitud, bajo enfoques de distintos experimentos que alternaban la presencia de las variables. De todos ellos, basado en aquellos que utilizaron todas juntas, el modelo de Seq2seq + atención logró una exactitud de 0.80, mientras que el modelo HAN alcanzó el valor de 0.60.

2.1.13. Decimotercer antecedente: «Estimating the Days to Success of Campaigns in Crowdfunding: A Deep Survival Perspective» (Jin, Zhao, Chen, Liu & Ge, 2019)

Jin y col. realizaron un resumen de la conferencia «The 33rd AAAI Conference on Artificial Intelligence (AAAI'2019)» del 27 de enero al 1 de febrero del 2019 en Honolulu, Hawái. Este fue titulado «Estimating the Days to Success of Campaigns in Crowdfunding: A Deep Survival Perspective» la cual traducida al español significa «Estimación de días de éxito de campañas en financiamiento colectivo: Una perspectiva de supervivencia profunda».

2.1.13.1. Planteamiento del Problema y objetivo

Dado el incremento de actividad en campañas de crowdfunding en sitios como Kickstarter o Indiegogo, los estudios sobre este tema también han ido de la mano. Sin embargo, la mayoría de investigaciones se enfocan en predecir el ratio de éxito de financiamiento en una campaña. El reto de predecir el tiempo de duración es más complicado porque se ve afectada por la variabilidad de la metadata y los comentarios del proyecto, resultando un alto nivel de fracaso (60 %) en la financiación de proyectos.

Los autores plantean la predicción de la fecha exacta de finalización de una campaña, así como controlar la distribución de patrocinios conforme a la evolución del tiempo.

2.1.13.2. Técnicas empleadas por los autores

Se elaboró un modelo Seq2seq (encoder + decoder) con previos multifacéticos (SMP) para la distribución de patrocinios y el tiempo de éxito, así como también un modelo evolutivo lineal para mantener el cambio de las distribuciones de patrocinios.

2.1.13.3. Metodología empleada por los autores

Se recolectó más de 14 mil proyectos de Indiegogo con información estática (descripción de campaña, descripción de las recompensas, categoría de la campaña, tipo del creador, duración declarada de financiación, meta declarada prometida, número de recompensas, precio máximo/mínimo/promedio de recompensas, plazo máximo/mínimo/promedio de entrega) y dinámica (comentarios), luego se definió el problema estudiado. A continuación, se introdujo los detalles técnicos de SMP, el conjunto pre-procesado de entrada, la arquitectura SMP, los previos multifacéticos y por último, el método de entrenamiento del modelo.

2.1.13.4. Resultados obtenidos

Respecto a la predicción de distribución de patrocinios, y evaluando mejor con RECM, los modelos propuestos de SMP fueron los más efectivos (0.135 para 70 % de ratio de participación) para el Encoder. Respecto a la predicción del tiempo de éxito, los modelos propuestos de SMP (el mejor con 0.96 para 50 % de ratio de participación) fueron mejores que modelos de referencia para el Decoder.

2.1.14. Decimocuarto antecedente: «Success Prediction on Crowdfunding with Multimodal Deep Learning» ([Cheng, Tan, Hou & Wei, 2019](#))

Cheng y col. realizaron un resumen de la conferencia «The Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)» del 10 al 16 de agosto del 2019 en Macao, China. Este fue titulado «Success Prediction on Crowdfunding with Multimodal Deep Learning» la cual traducida al español significa «Predicción del éxito en financiamiento colectivo con Aprendizaje Profundo Multimodal».

2.1.14.1. Planteamiento del Problema y objetivo

La mayoría de los enfoques de predicción existentes aprovechan solo la modalidad dominada por el texto a pesar de existir más información en los perfiles de los proyectos, como por ejemplo, metadatos e imágenes. A estos últimos se han realizado poco trabajo para evaluar sus efectos hacia la predicción del éxito. Además, la metainformación ha sido explotada en muchos enfoques existentes para mejorar la precisión de la predicción. Sin embargo, esta generalmente se limita a la dinámica después de la publicación de los proyectos, haciendo que tanto los creadores del proyecto como las plataformas no puedan predecir el resultado de manera oportuna. Se carecen estudios basados en distintas modalidades más allá de la metainformación.

El objetivo de los autores es construir esquemas avanzados de redes neuronales que combinan información de diferentes modalidades para predecir el estado de financiamiento del proyecto usando solo información pre-publicación.

2.1.14.2. Técnicas empleadas por los autores

Debido a que los autores construyeron un modelo apilado de 3 modelos para cada modalidad, cada una se distribuyó de la siguiente manera: un modelo de Máquina de Vectores de Soporte (SVM) para la metainformación; la Red Neuronal Convolutacional (CNN) pre-entrenada

de 16 capas VGG16, trabajando junto con la Bolsa de Palabras Visuales (BoVW) para las imágenes; y la Bolsa de Palabras (BoW) trabajando junto con el método Frecuencia de Términos - Frecuencia Inversa de Documentos (TF-IDF), con modelo de incrustaciones de palabras GloVe de 300 dimensiones.

2.1.14.3. Metodología empleada por los autores

Se recolectaron más de 20 mil campañas finalizadas (exitosas o fracasadas) en Kickstarter gracias a algoritmo creado para captura de datos en la página por 2 semanas. Se descartaron proyectos previos al 2015, resultando así proyectos entre este año y el 2018. Asimismo, entre el contenido obtenido se registraron perfil textual (título, resumen, descripción del financiamiento, y riesgos/retos), imágenes (en formato jpg), categorías, meta de financiamiento, fecha de inicio y de finalización de campaña. Luego, se realizó pre-procesamiento a los subconjuntos de datos y se dividieron en 3 subconjuntos: entrenamiento (proyectos del 2015 y 2016), validación (2017) y prueba (2018). A continuación, se configuró la arquitectura del marco de trabajo del modelo Aprendizaje Profundo Multimodal y finalmente se realizaron distintos experimentos elaborando combinatorias de distintas modalidades para compararlas entre sí mediante las métricas de sensibilidad, precisión, puntaje F1 y área bajo la curva (AUC).

2.1.14.4. Resultados obtenidos

Realizando una comparación entre la base de referencia de un modelo SVM con texto, imágenes y metadata (SVM-TIM), contra el modelo propuesto (MDL-TIM) por los autores, se observó que bajo todas las métricas, con excepción de la precisión, el marco de trabajo Multimodal obtuvo mejores resultados, con 0.7505 en sensibilidad, 0.7534 en puntaje F1 y 0.8326 en área bajo la curva, frente a 0.7411, 0.7483 y 0.7411 respectivamente.

2.1.15. Decimoquinto antecedente: «[Finding the Keywords Affecting the Success of Crowdfunding Projects](#)» ([L.-S. Chen & Shen, 2019](#))

L.-S. Chen y Shen realizaron un resumen de la conferencia «2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA)» del 12 al 15 de abril del 2019 en Tokyo, Japón. Este fue titulado «[Finding the Keywords Affecting the Success of Crowdfunding Projects](#)» la cual traducida al español significa «Encontrar las palabras clave que influyen en el éxito de los proyectos de financiación colectiva».

2.1.15.1. Planteamiento del Problema y objetivo

A la fecha de publicación del acta de congreso del presente antecedente, existían pocas investigaciones basadas en el impacto de la descripción y palabras para predecir el ratio de éxito de financiamiento de un proyecto crowdfunding.

Ante esta situación, los autores tuvieron como objetivo analizar el impacto del contenido textual en un proyecto de Kickstarter a partir del análisis de sus palabras clave que determinen el ratio de éxito de su financiamiento.

2.1.15.2. Técnicas empleadas por los autores

Los autores propusieron 2 modelos de Máquinas de Vectores de Soporte con Eliminación de Características Recursivas (SVM-RFE), usando las herramientas “Weka 3.8” y “libSVM”, los cuales fueron comparados.

2.1.15.3. Metodología empleada por los autores

Se recolectaron las descripciones de proyectos de Kickstarter de la categoría “Juegos”, desde el 1 de julio hasta el 30 de agosto del 2018, de los cuales la distribución quedó en 50 proyectos que alcanzaron el 100 % de su meta y otros 50 proyectos que no alcanzaron por lo menos el 75 %. A continuación, se pre-procesó la data eliminando caracteres especiales, términos distintos al idioma inglés y palabras de parada en inglés para poder armar una matriz de Término-Documento que usará los pesos TF-IDF. Luego, se seleccionaron las características realizando experimentos de validación cruzada antes de la construcción del modelo predictivo. Finalmente, se evaluaron los resultados con la métrica de exactitud.

2.1.15.4. Resultados obtenidos

Se realizó hasta dos experimentos manipulando los conjuntos de datos para el SVM-RFE y el mejor resultado fue una exactitud de 78.90 % para predecir el ratio de éxito de un proyecto crowdfunding a partir de 50 palabras claves importantes que se encuentren dentro de su contenido.

2.1.16. Decimosexto antecedente: «Deploying Natural Language Processing to Extract Key Product Features of Crowdfunding Campaigns: The Case of 3D Printing Technologies on Kickstarter» (Chaichi & Anderson, 2019)

Chaichi y Anderson realizaron un resumen de la conferencia «2019 Portland International Conference on Management of Engineering and Technology (PICMET)» del 25 al 29 de agosto del 2019 en Portland, Estados Unidos. Este fue titulado «Deploying Natural Language Processing to Extract Key Product Features of Crowdfunding Campaigns: The Case of 3D Printing Technologies on Kickstarter» la cual traducida al español significa «Implementación del procesamiento del lenguaje natural para extraer características clave del producto de las campañas de crowdfunding: el caso de las tecnologías de impresión 3D en Kickstarter».

2.1.16.1. Planteamiento del Problema y objetivo

Debido a la dificultad para predecir éxito de campañas en Kickstarter a partir de información textual (data no estructurada), los autores propusieron predecir éxito de campañas de proyectos de impresión 3D en Kickstarter a partir de la extracción de características de producto a partir de información textual como título, resumen y descripción.

2.1.16.2. Técnicas empleadas por los autores

Para ello, principalmente usaron paquetes UDPipe y técnicas de extracción de palabras clave como TextRank R.

2.1.16.3. Metodología empleada por los autores

Se extrajo contenido textual (incluyendo título, resumen y descripción de campaña) de 528 proyectos de tecnología que contenían las palabras «3d printer» (impresora 3D) recopilados en Septiembre del 2017 usando la herramienta Selenium, de los cuales al eliminar aquellos que no contenían estas 3 variables textuales y algunos duplicados, quedaron finalmente 246 proyectos. El siguiente paso consistió en pre-procesar la data con algunas librerías de Python, incluyendo el paquete de R UDPipe, para armar el corpus final. UDPipe permitió generar segmentación de oraciones, tokenización, etiquetado POS (Part Of Speech), lematización, árboles de análisis de dependencia y colocaciones (palabras que continúan a otra). A continuación, se usó el algoritmo basado en grafos TextRank para extraer palabras claves, así como la técnica no

supervisada RAKE (Extracción automática rápida de palabras clave) para extraer frases clave. Se probaron, además, otras tres técnicas más. Finalmente, los resultados de las 6 técnicas para extraer frases claves fueron comparados y se llegaron a conclusiones basadas en las similitudes entre estas.

2.1.16.4. Resultados obtenidos

Se encontraron palabras con mayor frecuencia según cada una de las 6 técnicas de extracción de palabras clave, ilustradas en nubes de palabras, como por ejemplo relacionadas a disponibilidad, buenas características y precios cómodos de impresoras 3D. También se observaron 2 problemas: no todas las palabras con mayor frecuencia son características del producto, y ambigüedad de algunas palabras.

2.1.17. Decimoséptimo antecedente: «Topic Predictions and Optimized Recommendation Mechanism Based on Integrated Topic Modeling and Deep Neural Networks in Crowdfunding Platforms» ([Shafqat & Byun, 2019](#))

Shafqat y Byun realizaron un artículo de investigación el cual fue publicado en la revista «Applied Sciences» en el año 2019. Este fue titulado «Topic Predictions and Optimized Recommendation Mechanism Based on Integrated Topic Modeling and Deep Neural Networks in Crowdfunding Platforms» la cual traducida al español significa «Predicciones de temas y mecanismo de recomendación optimizado basado en modelos integrados de temas y redes neuronales profundas en plataformas de financiación colectiva».

2.1.17.1. Planteamiento del Problema y objetivo

Los comentarios de los usuarios al momento de evaluar algún producto de su interés en el comercio electrónico pasa desapercibido. Estas reseñas suelen ser importantes en las plataformas de financiamiento colectivo dado que influye en otros usuarios en patrocinar un proyecto.

Ante ello, los autores plantearon fusionar técnicas de modelado de lenguaje con redes neuronales para identificar patrones de discusión ocultos en los comentarios.

2.1.17.2. Técnicas empleadas por los autores

Se usó un híbrido basado en las redes neuronales recurrentes de Memoria Larga a Corto Plazo (LSTM), alimentados con la distribución de temas latentes aprendidos del modelo pre-entrenado Asignación Latente de Dirichlet (LDA). Además, se utilizó el algoritmo Optimización por Enjambre de Partículas (PSO) para optimizar las recomendaciones.

2.1.17.3. Metodología empleada por los autores

Los autores diseñaron un flujo para la elaboración del conjunto final de datos, basándose en exclusión de potenciales proyectos fraudulentos en Kickstarter según distintas fuentes y navegación en la web, análisis de contenido y otros criterios adicionales para la lista restante de enlaces de campañas, y finalmente extrayendo los comentarios en archivos de texto. Como resultado, se obtuvieron inicialmente más de 645 mil comentarios provenientes de 600 proyectos. Sin embargo, se redujeron a 504,184 comentarios (841 en promedio por proyecto) cuyo contenido era visible públicamente. El 70% fue destinado al subconjunto de entrenamiento, mientras que el resto, al de prueba. A continuación, se elaboró la arquitectura del modelo híbrido, la cual comienza recibiendo de entrada a los comentarios para crear los vectores de palabras en el tiempo t de una campaña. Como resultado del proceso LDA, estos generan vectores latentes de temas, que alimentan al modelo LSTM con el fin de que logre predecir la clase del tema de la siguiente palabra en el documento. Con estas nuevas salidas, el módulo de recomendación sugiere algunos proyectos basados en los temas descubiertos por el patrocinador. Finalmente, este modelo fue evaluado y comparado contra otros estudios de referencia.

2.1.17.4. Resultados obtenidos

Tanto el modelo propuesto (LSTM-LDA) como los modelos de referencia fueron evaluados con la métrica de exactitud para 300 épocas de entrenamiento. El modelo de los autores RNN-LDA logró la mejor performance, alcanzando una exactitud de aproximadamente 0.96, frente al 0.94 de una red NN-LDA y 0.92 de una red NN.

2.2. Bases Teóricas

2.2.1. Inteligencia Artificial

La Inteligencia Artificial es la inteligencia llevado a cabo por máquinas en las que una máquina “inteligente” ideal es un agente flexible que percibe su entorno y lleva a cabo acciones que maximicen sus posibilidades de éxito en algún objetivo ([Poole y col., 1998](#)). Este término se aplica cuando una máquina imita las funciones “cognitivas” que asocian los humanos con otras mentes ([Russell & Norvig, 2009](#)).

Durante la historia de la humanidad, se han seguido 4 enfoques: dos centrados en el comportamiento humano y dos enfocados en torno a la racionalidad. El enfoque centrado en el comportamiento humano se basa en una ciencia empírica, es decir, mediante experimentos que incluyen hipótesis y confirmaciones. Este enfoque nace a partir de la prueba de Alan Turing, en 1950, en la cual, el célebre matemático inglés diseñó una prueba basada en la incapacidad de diferenciar entre entidades inteligentes indiscutibles y seres humanos por parte de un computador. Si este era capaz de diferenciar y superar la prueba mientras que el humano no, se afirma que se trataba de una “máquina inteligente”. Por ello, el computador debía contar con las siguientes capacidades: procesamiento de lenguaje natural para poder comunicarse, representación del conocimiento describiendo lo que percibe de su entorno, razonamiento automático utilizando la información procesada en su interior, y aprendizaje automático para adaptarse a nuevos eventos. Si el evaluador decide incluir una señal de video para evaluar la percepción de la computadora, se dice que se está realizando la Prueba Global de Turing. Para superarla, además de las 4 anteriormente mencionadas, la computadora debe contar además con las capacidades de visión computacional para percibir objetos y robótica con el fin de manipularlos. Todas estas seis capacidades o disciplinas abarcan la mayor parte de la Inteligencia Artificial ([Russell & Norvig, 2004](#)).

Por el otro lado, el enfoque racional implica una combinación de ingeniería y matemáticas basándose en las “leyes del pensamiento”. Estas parten de la Grecia antigua, planteadas por grandes filósofos como Aristóteles en su intento de codificar la “manera correcta de pensar”, lo que más adelante derivó al estudio de la lógica. Más adelante, en el siglo XIX, se construyeron programas capaces de resolver problemas en notación lógica. De ahí que la tradición logista dentro del campo de la Inteligencia Artificial trata de construir sistemas inteligentes con estas capacidades. De todo lo anterior dicho respecto al enfoque racional se creó el término de un agente racional, el cual actúa intentando lograr el mejor resultado, o de existir incertidumbre, el mejor resultado esperado. Finalmente, la amplia aplicación de la Inteligencia Artificial y sus fundamentos derivan en muchas ciencias de las cuales se pueden mencionar, además de la filo-

sofía y las matemáticas, a la economía, neurociencia, psicología, la ingeniería computacional, la teoría de control y cibernetica, y hasta la lingüística ([Russell & Norvig, 2004](#)).

Pero, ¿cómo es surge este amplio estudio de la Inteligencia Artificial? En 1943, basándose en la fisiología básica y funcionamiento de las neuronas en el cerebro, el análisis formal de la lógica proposicional de Russell y Whitehead, y la teoría computacional de Turing, dos estudiantes en neurociencia realizaron juntos el que sería considerado primer trabajo de Inteligencia Artificial. Warren McCulloch y Walter Pitts propusieron un modelo constituido por neuronas artificiales, en el que cada una de ellas se caracterizaba por estar “activada” o “desactivada”; la del primer tipo daba como resultado a la estimulación producida por una cantidad suficiente de neuronas vecinas. Como ejemplo, mostraron que cualquier función de cómputo podría calcularse mediante alguna red de neuronas interconectadas y que todos los conectores lógicos eran capaces de ser implementados usando estructuras sencillas de red. Seis años más adelante, Donald Hebb propuso una regla de actualización de intensidades de conexiones entre las neuronas, la que actualmente se le conoce como la “regla de aprendizaje Hebbiano” vigente hasta nuestros días. En 1956, Allen Newell y Herbert Simon inventaron un programa de computación en el taller de Dartmouth de John McCarthy, que era capaz de pensar de forma no numérica, basado en el Teórico Lógico, artículo que, además, fue rechazado de ser publicado en la revista *Journal of Symbolic Logic*. A pesar de ello, los trabajos de los colaboradores presentes en dicho taller se mantuvieron por 20 años más, siendo McCarthy quien acuñó el término de “Inteligencia Artificial” a este campo ([Russell & Norvig, 2004](#)).

En la década de los años 80, la Inteligencia Artificial dio el gran salto de formar parte de la industria, en especial, de las compañías más grandes de los países desarrollados a través de grupos especializados para la realización de investigaciones de sistemas expertos, así como en la construcción de computadoras cada vez más potentes y capaces de resolver tareas más complejas.

Actualmente, la IA cuenta con muchas aplicaciones como la Minería de Datos, el procesamiento de lenguaje natural, la robótica, los videojuegos, entre otros. Dentro de ella se pueden encontrar otras ramas como por ejemplo el Aprendizaje Automático, Visión computacional, etcétera.

2.2.2. Aprendizaje Automático

El Aprendizaje Automático (*Machine Learning* por su nombre en inglés) es una rama de la Inteligencia Artificial cuyo fin es desarrollar técnicas que las computadoras pueden aprender a través de encontrar algoritmos y heurísticas que conviertan muestras de datos en programas

sin necesidad de hacerlos ([Russell & Norvig, 2009](#)). Sus algoritmos están compuestos por muchas tecnologías, como por ejemplo Aprendizaje Profundo, Redes Neuronales y Procesamiento de lenguaje natural, utilizadas en el aprendizaje supervisado y no supervisado, las cuales operan guiadas por lecciones de información existente ([Gartner, 2019a](#)). La premisa básica del aprendizaje automático es construir algoritmos que puedan recibir datos de entrada y usar análisis estadísticos para predecir una salida mientras se actualizan las salidas a medida que se dispone de nuevos datos ([Alpaydin, 2014](#)).

Los tres tipos de aprendizaje principales son:

- **Aprendizaje supervisado:** Se trabajan con datos etiquetados buscando obtener una función que asigne una respuesta de salida adecuada, denominadas etiquetas, a partir de unos datos de entrada denominadas características ([Zambrano, 2018](#)). Por lo general, los datos de entrada son conocidos como variables dependientes o X, mientras que los datos de salida son llamadas variables independientes o Y. Se le dice supervisado ya que el resultado depende de los datos que recibe de entrada, afectando su performance si estos son alterados.

Existen dos tipos de aprendizaje supervisado (ver Figura 2.1). El primero es la regresión, que consiste en obtener como resultado un número específico a partir de un conjunto de variables de las características; mientras que por otra parte está la clasificación, el cual se basa en encontrar distintos patrones ocultos para clasificar los elementos del conjunto de datos en diferentes grupos ([Zambrano, 2018](#)).

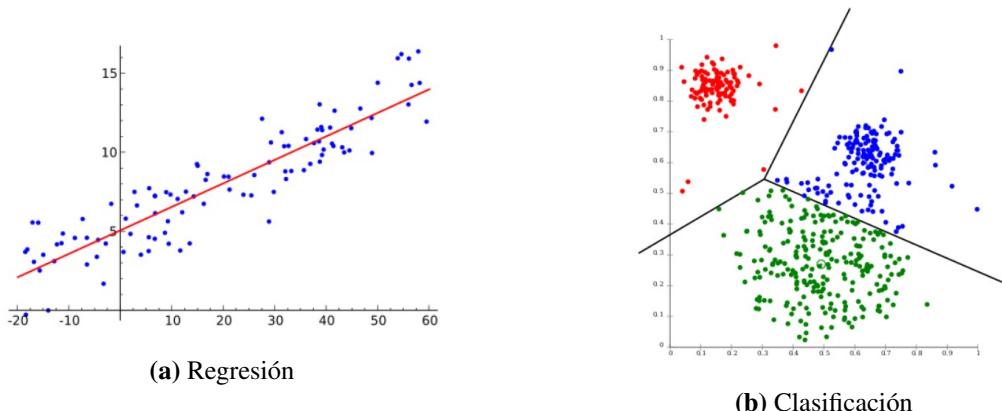


Figura 2.1: Ejemplo de algoritmos de regresión y clasificación. Fuente: [Zambrano, 2018](#)

Para el segundo tipo de aprendizaje supervisado, el algoritmo más usado es el de los K Vecinos más cercanos o *k-NN Nearest Neighbour* en inglés. Este se basa en la idea de que los nuevos ejemplos serán clasificados a la clase a la cual pertenezca la mayor cantidad de vecinos más cercanos del conjunto de entrenamiento más cercano a él. Sin embargo, el

número k de vecinos más cercanos lo decide el usuario, de preferencia impar, para evitar ambigüedad al momento de clasificar un registro por parte del algoritmo (esto puede ocurrir por las mismas distancias existentes entre dos o más registros). Otra variante aplicada consiste en la ponderación de cada vecino de acuerdo a la distancia entre él y el ejemplar a ser clasificado, asignando mayor peso a los más próximos ([Sancho Caparrini, 2018](#)). Para ello, se tiene la Ecuación 2.1 y su variante en la Ecuación 2.2:

$$W_i = \frac{1}{d(x, x_i)^2}$$

Ecuación 2.1: Cálculo de los pesos para el algoritmo K-NN mediante ponderación de sus distancias.

Fuente: [Sancho Caparrini, 2018](#)

$$\operatorname{argmax}_{v \in V} \sum_{i=1 \dots k, x_i \in v} W_i$$

Ecuación 2.2: Fórmula alternativa del algoritmo K-NN mediante sumatoria de pesos. Fuente: [Sancho Caparrini, 2018](#)

Donde:

x = ejemplo que se desea clasificar

V = posibles clases de clasificación

x_i = conjunto de los k ejemplos de entrenamiento más cercanos

y finalmente, la clase asignada a x es aquella que verifique que la suma de los pesos de sus representantes sea la máxima, representándose en la Figura 2.2:

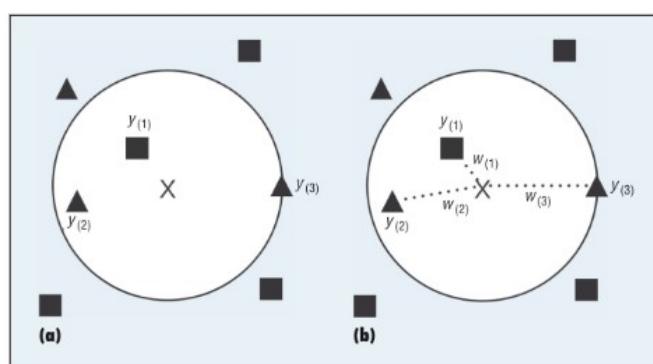


Figura 2.2: Algoritmo de K Vecinos más cercanos con pesos ponderados. Fuente: [Sancho Caparrini, 2018](#)

- **Aprendizaje no supervisado:** A diferencia de la anterior, aquí se trabaja con datos no etiquetados para entrenar el modelo, ya que el fin es de carácter exploratorio y descriptivo de la estructura de los datos. No existen variables independientes o Y.

La función es agrupar ejemplares, por lo que el algoritmo los cataloga por similitud en sus características y a partir de ahí, crea grupos o clústeres sin tener la capacidad de definir cómo es cada individualidad de cada uno de los integrantes de los mismos ([Zambrano, 2018](#)).

El algoritmo usado para este tipo de aprendizaje es el de las K medias o *k-means* en inglés. Este intenta encontrar una partición de las muestras en K agrupaciones, de manera que cada ejemplar pertenezca a una de ellas de acuerdo al centroide más cercano. Si bien el valor de K es definido por el usuario, a partir de pruebas de varias iteraciones se le puede consultar al algoritmo cuál es su valor óptimo. La función para lograr esto se observa en la Ecuación [2.3](#):

$$\sum_i \sum_j d(x_j^i, c_i)^2$$

Ecuación 2.3: Fórmula del algoritmo k-means. Fuente: [Sancho Caparrini, 2018](#)

Donde:

c_i = centroide de la agrupación i-ésima

x_j^i = conjunto de ejemplos clasificados de la agrupación i-ésima

Representándose en la Figura [2.3](#), los pasos seguidos para este algoritmo comienzan con la selección de los K puntos como centros de los grupos. Luego, se asignarán los ejemplos al centro más cercano y se calculará el centroide de los ejemplos asociados a cada grupo. Finalmente, estos dos últimos pasos se repetirán hasta que ninguno de los centros pueda ser reasignados en las iteraciones.

- **Aprendizaje por refuerzo:** Se basa en que un agente racional puede tomar una decisión a partir de una retroalimentación llamada recompensa o refuerzo. A diferencia del Aprendizaje Supervisado, en donde el agente puede aprender solamente a partir de ejemplos dados, en este caso no basta solamente con proporcionárselos sino también de “informarle” si lo está haciendo de la manera correcta o no. Por ejemplo, un agente que intenta aprender a jugar ajedrez necesita saber que algo bueno ha ocurrido cuando gana y algo malo ha ocurrido cuando pierde. La mejor recompensa que busca al finalizar el juego es vencer al oponente, y para ello debe estudiar todos los movimientos que este haga, la posición de las fichas en el tablero, entre otros. A este conjunto se le conoce como entorno

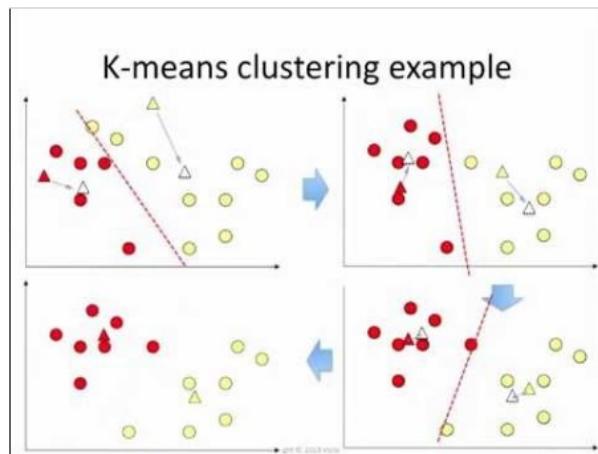


Figura 2.3: Funcionamiento del algoritmo de K medias. Fuente: [Sancho Caparrini, 2018](#)

o medio ambiente ([Russell & Norvig, 2004](#)). Entonces, en resumen, representando en la Figura 2.4, y mencionando otro ejemplo, el aprendizaje por refuerzo está compuesto por un agente (Pacman) en un estado determinado (su ubicación o posición actual) dentro de un medio ambiente (el laberinto). La recompensa positiva que busca Pacman son los puntos por comer, mientras que la negativa será la de morir si se cruza con un fantasma, en base a la acción (desplazamiento a un nuevo estado) que realice ([Merino, 2019](#)).

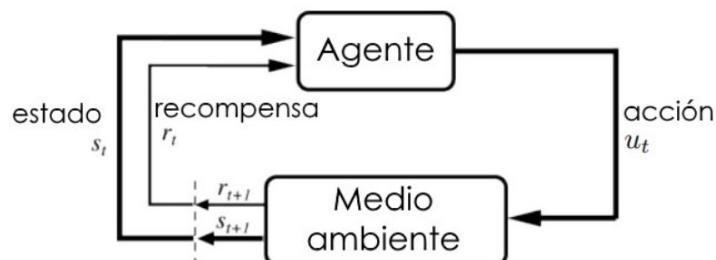


Figura 2.4: Componentes del Aprendizaje por Refuerzo. Fuente: [Sutton y Barto, 2018](#)

2.2.3. Aprendizaje Profundo

El Aprendizaje Profundo (*Deep Learning* por su nombre en inglés) es un tipo de Aprendizaje Automático que entrena a una computadora para que realice tareas como las realizadas por los seres humanos, desde la identificación de imágenes hasta realizar predicciones y reconocer el lenguaje humano. El Aprendizaje Profundo configura parámetros básicos acerca de los datos y entrena a la computadora para que aprenda por su cuenta reconociendo patrones mediante el uso de múltiples capas de procesamiento ([SAS Institute, s.f.](#)). Se basa en teorías acerca de cómo funciona el cerebro humano ([BBVA OpenMind, 2019](#)).

La principal diferencia con el Aprendizaje Automático es que el Aprendizaje Profundo se basa en la extracción de características y clasificación al mismo tiempo luego de recibir una entrada, algo que en la primera técnica ocurre por separado, como se aprecia en la Figura 2.5.

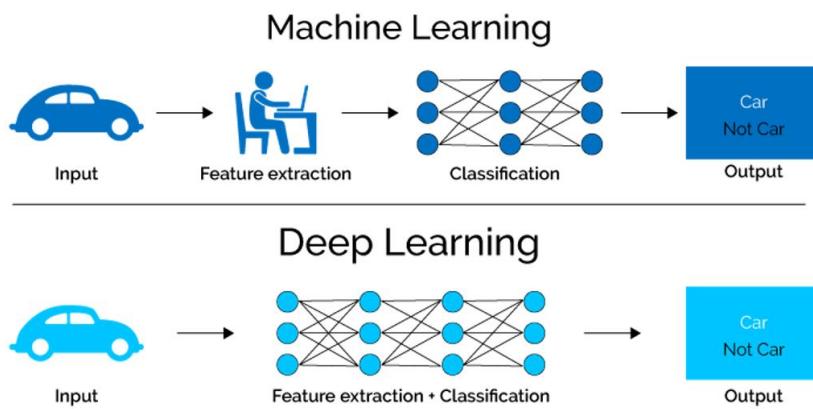


Figura 2.5: Diferencia entre Aprendizaje Automático y Aprendizaje Profundo. Fuente: [House of Bots, 2018](#)

Por un lado, mientras en el Aprendizaje Automático o de máquina, el ordenador extrae conocimiento a través de experiencia supervisada, en el aprendizaje profundo está menos sometido a supervisión. Mientras que el primer tipo de aprendizaje consume muchísimo tiempo y se basa en proponer abstracciones que permiten aprender al ordenador, en el segundo no consume demasiado tiempo y por el contrario de su par, crea redes neuronales a gran escala que permiten que el ordenador aprenda y piense por sí mismo sin necesidad directa de intervención humana. Actualmente, el Aprendizaje Profundo se usa para crear softwares capaces de determinar emociones o eventos descritos en textos, reconocimiento de objetos en fotografías y realizar predicciones acerca del posible comportamiento futuro de las personas. Empresas como Google (proyecto Google Brain) o Facebook (Unidad de investigación en IA) han puesto en marcha proyectos basados en esta rama para potenciar y mejorar sus algoritmos con el fin de ofrecer una mejor experiencia de sus servicios a sus clientes ([BBVA OpenMind, 2019](#)).

2.2.4. Modelo Predictivo

Son modelos de datos estadísticos utilizados para predecir el comportamiento futuro. En estos, se recopilan datos históricos y actuales, se formula un modelo estadístico, se realizan predicciones y el modelo se valida a medida que se dispone de datos adicionales. Los modelos predictivos analizan el rendimiento pasado para evaluar la probabilidad de que un cliente muestre un comportamiento específico en el futuro. En esta categoría también abarca la búsqueda de patrones ocultos ([Gartner, 2019b](#)).

2.2.5. Minería de Datos

La Minería de Datos es un campo de la estadística y las ciencias de la computación referido al proceso que intenta descubrir patrones en grandes volúmenes de conjuntos de datos ([Maimon & Rokach, 2010](#)). Normalmente, estos patrones no pueden detectarse mediante la exploración tradicional de datos porque sus relaciones son demasiado complejas o por su gran volumen. Para ello, utiliza métodos de Inteligencia Artificial, Aprendizaje Automático, estadística y sistemas de bases de datos. Estos patrones son recopilados y definidos como un modelo de minería de datos, los cuales pueden aplicarse en los siguientes escenarios ([Microsoft, 2019](#)):

- Previsión.
- Riesgo y probabilidad.
- Recomendaciones.
- Buscar secuencias.
- Agrupación.

La generación de un modelo de minería de datos forma de un macro-proceso descrita en los siguientes seis pasos representados en la Figura 2.6 ([Microsoft, 2019](#)):

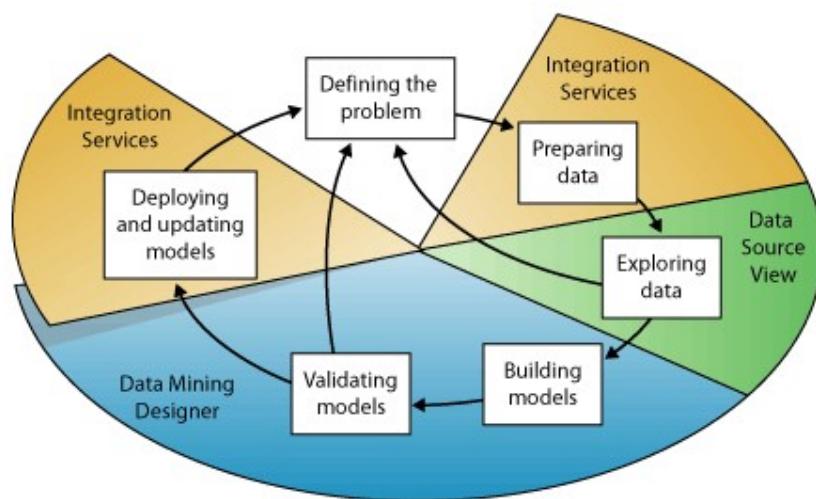


Figura 2.6: Diagrama de los seis pasos básicos. Fuente: [Microsoft, 2019](#)

2.2.6. Metodologías de Minería de Datos

Dentro de los sistemas de analítica de negocio, Big Data y Minería de Datos, las tres metodologías más usadas se encuentran CRISP-DM, SEMMA y KDD ([Braulio Gil & Curto Díaz, 2015](#)).

- **CRISP-DM** (Cross Industry Standard Process for Data Mining):

Esta metodología presenta seis fases representadas en la Figura 2.7 a continuación.

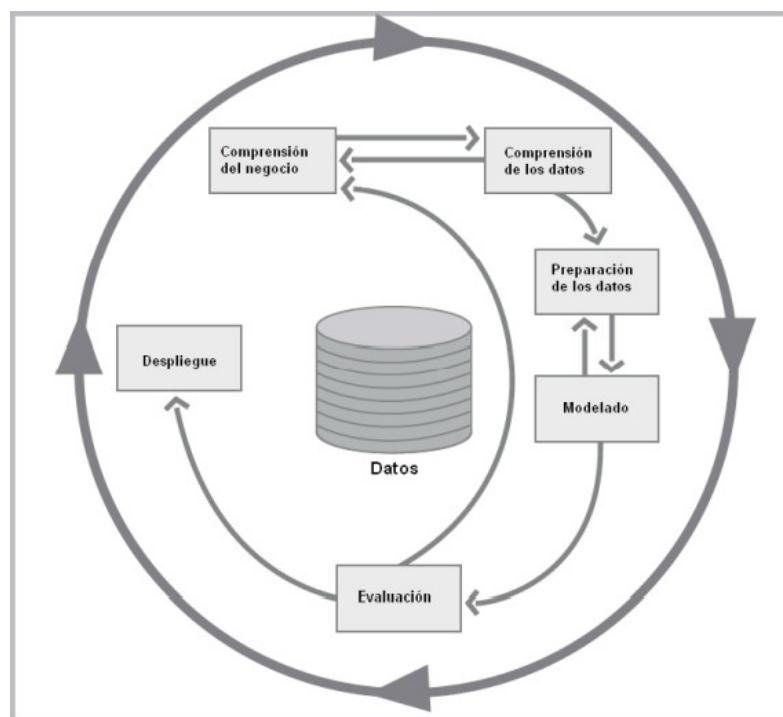


Figura 2.7: Fases de la metodología CRISP-DM. Fuente: [Braulio Gil y Curto Díaz, 2015](#)

- En la comprensión del negocio se determinan los objetivos y requerimientos desde el lado del negocio, así como generar plan del proyecto.
- En la comprensión de los datos se logra entender el significado de las variables existentes, así como el entendimiento de los datos desde su recopilación hasta su verificación de calidad.
- En la preparación de los datos se prepara el conjunto de datos adecuado que servirán para la construcción del modelo. Por ello, la calidad de los datos es un factor relevante y ello requiere la exclusión de redundancia y valores que no ayuden a establecer buena comprensión y resultados más adelante. A esto se le conoce como limpieza de datos.

- En el modelado se aplican técnicas de minería de datos en el conjunto de datos creado en el paso anterior. Para ello, se evalúan entre varias la que mejor performance desempeñe y luego se construye el o los modelos que busquen determinar un objetivo.
 - En la evaluación se evalúan los posibles modelos del paso anterior a partir del nivel de importancia de acuerdo a las necesidades del negocio y performance que estos cuentan.
 - El despliegue, finalmente, utiliza el modelo final creado para determinar los objetivos que se buscan cumplir en los requerimientos y ayudar en la toma de decisiones.
- **SEMMA** (Sample – Explore – Modify – Model – Assess):

Esta metodología cuenta con cinco fases como se aprecia en la Figura 2.8. A diferencia de la anterior, esta metodología se enfoca más en el modelado.

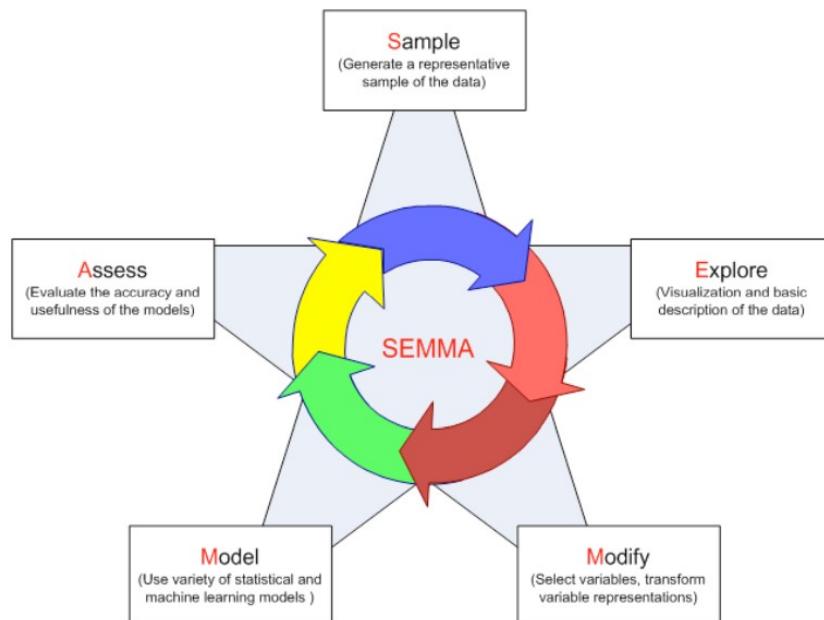


Figura 2.8: Fases de la metodología SEMMA. Fuente: [Braulio Gil y Curto Díaz, 2015](#)

- En la Muestra (*Sample*) se crea una muestra significativa.
- En la Exploración (*Explore*) se comprenden los datos con el fin de encontrar relaciones entre variables y anomalías.
- En la Modificación (*Modify*) se transforman las variables para las necesidades del modelo.
- En la Modelización (*Model*) se aplican uno o varios modelos sobre el conjunto de datos para buscar resultados.

- En el Asesoramiento (*Assessment*) se evalúan los resultados obtenidos del modelo.
- **KDD** (Knowledge Discovery and Data Mining):

Esta metodología se refiere al proceso de encontrar conocimiento alguno en el dato y, a diferencia de sus predecesores, se enfoca en crear aplicaciones de minería de datos. Consiste de cinco fases más 1 previa y 1 posterior basadas en la generación de conocimiento como se muestra en la Figura 2.9.

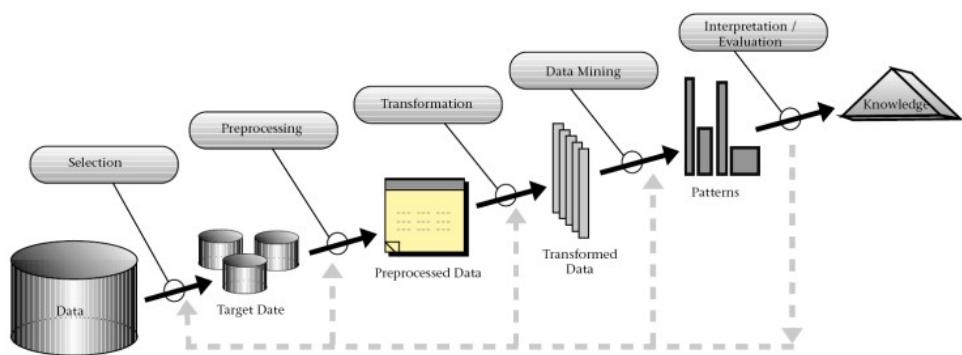


Figura 2.9: Fases de la metodología KDD. Fuente: [Braulio Gil y Curto Díaz, 2015](#)

- En la fase Pre KDD se comprende el dominio del negocio, así como también se identifican las necesidades del cliente.
- En la selección, primero se identifica el conjunto de datos a usar y luego se seleccionan la muestra y las variables para la exploración.
- En el pre-procesamiento, se realiza la limpieza de datos y se elimina el ruido, así como los valores atípicos.
- En la transformación se implementan métodos de reducción de dimensiones para reducir el número de variables efectivas.
- En la Minería de datos, se elige el tipo de tarea de minería de datos (clasificación, regresión, agrupamiento, entre otros) así como el algoritmo, los métodos, los modelos y parámetros apropiados.
- En la interpretación y evaluación se analizan los resultados dados.
- En la fase Post KDD finalmente se consolida el conocimiento adquirido.

Luego de presentar las tres metodologías más usadas, la pregunta dada es ¿cuál de los tres representa la mejor opción para usar? Las tres metodologías tienen distinto número de pasos, así como distintos enfoques, tal cual se observa en el siguiente resumen de la Tabla 2.1.

Modelo de Procesos de Minería de Datos	KDD	CRISP-DM	SEMMA
Número de pasos	9	6	5
Nombre de los pasos	Desarrollo y entendimiento de la aplicación	Entendimiento del negocio	-
	Creación de un conjunto de datos de destino	Entendimiento de los datos	Muestreo
	Limpieza de datos y pre-procesamiento		Exploración
	Transformación de datos	Preparación de los datos	Modificación
	Elección de la tarea adecuada de Minería de datos	Modelamiento	Modelo
	Elección del algoritmo adecuado de Minería de datos		
	Implementación del algoritmo de Minería de datos		
	Interpretación de patrones minados	Evaluación	Evaluación
	Uso de conocimiento descubierto	Despliegue	-

Tabla 2.1: Cuadro comparativo entre características de las tres metodologías. Fuente: [Shafique y Qaiser, 2014](#)

Sin embargo, la elección depende de los involucrados que finalmente usarán el modelo en el negocio. La mayoría de investigadores siguen la metodología KDD debido a que es más completo y su exactitud. Para aquellos objetivos enfocados más en la compañía como la integración usada por SAS Enterprise Miner con su software se utilizan SEMMA y CRISP-DM. Esta última resulta ser más completa de acuerdo a los estudios.

2.2.7. Técnicas de Minería de Datos

Existe una gran variedad de técnicas para la Minería de Datos. Las más importantes y utilizadas en los antecedentes de la investigación se mencionan a continuación ([Microsoft, 2018](#)).

- **Redes Neuronales Artificiales (RNA):** Es un sistema de computación que consiste en un número de elementos o nodos simples, pero altamente interconectados, llamados “neuronas”, que se organizan en capas que procesan información utilizando respuestas de estado dinámico a entradas externas ([Inzaugarat, 2018](#)).

Este sistema de programas y estructura de datos se aproxima al funcionamiento del cerebro humano. Una red neuronal implica tener un gran número de procesadores funcionando en paralelo, teniendo cada uno de ellos su propia esfera de conocimiento y acceso a datos en su memoria local. Normalmente, una se alimenta con grandes cantidades de datos y un conjunto dado de reglas acerca de las relaciones. Luego, un programa puede indicar a la red cómo debe comportarse en respuesta a un estímulo externo o si puede iniciar la actividad por sí misma ([BBVA OpenMind, 2019](#)).

Para entender mejor cómo funciona una red neuronal, hay que describir qué es una neurona. Una neurona es una célula del cerebro cuya función principal es la recogida, procesamiento y emisión de señales eléctricas. Debido a que se piensa que la capacidad de procesamiento de información del cerebro proviene de redes de este tipo de neuronas, los primeros trabajos en Inteligencia Artificial se basaron en crear redes neuronales artificiales para emular este comportamiento, en 1943 con un modelo matemático, mostrado en la Figura 2.10, por los ya mencionados anteriormente McCulloch y Pitts.

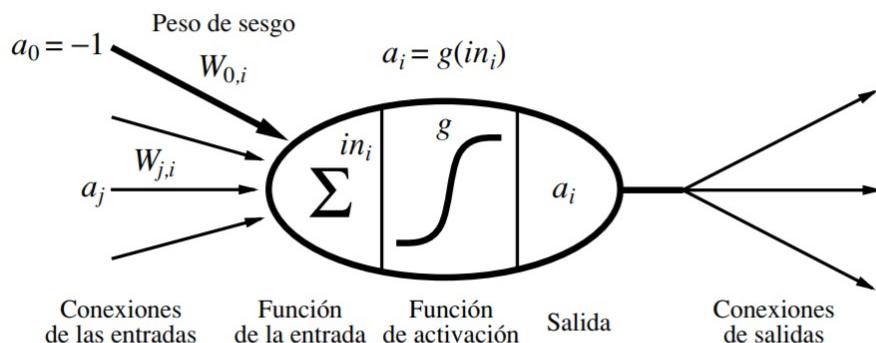


Figura 2.10: Modelo para representar una neurona propuesto por McCulloch y Pitts (1943). Fuente: [Russell y Norvig, 2004](#)

Estos y posteriores trabajos potenciaron lo que hoy en día se conoce como el campo de

la neurociencia computacional ([Russell & Norvig, 2004](#)). Años más tarde, en 1958, se desarrolló el concepto del perceptrón por Rosenblatt, el cual tenía la capacidad de aprender y reconocer patrones sencillos, formado por entradas, neurona, función de adaptación (sigmoidal, tangencial, en escalón, etc.) y salida.

La última figura descrita muestra, además de los pesos, funciones de activación tanto para la entrada (a_j) como para la salida (a_i). Pero, ¿qué son estas funciones y para qué sirven?

Para comenzar, las redes neuronales están compuestas de nodos (la elipse) conectados a través de conexiones dirigidas (las flechas). Una conexión del nodo j a la unidad i sirve para propagar la activación a_j de j a i . Asimismo, cada conexión tiene un peso numérico $W(j, i)$ que determina la fuerza y el signo de la conexión. Para calcular cada nodo i , se realiza una suma ponderada de sus entradas (producto entre pesos y nodos de entrada j), y se le añade el sesgo (*bias*) θ_i (aumenta/disminuye el valor de la combinación lineal de las entradas) como se observa en la Ecuación [2.4](#):

$$in_i = \sum_{j=0}^n W_{j,i} * a_j + \theta_i$$

Ecuación 2.4: Fórmula del cálculo del valor de un nodo i . Fuente: [Russell y Norvig, 2004](#)

Posteriormente, se efectúa una función de activación g a esta suma para producir la salida de la Ecuación [2.5](#):

$$a_i = g(in_i) = g \left(\sum_{j=0}^n W_{j,i} * a_j + \theta_i \right)$$

Ecuación 2.5: Fórmula de una función de activación g para la salida del nodo. Fuente: [Russell y Norvig, 2004](#)

Entonces, aquí se explica los dos objetivos de una función de activación. En primer lugar, se desea que el nodo esté “activo” (cercano a +1) cuando las entradas correctas sean dadas, e “inactiva” (cercano a 0) cuando las entradas erróneas sean proporcionadas. En segundo lugar, la activación tiene que ser no lineal porque, de lo contrario, la red neuronal colapsaría en su totalidad con una función lineal sencilla (Figura 2.11).

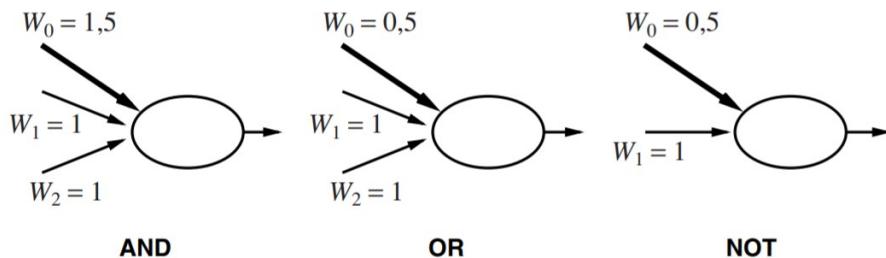


Figura 2.11: Nodos con funciones de activación umbral en forma de puertas lógicas. Fuente: [Russell y Norvig, 2004](#)

Entre las funciones de activación que más destacan son las siguientes:

- **Función sigmoide o logística:** Toma los valores de entrada que oscilan entre infinito negativo y positivo, y restringe los valores de salida al rango entre 0 y 1. Frecuentemente es usada en Redes Multicapa (MLP) entrenadas con el algoritmo de propagación inversa. Se representa como en la Figura 2.12 y su fórmula para calcular su nuevo valor es la Ecuación 2.6:

$$a = \text{Logsig}(n) = \frac{1}{1 + e^{-n}}$$

Ecuación 2.6: Fórmula de la función de activación sigmoide. Fuente: [Dorofki y col., 2012](#)

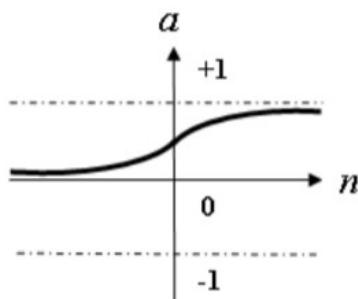


Figura 2.12: Función de activación sigmoide. Fuente: [Dorofki y col., 2012](#)

Un dato curioso de esta función relacionado con la regresión logística es que el nombre de esta última no deriva de una regresión. Por el contrario, se debe a que,

al principio de la neurona, se realiza una combinación lineal muy parecida a una regresión lineal y después se aplica la función logística o sigmoide. De ahí el origen del nombre ([IArtificial.net, 2019a](#)).

- **Regresión Logística:** Como se mencionó antes, es similar a un modelo de regresión lineal, pero está adaptado para modelos en los que la variable dependiente es dicotómica, es decir, presenta solo dos posibles valores. Resulta muy útil para los casos en los que se desea predecir la presencia o ausencia de una característica o resultado según los valores de un conjunto de predictores ([IBM, 2019](#)). Su función de coste que se optimiza con gradiente descendiente se representa mediante la Ecuación [2.7](#):

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m [-y_i * \log(h_\theta(x_i)) - (1 - y_i) * \log(1 - h_\theta(x_i))]$$

Ecuación 2.7: Fórmula de función de coste de una regresión logística. Fuente: [Pardo, s.f.](#)

Donde:

$$\begin{aligned} h_\theta(x_i) &= \text{función sigmoide de } \theta^T x \\ \theta &= \text{vector de longitud theta para } j=0,1,2,3\dots n \\ x &= \text{matriz de entradas} \\ y &= \text{vector de salidas} \end{aligned}$$

La primera parte de la ecuación está conformada por el logaritmo de la probabilidad de éxito y la segunda, por la de fracaso.

- **Gradiente descendiente:** Es un método de optimización numérica para estimar los mejores coeficientes, fundamental en Deep Learning para entrenar redes neuronales y en muchos casos, para la regresión logística. A través de una función E(W), proporciona el error que comete la red en función del conjunto de pesos sinápticos W. El objetivo del aprendizaje será encontrar la configuración de pesos que corresponda al mínimo global de la función de error o coste ([Bertona, 2005](#)).

En general, la función de error es una función no lineal, por lo que el algoritmo realiza una búsqueda a través del espacio de parámetros que, se aproxime de forma iterada a un error mínimo de la red para los parámetros adecuados, como se aprecia en la Figura [2.13](#) ([Sancho Caparrini, 2017](#)).

El Descenso del Gradiente, como también se le conoce, es el algoritmo de entrenamiento más simple y también el más extendido y conocido. Solo hace uso del vector gradiente, y por ello se dice que es un método de primer orden

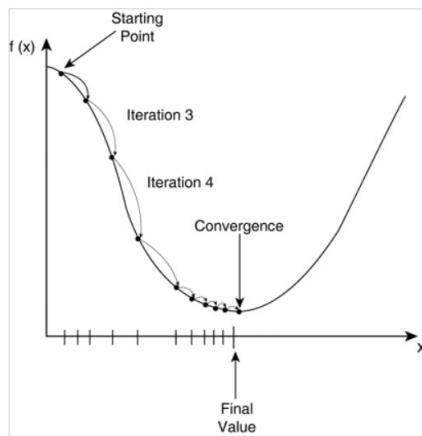


Figura 2.13: Ilustración del algoritmo gradiente descendente. Fuente: [Sancho Caparrini, 2017](#)

([Sancho Caparrini, 2017](#)). Un gradiente es la generalización de la derivada. Matemática, la derivada de una función mide la rapidez con la que cambia el valor de esta, según varíe el valor de su variable independiente. La gradiente se calcula con derivadas parciales, por lo que al actualizar los coeficientes W para un tiempo t, se usa la Ecuación 2.8 ([IArtificial.net, 2019b](#)):

$$W_{(t+1)} = W_{(t)} - \alpha \left(\frac{\partial MSE}{\partial W} \right)$$

Ecuación 2.8: Actualización de pesos W mediante gradiente descendente. Fuente: [IArtificial.net, 2019b](#)

Donde: α = ratio de aprendizaje

Este ratio controla el tamaño de la actualización. Si este es demasiado grande, será más difícil encontrar los coeficientes que minimicen la función de coste o error; la actualización de W es proporcional al gradiente; y se usa la resta para ir en dirección opuesta al gradiente como en la Figura 2.14.

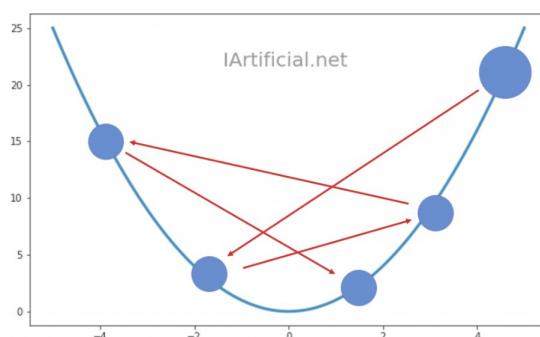


Figura 2.14: Actualización de pesos W con el algoritmo. Fuente: [IArtificial.net, 2019b](#)

- **Propagación hacia atrás:** También conocido en inglés como *Backpropagation*, es un método que consta de dos fases: en la primera se aplica un patrón, el cual se propaga por las distintas capas que componen la red hasta producir la salida de la misma. Luego, esta se compara con la salida deseada y se calcula el error cometido por cada neurona de salida. Estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de las capas intermedias [Fritsch, 1996] ([Bertona, 2005](#)). La actualización iterativa de los pesos que el algoritmo propone es mediante la Ecuación 2.9:

$$W_{ji}(t+1) = W_{ji}(t) + [\alpha \delta_{pj} y_{pj} + \beta \Delta W_{ji}(t)]$$

Ecuación 2.9: Fórmula del algoritmo de propagación hacia atrás. Fuente: [Bertona, 2005](#)

$$\text{siendo } \delta_{pj} = \begin{cases} (d_{pj} - y_{pj})f'_j(h_j) & \text{si j es una neurona de salida} \\ \left(\sum_k \delta_{pk} W_{kj} \right) f'_j(h_j) & \text{si j es una neurona oculta} \end{cases}$$

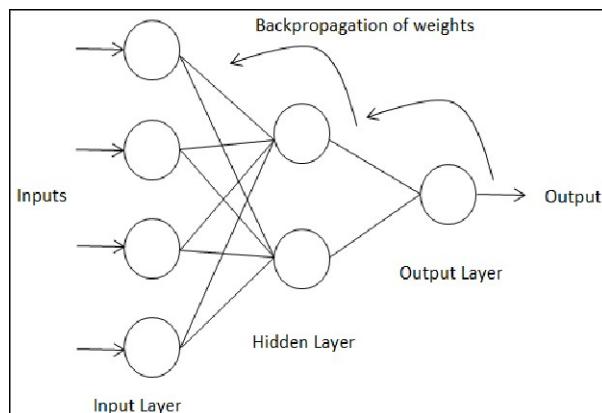


Figura 2.15: Capa oculta simple MLP con propagación hacia atrás. Fuente: [IArtificial.net, 2019b](#)

Para entender mejor la teoría y la fórmula de actualización de pesos, se seguirá el siguiente ejemplo del conjunto de redes de la Figura 2.16.

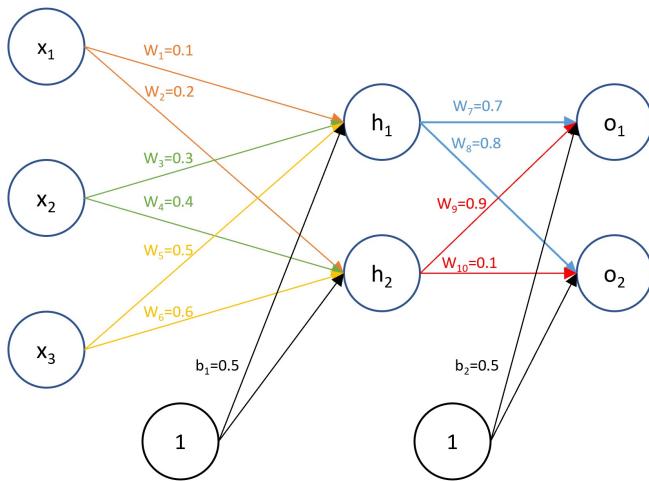


Figura 2.16: Redes neuronales de ejemplo. Fuente: [A Not So Random Walk, 2019](#)

Se tiene una red neuronal con tres nodos de entradas ($x_1=1$, $x_2=4$ y $x_3=5$) con dos pesos respectivos cada una ($W_1=0.1$ y $W_2=0.2$ para x_1 ; $W_3=0.3$ y $W_4=0.4$ para x_2 ; $W_5=0.5$ y $W_6=0.6$ para x_3), dos capas ocultas (h_1 y h_2) con dos peso cada una ($W_7=0.7$ y $W_8=0.8$ para h_1 ; $W_9=0.9$ y $W_{10}=0.1$ para h_2) y dos nodos de salida (O_1 y O_2).

El proceso normal para calcular el valor del nodo final se da, tanto con los nodos de entrada y los de capa oculta, mediante la sumatoria de producto de cada peso con su valor, es decir, mediante la fórmula de las RNA $in_i = \sum_{j=0}^n W_{j,i} * a_j + b_{j,i}$, al mismo tiempo que devuelve un valor del error cometido. Este último se calcula mediante la Ecuación 2.10:

$$E_k = (T_k - O_k) * O_k * (1 - O_k)$$

Ecuación 2.10: Cálculo del error cometido en una red neuronal. Fuente: [Viera Balanta, 2013](#)

Donde:

T_k = salida correcta de cada nodo de salida

O_k = salida actual que cada nodo genera

Con estos errores, se retrocede hacia la capa oculta y se procede a calcular los nuevos pesos para sus nodos. Esto se realiza mediante la Ecuación 2.11:

$$W_{jk} = W_{jk} + L * E_k * O_j$$

Ecuación 2.11: Actualización de pesos mediante propagación hacia atrás. Fuente: [Viera Balanta, 2013](#)

Donde:

W_{jk} = peso a actualizar para cada nodo de la capa oculta (W_7, W_8, W_9 y W_{10})

L = porcentaje de aprendizaje

O_j = valor de los nodos que entrarán a las salidas

Estos nuevos pesos permitirán redefinir los errores de ambos nodos, con una pequeña diferencia en su cálculo (Ecuación 2.12):

$$E_j = O_j * (1 - O_j) * \sum E_k * W_{jk}$$

Ecuación 2.12: Cálculo de errores de nodos usando pesos actualizados. Fuente: [Viera Balanta, 2013](#)

El error de cada nodo de la capa oculta se obtiene multiplicando su valor por su complemento por la sumatoria del producto de sus pesos y los errores de los nodos de salida. Por ejemplo, para h_1 sería $E_{h1} = h_1 * (1 - h_1) * (0.7 * O_1 + 0.8 * O_2)$.

Finalmente, se retrocede hacia los nodos de entrada y se repite el mismo proceso para la actualización de sus pesos y errores.

- **Función tangente hiperbólica:** Esta función está relacionada con una sigmoide bipolar. Sin embargo, sus salidas estarán en el rango de -1 y +1. Para redes neuronales, donde la velocidad es más importante que la forma de la función misma, es recomendable usar esta. Se representa como en la Figura 2.17 y su fórmula para calcular su nuevo valor es la Ecuación 2.13:

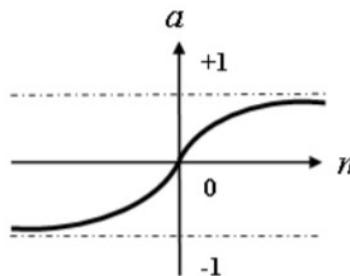


Figura 2.17: Función de activación tangente hiperbólica. Fuente: [Dorofki y col., 2012](#)

$$a = Tansig(n) = \frac{2}{1 + e^{-2n}} - 1$$

Ecuación 2.13: Fórmula de la función de activación tangente hiperbólica. Fuente: [Dorofki y col., 2012](#)

- **Función puramente lineal (purelin):** Se caracteriza porque su salida es igual a su entrada debido a su linealidad. Normalmente se usa para obtener los mismos valores de entrada. Se representa en la Figura 2.18 y se calcula mediante la Ecuación 2.14:

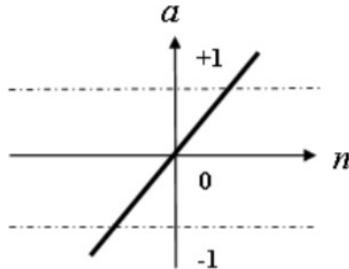


Figura 2.18: Función de activación puramente lineal. Fuente: [Dorofki y col., 2012](#)

$$a = n$$

Ecuación 2.14: Fórmula de la función de activación puramente lineal. Fuente: [Dorofki y col., 2012](#)

- **Función Unidad Lineal Rectificada (ReLU):** Se caracteriza por conservar los valores positivos y convertir los negativos de entrada en 0, con la finalidad de no considerarlos en la siguiente capa de convolución ([SitioBigData.com, 2019b](#)). Si bien tiene un buen desempeño en redes convolucionales y es muy usada para procesar imágenes, al no estar acotada pueden morirse demasiadas neuronas ([Calvo, 2018b](#)). Se representa en la Figura 2.19 y se calcula mediante la Ecuación 2.15:

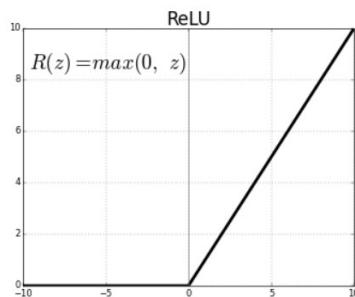


Figura 2.19: Función de activación ReLU. Fuente: [Machine Learning for Artists, 2019](#)

$$f(x) = \max(0, x) = \begin{cases} 0 & \text{para } x < 0 \\ x & \text{para } x \geq 0 \end{cases}$$

Ecuación 2.15: Fórmula de la función de activación ReLU. Fuente: [Calvo, 2018b](#)

Además de existir distintas funciones de activación, las redes neuronales artificiales se clasifican según la topología de red, siendo algunas de las más importantes ([Calvo, 2017](#)).

- **Red Neuronal Monocapa – Perceptrón simple:** Es la red neuronal más simple ya que está compuesta solamente de una capa de neuronas que componen varios nodos de entrada para proyectar una capa de neuronas de salida, como se aprecia en la Figura 2.20. Esta última capa se calcula usando la misma Ecuación 4 que implica la suma de productos de cada uno de los pesos de los nodos de entrada con sus instancias, añadiéndole finalmente el sesgo, aquel que controla la predisposición de la neurona a disparar un 1 o 0 independientemente de los pesos, para que el valor resultante se le aplique la función de activación que ayudarán a modelar funciones curvas o no triviales ([Machine Learning for Artists, 2019](#)).

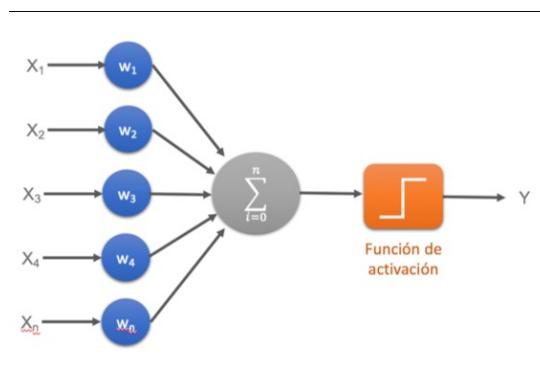


Figura 2.20: Ejemplo de perceptrón simple. Fuente: [Calvo, 2017](#)

- **Red Neuronal Multicapa – Perceptrón multicapa:** Con arquitectura similar al perceptrón simple, con el añadido de contener capas intermedias entre la capa de neuronas de entrada y la de salida, conocidas como capas ocultas, como en el ejemplo de la Figura 2.21.

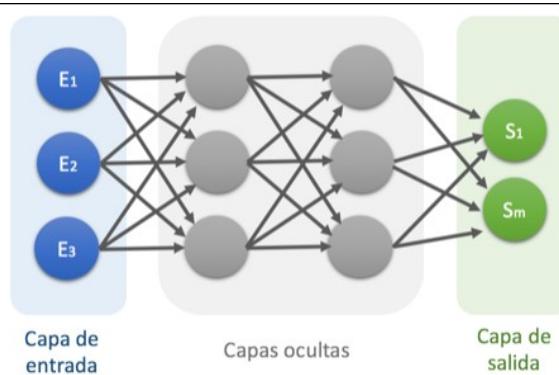


Figura 2.21: Ejemplo de perceptrón multicapa. Fuente: [Calvo, 2017](#)

- **Redes Neuronales Convolucionales (CNN):** También conocidas por su nombre en inglés *Convolutional Neural Networks*, se diferencia del perceptrón multicapa en que cada neurona no necesita estar unida con todas las que le siguen, sino más bien solo con un subgrupo de estas con el fin de reducir la cantidad de neuronas necesarias para su funcionamiento, como se observa en la Figura 2.22 ([Calvo, 2017](#)).

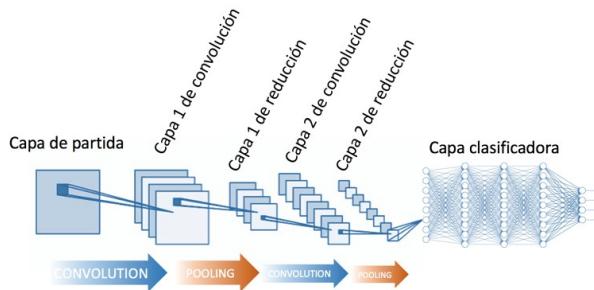


Figura 2.22: Ejemplo de red neuronal convolucional. Fuente: [Calvo, 2017](#)

Hoy en día, las redes neuronales convolucionales tienen múltiples usos desde que la idea fue concebida. Algunos de los problemas en las que pueden ser usados son de clasificación de objetos, recuperación de imágenes, detección y segmentación de objetos, distorsión y filtros de imágenes, por citar los ejemplos más comunes. El modelo de CNN más conocido es “AlexNet” (2012) por ser uno de los pioneros en clasificar imágenes ([F.-F. Li y col., 2019](#)).

Estas redes tienen su origen en el Neocognitron introducido por Fukushima en 1980 como modelo de red neuronal para el mecanismo de reconocimiento de patrón visual sin la enseñanza de un “profesor” (ver Figura 2.23), mismo que en el año 1998 sería mejorado por LeCun, Bottou, Bengio y Haffner al agregar un método de aprendizaje de gradiente aplicado al reconocimiento de documento basado en la propagación hacia atrás (ver Figura 2.24) ([F.-F. Li y col., 2019](#)).

Estos modelos se inspiraron en el estudio de la información visual en la corteza donde se ubican hasta 5 áreas. La primera, V1, contiene la información visual donde sus neuronas se ocupan de características visuales de bajo nivel, alimentando así a otras áreas adyacentes. Cada una de ellas se encarga de aspectos más específicos y detallados de la información obtenida. La idea de su implementación es la de solucionar el problema que surgen al escalar imágenes de mucha definición por las redes neuronales ordinarias. Por ello, este tipo de redes trabajan modelando de forma consecutiva piezas pequeñas de información para luego combinarlas en sus capas más profundas ([López Briega, 2016](#)).

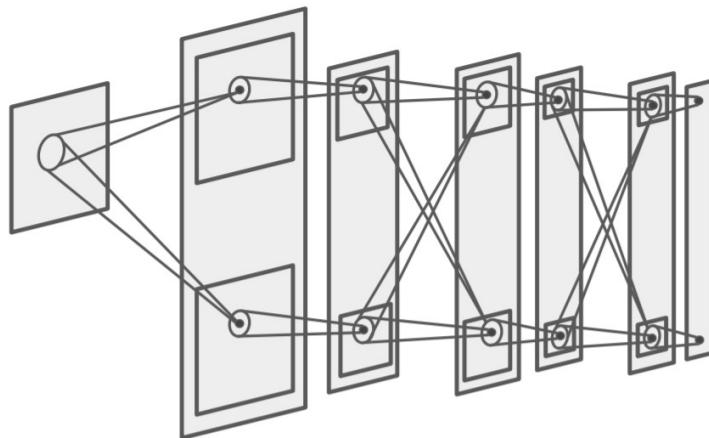


Figura 2.23: Modelo Neocognitron de Fukushima (1980). Fuente: [F.-F. Li y col., 2019](#)

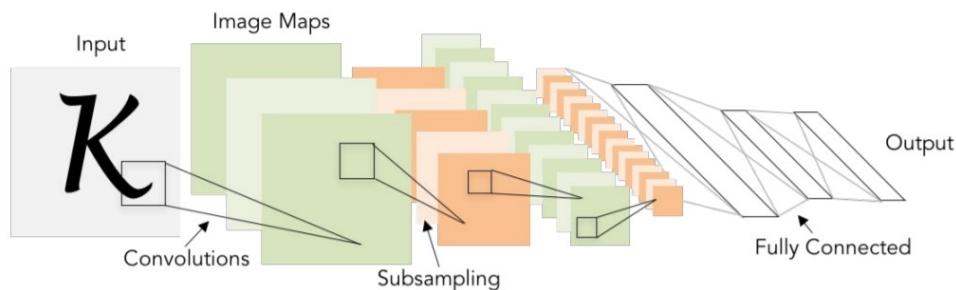


Figura 2.24: Modelo LeNet-5 de LeCun (1998). Fuente: [F.-F. Li y col., 2019](#)

Su nombre deriva del concepto convolución. La convolución es un término en las matemáticas usado como operador matemático que convierte dos funciones f y g en una tercera función en donde la primera se superpone a una versión invertida y trasladada de la segunda, así como para denotar la distribución de la función de probabilidad de la suma de dos variables independientes aleatorias. Esta se da por la Ecuación 2.16 ([Figueroa M., s.f.](#)):

$$(f * g)(t) = \int_0^t f(t - \tau)g(\tau)d\tau$$

Ecuación 2.16: Fórmula matemática de la convolución. Fuente: [Figueroa M., s.f.](#)

Donde el rango puede variar entre un conjunto finito (como en la fórmula desde 0 hasta un valor t) o uno infinito.

La estructura de las Redes Neuronales Convolucionales se constituye en tres tipos de capas ([López Briega, 2016](#)).

- **Capa convolucional (*Convolutional Layer*):** Es la capa que hace distinta a esta red de otros tipos de redes neuronales artificiales. Se aplica la operación de la convolución, que recibe como entrada (*input* en inglés) a la imagen para luego

aplicarle un filtro (*kernel* en inglés), devolviendo un mapa de las características de la imagen original, logrando así reducir el tamaño de los parámetros, como se observa en la Figura 2.25.

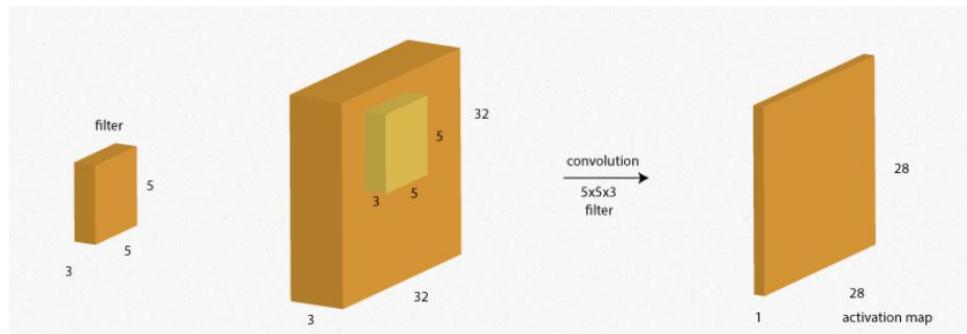


Figura 2.25: Ejecución de la convolución en una entrada. Fuente: [López Briega, 2016](#)

Por ejemplo, en la anterior figura se tiene una imagen de entrada con dimensiones de 32 de alto, 32 de ancho y 3 de profundidad (32x32x3). A ella se le aplica un filtro de dimensiones (5x5x3) que recorrerá toda la imagen para extraer características de cada pixel. Tanto la profundidad de la entrada como del filtro siempre son iguales. El resultado de tomar un producto escalar entre el filtro y un pequeño fragmento de 5x5x3 de la imagen es un número, generando así un mapa de activación de nuevas dimensiones (28x28x1). Por cada n filtros aplicados a la entrada se generan n de estos mapas. Al final, la cantidad de mapas de activación determinará una nueva imagen de n de profundidad, como en la Figura 2.26.

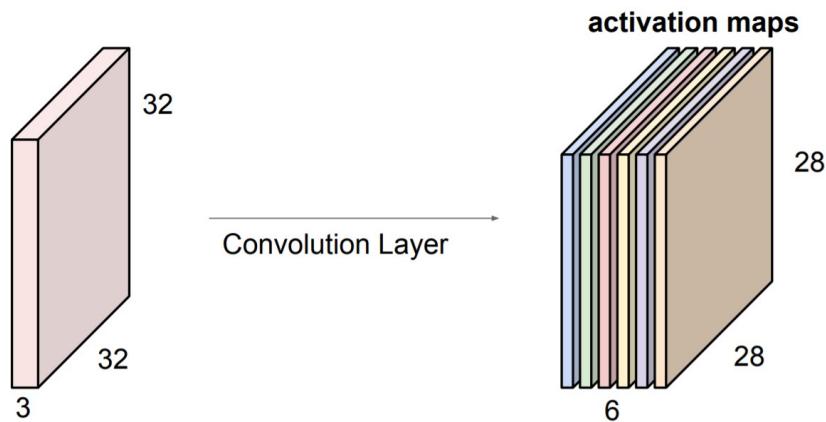


Figura 2.26: Generación de una nueva imagen a partir de filtros. Fuente: [F.-F. Li y col., 2019](#)

Asimismo, cada vez que se aplica una convolución a una imagen, se aplicará una función de activación como en la secuencia de la Figura 2.27.

A nivel visual, en la Figura 2.28 se aprecia un ejemplo de los resultados de aplicar varias convoluciones a una imagen.

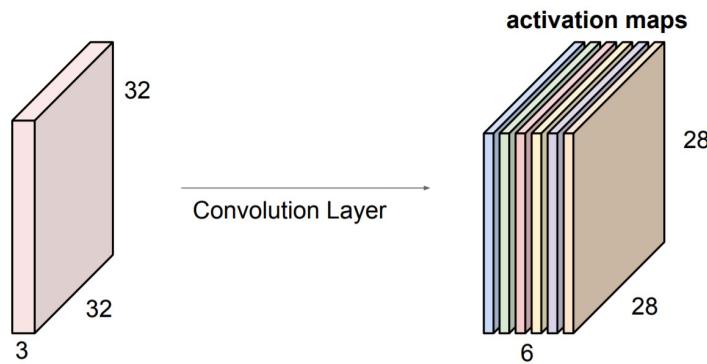


Figura 2.27: Secuencia de varias capas convolucionales. Fuente: [F.-F. Li y col., 2019](#)

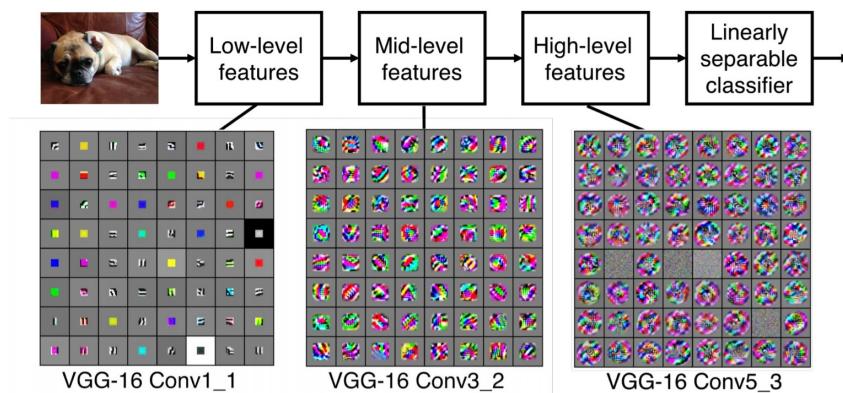


Figura 2.28: Extracción de características a partir de convoluciones. Fuente: [F.-F. Li y col., 2019](#)

Finalmente, se calcula el volumen de la dimensión de la salida de la Figura 2.29 mediante la Ecuación 2.17:

- ◊ Se tiene una entrada de dimensiones $(h * w * d)$.
- ◊ Se tiene un filtro de dimensiones $(f_h * f_w * d)$.

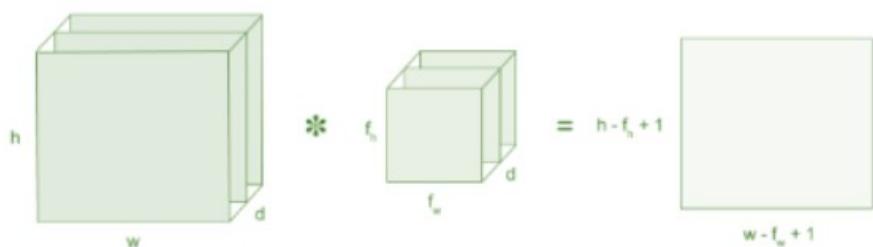


Figura 2.29: Ejemplo de matriz de imagen de entrada y un filtro. Fuente: [Prabhu, 2018](#)

$$\text{Volumen} = (h - f_h + 1) * (w - f_w + 1) * 1$$

Ecuación 2.17: Cálculo del volumen del mapa de activación. Fuente: [Prabhu, 2018](#)

- **Capa de reducción (Pooling Layer):** Esta capa le sucede a la capa convolucional (luego de aplicar la función de activación). Sirve principalmente para reducir las dimensiones espaciales del volumen de la entrada (alto x ancho) para la siguiente capa convolucional. Sin embargo, no afecta la profundidad de la misma. Esta operación que realiza se le conoce también como “reducción de muestreo” debido a que, si bien logra reducir las dimensiones para procesar mejor en la siguiente capa, también conlleva perder información. Por el contrario de lo que se piensa, además de reducir la sobrecarga del cálculo para las siguientes capas, el modelo se beneficia también disminuyendo el sobreajuste. Para determinar las dimensiones de la nueva imagen generada (siempre que sea cuadrada como en la Figura 2.30) con esta capa, se aplica la Ecuación 2.18:

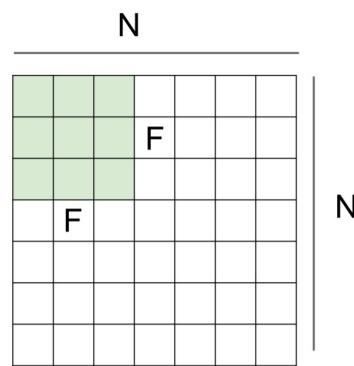


Figura 2.30: Dimensiones de una entrada y un filtro. Fuente: [F.-F. Li y col., 2019](#)

$$\text{Salida} = \frac{N - F}{S} + 1$$

Ecuación 2.18: Cálculo del tamaño de la imagen reducida. Fuente: [F.-F. Li y col., 2019](#)

Donde:

N = tamaño del lado de la imagen de entrada

F = tamaño del lado del filtro

S = número de desplazamiento de pixeles sobre la matriz de entrada

Por ejemplo, cuando el paso es 1, los filtros se mueven a 1 pixel por vez, cuando el paso es 2, se mueven a 2 píxeles (como en la Figura 2.31) y así sucesivamente ([Prabhu, 2018](#)).

Si, por el contrario, se desea aplicar convolución a una imagen sin afectar sus dimensiones luego de pasar por la capa de reducción, se construye bordes de ceros de n píxeles. A este tamaño de borde se le llama Relleno (*pad* en inglés), por lo que el tamaño de la nueva salida se obtiene mediante la fórmula:

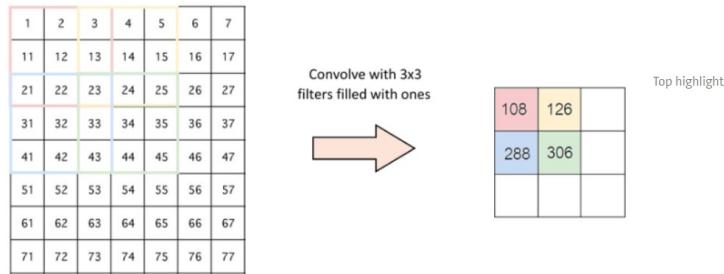


Figura 2.31: Paso de 2 píxeles por parte de un filtro. Fuente: [Prabhu, 2018](#)

$$\text{Salida} = \frac{N - F + 2 * \text{Relleno}}{S} + 1$$

Ecuación 2.19: Cálculo del tamaño de la imagen reducida con bordes rellenos con ceros. Fuente: [F.-F. Li y col., 2019](#)

Existen diferentes tipos de reducción ([Prabhu, 2018](#)):

- ◊ Max Pooling: Toma el elemento más grande dentro del mapa de características.
- ◊ Average Pooling: Toma el promedio de los elementos dentro del mapa de características.
- ◊ Sum Pooling: Toma la suma total de los elementos dentro del mapa de características.
- **Capa totalmente conectada (Fully Connected Layer):** Al final de las capas de convolución y reducción, se usan redes completamente conectadas a cada pixel considerando que cada uno como una neurona separada al igual que en una red neuronal regular ([López Briega, 2016](#)). En esta capa, se aplana la matriz de todas las características obtenidas anteriormente a un vector y se alinea en una capa completamente conectada a una red neuronal (Figura 2.32).

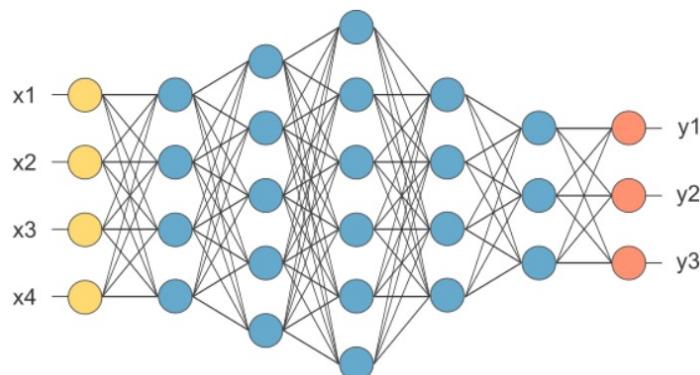


Figura 2.32: Aplanado de matrices luego de agrupar la capa. Fuente: [Prabhu, 2018](#)

Para concluir, se muestra a continuación (Figura 2.33) la representación de la arquitectura completa de una Red Neuronal Convolutiva resumiendo los conceptos anteriores.

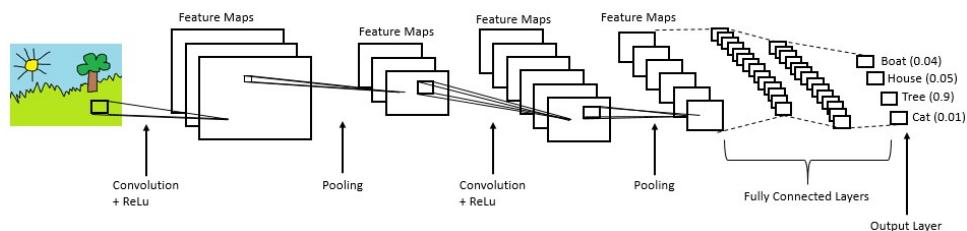


Figura 2.33: Arquitectura completa de una CNN. Fuente: [Prabhu, 2018](#)

- **Redes Neuronales Recurrentes (RNN):** También conocidas por su nombre en inglés *Recurrent Neural Networks*, se caracterizan por no tener una estructura de capas como se aprecia en la Figura 2.34, sino más bien por permitir conexiones entre sus neuronas de manera arbitraria para crear temporalidad y que toda la red obtenga memoria. Todo esto permite generar una red muy potente para el análisis de secuencias, entre algunos ejemplos se mencionan el análisis de textos, sonidos o video ([Calvo, 2018a](#)).

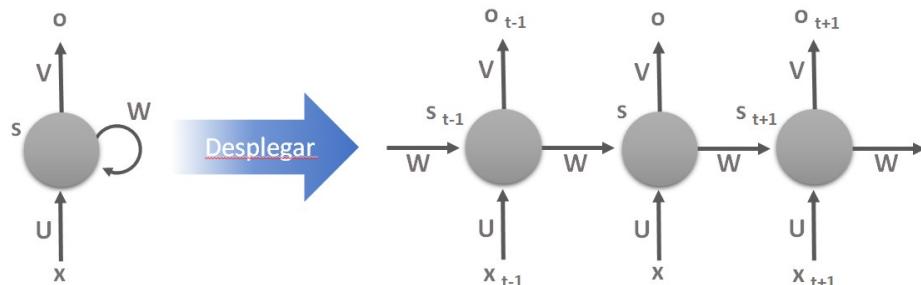


Figura 2.34: Ejemplo de red neuronal recurrente. Fuente: [Calvo, 2018a](#)

- **Máquina de Vectores de Soporte (SVM):** Es un algoritmo usado para tareas de regresión y clasificación, buscando un hiperplano en un espacio N-dimensional que clasifique claramente los puntos de datos a partir de la distancia máxima entre los puntos de datos de ambas clases. Para ello, maximiza la distancia del margen proporcionando cierto refuerzo para que los puntos de datos futuros puedan clasificarse con más confianza, es decir, que permita distinguir claramente dos clases, como se muestra en la Figura 2.35 ([Gandhi, 2018](#)).

Este algoritmo tiene sus orígenes en la década de los años 60 en Rusia, desarrollados por Vapnik y Chervonenkis. Inicialmente se enfocó en el reconocimiento óptico de caracteres

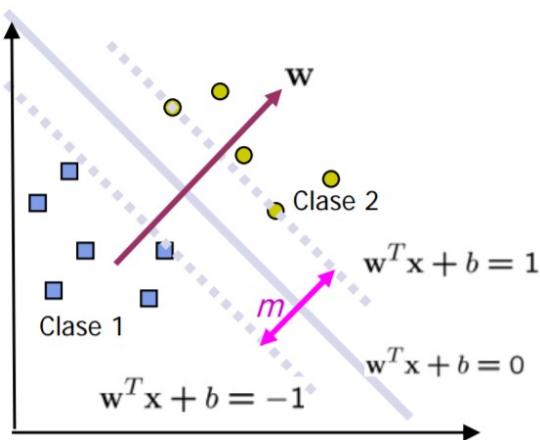


Figura 2.35: Hiperplano con dos clases separadas por una distancia m . Fuente: [Betancourt, 2005](#)

(OCR). Más tarde, los clasificadores de Vectores de Soporte se volvieron competitivos con los mejores sistemas disponibles en ese momento para resolver no solamente el anterior tipo de problema, sino también abarcar tareas de reconocimiento de objetos. En 1998, se publicó el primer manual de estos algoritmos por Burges. Y debido a sus grandes resultados obtenidos en la industria, actualmente se usa con frecuencia en el campo del aprendizaje automático ([Smola & Schölkopf, 2004](#)).

Los vectores de soporte hacen referencia a un pequeño subconjunto de las observaciones de entrenamiento que se utilizan como soporte para la ubicación óptima de la superficie de decisión ([MathWorks, s.f.](#)).

Una Máquina de Vectores de Soporte aprende la superficie decisión de dos clases distintas de los puntos de entrada. Como un clasificador de una sola clase, la descripción dada por los datos de los vectores de soporte es capaz de formar una frontera de decisión alrededor del dominio de los datos de aprendizaje con muy poco o ningún conocimiento de los datos fuera de esta frontera. Los datos son mapeados por medio de un *kernel* Gaussiano u otro tipo de *kernel* a un espacio de características en un espacio dimensional más alto, donde se busca la separación máxima entre clases. Cuando es traída de regreso al espacio de entrada, la función de frontera puede separar los datos en todas las clases distintas, cada una formando un agrupamiento. Esta teoría se basa en la idea de minimización de riesgo estructural (SRM), demostrando en muchas aplicaciones tener mejor desempeño que otros algoritmos de aprendizaje tradicional como las redes neuronales para resolver problemas de clasificación ([Betancourt, 2005](#)).

Cabe mencionar que hay casos en que el conjunto de datos de dos clases puede ser separables no necesariamente de forma lineal. En la Figura 2.36 se observan casos linealmente y no linealmente separables, respectivamente.

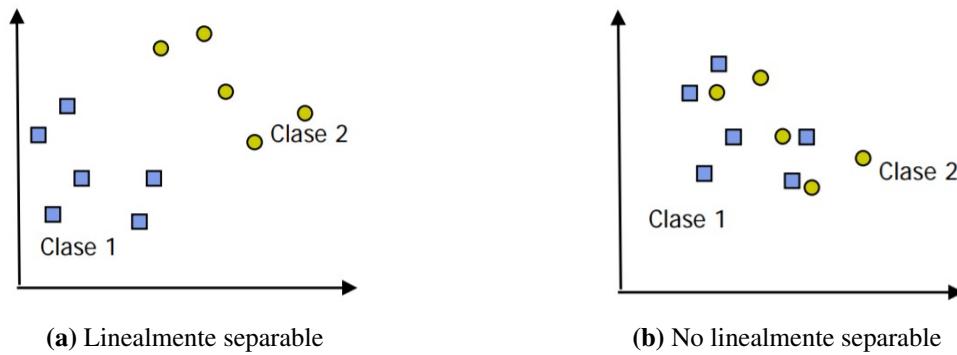


Figura 2.36: Ejemplo de separación de 2 clases. Fuente: [Betancourt, 2005](#)

Lo que se debe hacer para el primer caso es crear el hiperplano a través de una función lineal $w * z + b = 0$ y, definido el par (w, b) , separar el punto x_i según la Ecuación 2.20:

$$f(x_i) = \text{sign}(w * z + b) = \begin{cases} 1 & y_i = 1 \\ -1 & y_i = -1 \end{cases}$$

Ecuación 2.20: Ecuación del hiperplano para clasificar dos clases. Fuente: [Betancourt, 2005](#)

Para el segundo caso, debido a su mayor complejidad, se puede introducir algunas variables no-negativas a la función del hiperplano para hallar su valor óptimo; o también es viable utilizar una función *kernel* que calcule el producto punto de los puntos de entrada en el espacio de características Z, como se aprecia en la Figura 2.37.

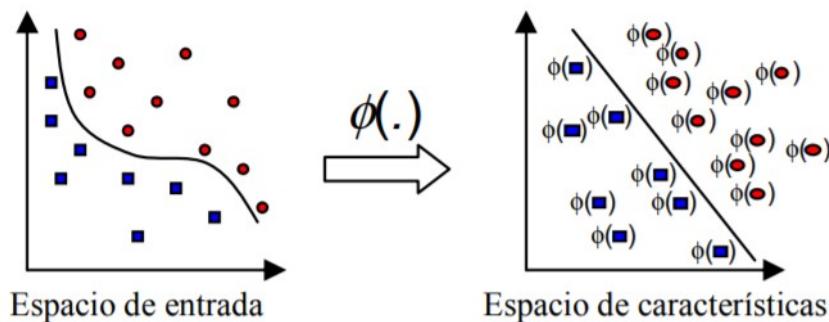


Figura 2.37: Aplicación de un kernel para transformar el espacio de los datos. Fuente: [Betancourt, 2005](#)

- **Árboles de Decisión:** Representación visual de decisiones y toma de decisiones utilizada en la minería de datos para derivar una estrategia y alcanzar un objetivo particular. Se dibuja boca abajo con su raíz en la parte superior. Consta de nodos internos, los cuales se subdividen en ramas o bordes y su contenido, las hojas o decisiones ([Gupta, 2017](#)).

Un árbol de decisión toma como entrada un objeto descrito a través de un conjunto de atributos y devuelve una “decisión”. Estos pueden ser discretos o continuos. La salida puede tomar cualquiera de estos dos tipos de valores; en el caso que aprenda una función tomando valores discretos se le denominará clasificación, y en el caso que la función sea continua será llamada regresión. En las clasificaciones booleanas, es decir de dos valores o binaria, clasificará como verdadero (positivo) o falso (negativo). Para alcanzar una decisión, el árbol desarrolla una serie de pruebas a través de sus nodos y las ramas que salen del nodo son etiquetadas con los valores posibles de dicha propiedad. Además, cada nodo hoja del árbol representa el valor que ha de ser devuelto si es alcanzado ([Russell & Norvig, 2004](#)).

Por ejemplo, representando un ejemplo de este algoritmo, se ilustra en la Figura 2.38 para decidir si se debe esperar por una mesa en un restaurante.

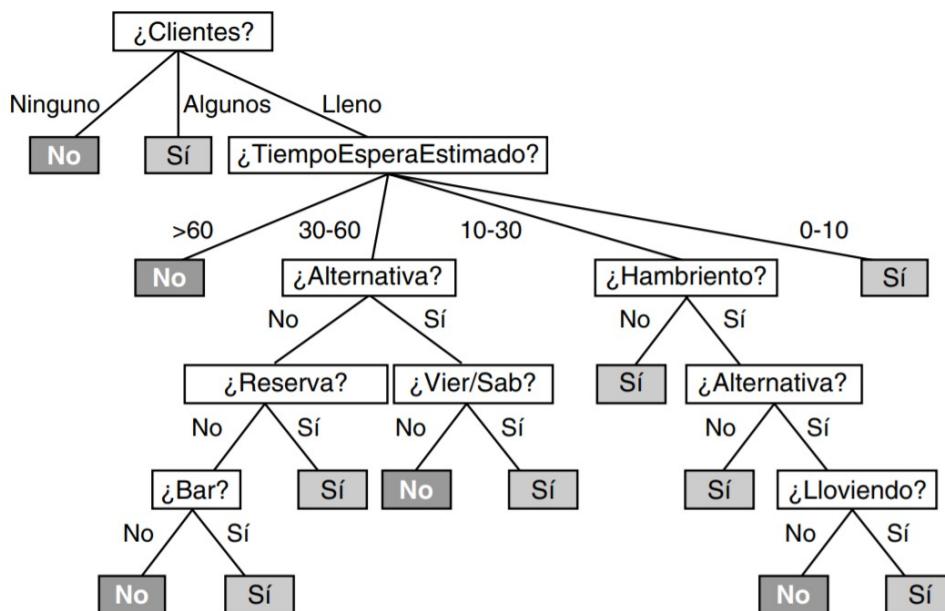


Figura 2.38: Ejemplo del algoritmo de árbol de decisión. Fuente: [Russell y Norvig, 2004](#)

2.2.8. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural (*Natural Language Processing (NLP)* por su nombre en inglés) es un campo interdisciplinario que combina lingüística computacional, ciencias de la computación, ciencia cognitiva e inteligencia artificial. Investiga el uso de las computadoras para entender el lenguaje humano con el fin de realizar tareas útiles, así como lograr una interacción entre ambas partes. Algunas de estas tareas son, por ejemplo, reconocimiento de voz, comprensión del lenguaje hablado, sistemas de diálogo, análisis léxico, análisis de sentimientos, entre otros (Deng & Liu, 2018).

Los autores Deng y Liu explican en su libro *Deep Learning in Natural Language Processing* el desarrollo del estudio de este campo, separando desde una perspectiva histórica en 3 olas: el Racionalismo, el Empiricismo y el Aprendizaje Profundo.

El Racionalismo, nomenclatura establecida entre las décadas de los años 60s y 80s de acuerdo a los argumentos por Noam Chomsky sobre la concepción del lenguaje humano, se basa en el conocimiento de este último fijado por herencia genética y concebido desde el nacimiento. Las primeras apariciones de la aplicación de este enfoque se remontan a la década de los años 50s, donde Alan Turing, en los experimentos que se conocen como "las pruebas de Turing", intentó simular conversaciones de lenguaje natural entre un humano y un computador para generar respuestas similares a la de una persona con el fin de poder evaluar las habilidades que ellas pueden alcanzar. Durante las décadas de los 70s y 80s, los sistemas de entendimiento de lenguaje hablado y sistemas de diálogo se basaron en conjuntos de reglas, desarrollados por la ingeniería de conocimiento experto.

El Empirismo, la segunda ola, se caracterizó por la explotación del cuerpo de texto (*data corpora*) y su uso con Aprendizaje Automático. Esta idea plantea que la mente humana comienza con operaciones de asociación, patrones de reconocimiento y generalización. Algunos ejemplos son el Modelo Oculto de Markov (HMM) y los modelos de traducción de IBM.

El Aprendizaje Profundo, la tercera y última ola, se basa en el uso de estas técnicas para resolver problemas de Lenguaje Natural que el Aprendizaje Automático no puede lograr para entrenar con grandes cantidades de datos, por ejemplo. Las más comunes son las Redes Neuronales Artificiales, debido a que su configuración permite personalizar la arquitectura basada en múltiples capas e hiperparámetros para soportar el entrenamiento con volúmenes considerables. Sin embargo, pese a sus ventajas, algunas de las limitaciones que presenta son justamente este último detalle para lograr resultados estadísticos impresionantes, mucha capacidad computacional, o habilidades pobres para entender las relaciones de inter-oracional como frases o palabras progresivas dentro de una oración.

Algunas técnicas de Aprendizaje Profundo utilizados actualmente para resolver problemas de NLP comprenden modelos anteriormente mencionados como las Redes Neuronales Convolucionales (CNN) o las Redes Neuronales Recurrentes (RNN). A continuación, se detallarán algunas de las más usadas, así como las principales características y diferencias de ejemplos ya explicados bajo este contexto.

- **Redes Neuronales Convolucionales (*Convolutional Neural Networks*):** Entre las técnicas de minería de datos, se explicaron los conceptos y tipos de Redes Neuronales Artificiales, entre ellas, la actual mencionada. Se comentó que entre sus mayores usos se dan actualmente en el tratamiento de imágenes, problemas de clasificación a partir de estas y visión por computador. El ejemplo más popular fue ImageNet desarrollado por Yann LeCun, informático reconocido por ser el fundador de este tipo de redes, para reconocer objetos dentro de imágenes.

Sin embargo, también son utilizadas para problemas de clasificación de texto. Ronan Collobert y Jason Weston fueron los pioneros de la aplicación de las CNN en tareas de procesamiento de lenguaje natural, modificando y adaptando su arquitectura y parámetros internos ([U. Kamath y col., 2019](#)). En la Figura 2.39 se ilustra la arquitectura de una CNN para problemas de NLP.

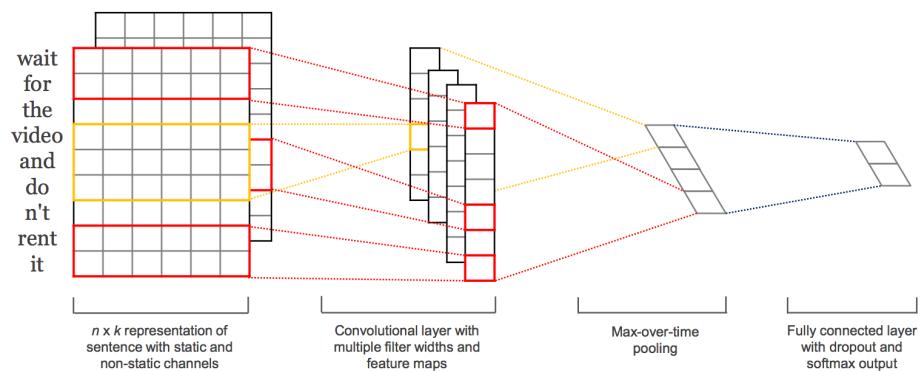


Figura 2.39: Arquitectura de modelo CNN con 2 canales para una oración de ejemplo. Fuente: [Kim, 2014](#)

Una de las principales diferencias entre ambos tipos de aplicación es que las convoluciones para imágenes son bidimensionales (2d) debido a que se desplazan a través de matrices de 2 dimensiones (largo y ancho). Mientras que las convoluciones unidimensionales (1d) son muy útiles para series de tiempo y operaciones de NLP por estar conformadas por vectores, aprendiendo así patrones en la dimensión de secuencia ([Rao & McMahan, 2019](#)). En la Figura 2.40 se observa una comparación entre ambos tipos de convolución según el tamaño de dimensión.

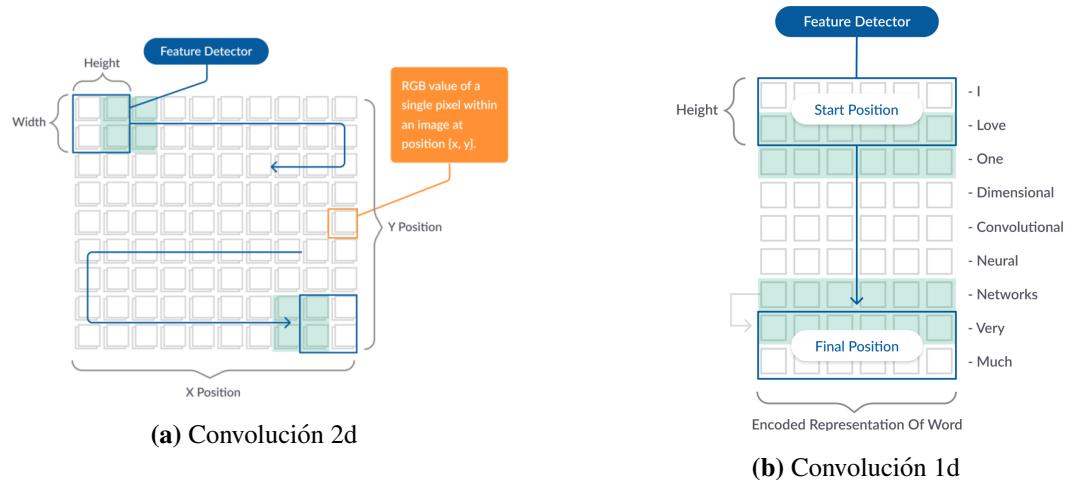


Figura 2.40: Diferencias entre convoluciones según su dimensión. Fuente: [MissingLink.ai, s.f.](#)

En la anterior imagen, donde cada palabra codificada se representa por un vector, un kernel de convolución de tamaño de 2 bloques (*kernel_size*) recorre toda la oración con paso (*stride*) de 1 bloque.

Para problemas de clasificación de texto como los que se considera en esta investigación y en algunos antecedentes con contenido textual, el proceso de la arquitectura CNN de manera general se basa en la Figura 2.41.

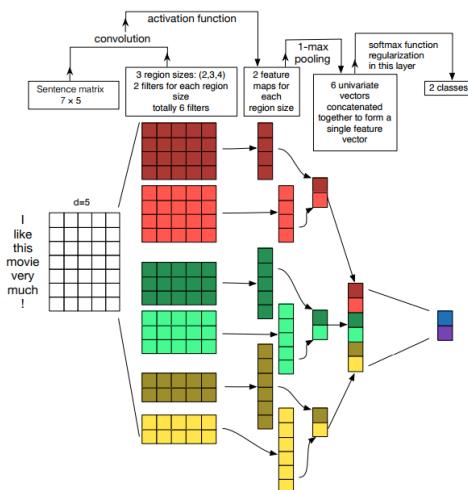


Figura 2.41: Arquitectura de modelo CNN para clasificación de oraciones. Fuente: [Zhang y Wallace, 2017](#)

La idea general es básicamente formar vectores de palabras codificadas para generar una matriz, la cual al ser recorrida por filtros de una dimensión determinada, se obtengan mapas de características para la siguiente entrada. De cada capa resultante, se agruparán (*pooling*) según el criterio del usuario (puede ser valor máximo, mínimo, promedio, etc)

para generar nuevos vectores univariantes que serán concatenados y luego el valor final de predicción regularizado entre un rango dependiendo de la función de activación asignada para el problema (*sigmoid* para clasificación de 2 clases o *softmax* a partir de 3 clases).

- **Redes Neuronales Recurrentes (Recurrent Neural Networks):** Este tipo de redes también fue comentada brevemente en los tipos de redes neuronales más conocidas. Las RNN son muy usadas incluso también en series de tiempo, ya que los datos para estas casuísticas son secuencias, es decir, una colección ordenada de elementos. Para el caso del lenguaje humano, en donde el habla es un conjunto de secuencia de palabras llamadas fonemas, se busca predecir la siguiente palabra en una oración dada a partir de un elemento dependiente, en este caso, las palabras previas (Rao & McMahan, 2019). Como ejemplos de estos casos de uso más comunes se mencionan a los motores de búsqueda en navegadores o sitios web, traductores, entre otros.

Según [Brownlee, 2017a](#), las RNN generalmente más usadas son los siguientes 3 tipos:

- **RNN Simple (Elman Network o S-RNN):** Son el tipo de redes neuronales recurrentes más básicas, propuesto por Jeffrey L. Elman en 1990, cuya arquitectura se caracteriza por la secuencia de elementos, como en la Figura 2.35. Las S-RNN proporcionan resultados sólidos para el etiquetado de secuencias, así como para el modelado del lenguaje.

La ecuación para representar un estado determinado en una RNN toma la siguiente forma:

$$s_i = R_{SRNN}(x_i, s_{i-1}) = g([s_{i-1}; x_i]W + b)$$

Ecuación 2.21: Fórmula para determinar un estado en una S-RNN. Fuente: [Brownlee, 2017a](#)

Donde se observa que un estado depende de la información del estado previo.

- **Memoria Larga a Corto Plazo (Long-Short Term Memory o LSTM):** Este modelo fue desarrollado para encargarse del problema de los gradientes que desaparecen de la RNN Simple, que limitaba más adelante el entrenamiento de las RNN profundas. Según el mismo autor, este problema surge debido a que los gradientes incluyen la multiplicación repetida de la matriz W (de la Ecuación 2.21), haciendo que los valores desaparezcan.

Esta arquitectura divide el vector de un estado observado s_i en dos mitades, donde una es tratada como “celdas de memoria” y la otra es la memoria de trabajo. Las del primer grupo tienen como función preservar la memoria, así como también los gradientes de error, a lo largo del tiempo. Son controladas mediante componentes

de puerta diferenciables, que son funciones matemáticas suaves simuladoras puertas lógicas. En cada estado de entrada, se utiliza una puerta para decidir qué cantidad de la nueva entrada se debe escribir en la memoria.

La arquitectura LSTM es actualmente la más exitosa dentro de las RNN. Un ejemplo de su representación puede ilustrarse en la Figura 2.42.

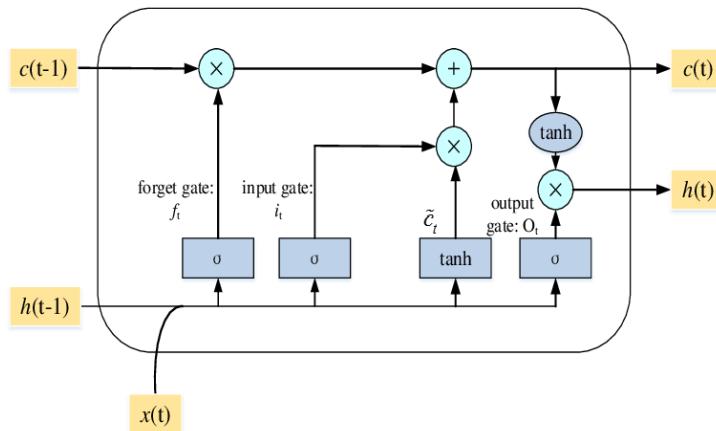


Figura 2.42: Arquitectura de una LSTM. Fuente: [X. Yuan y col., 2019](#)

En donde x son las entradas, c_{t-1} y c_t los estados de la celda, y h_{t-1} y h_t las salidas. Una variante muy útil de las RNN son las RNN Bidireccionales (*Bidirectional RNN* o *BiRNN*). Mientras una red neuronal recurrente toma una palabra del pasado para predecir el siguiente, las bidireccionales permiten mirar arbitrariamente lejos tanto al pasado como al futuro dentro de una secuencia (Goldberg, 2017). La ventaja que se logra a partir de estas características es que, por ejemplo, el modelo puede identificar y discernir mejor en la predicción de la siguiente palabra en el caso de existir 2 o más secuencias idénticas (problema de reconocimiento de entidades nombradas) al lograr conocer más información.

Ng, fundador de Coursera, DeepLearning.AI, Landing AI y Google Brain, explica lo anterior con el siguiente ejemplo como parte del curso Redes Neuronales Recurrentes (Ng, 2018):

Sean las oraciones *He said, “Teddy bears are on sale!”* y *He said, “Teddy Roosevelt was a great President!”*, se desea saber si *Teddy* es parte del nombre de una persona. El problema es que se cuenta con 2 oraciones cuya secuencia inicial de 3 palabras resultan ser las mismas, pero con distinto panorama posterior a estas. Por ello, la red es duplicada pero con orden de secuencia invertida y colocada en paralelo con la original para que cada una de las nuevas capas conecten sus salidas con las preexistentes y originen una nueva predicción que toma información previa y posterior. Bajo la misma premisa, pero con la oración de ejemplo *The brown fox jumped over*

the dog, se representa la arquitectura de una BiRNN en la Figura 2.43.

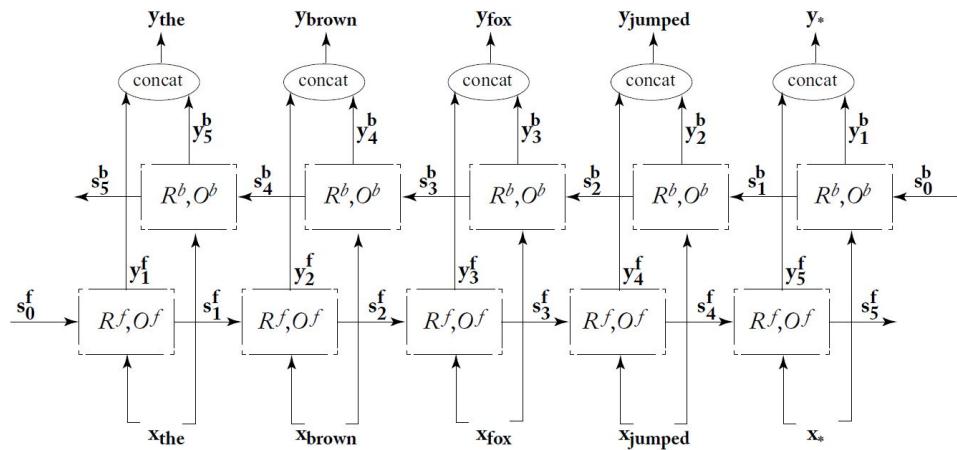


Figura 2.43: Representación de arquitectura BiRNN de la palabra *jumped* en la oración. Fuente: [Goldberg, 2017](#)

Como parte de las BiRNN, se encuentra la LSTM Bidireccional (*Bidirectional LSTM* o *BiLSTM*), mejora de la LSTM, que sobresale particularmente en la representación de palabras en la secuencia junto con sus contextos, capturando la palabra y las incontables posibilidades existentes a su alrededor ([Deng & Liu, 2018](#)).

- **Unidad Recurrente Cerrada (Gated Recurrent Unit o GRU):** Este modelo se desarrolló con la intención de simplificar la LSTM debido a su complejidad. Al igual que esta, consiste en un mecanismo de puertas pero en menor cantidad y sin un componente de memoria separada (ver Figura 2.44). La red GRU muestra ser efectiva en modelamiento de lenguaje y máquina traductora ([Goldberg, 2017](#)).

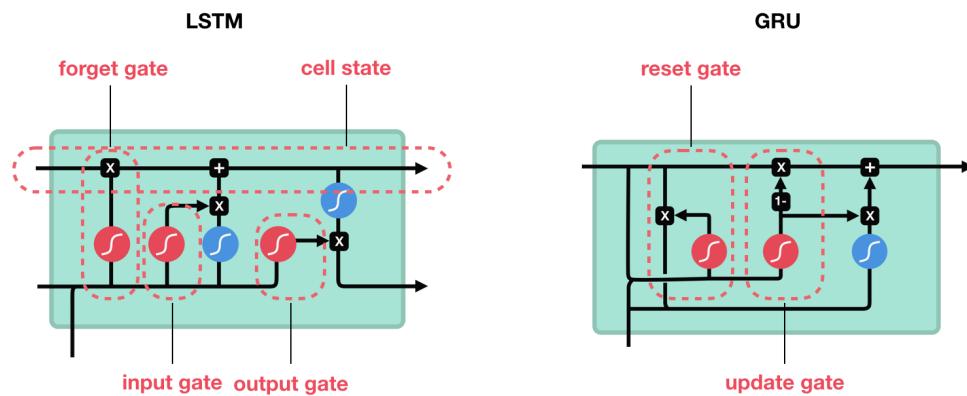


Figura 2.44: Comparación entre arquitecturas LSTM y GRU. Fuente: [Phi, 2018](#)

Sin embargo, al compararse los resultados entre estos 3 tipos de modelos, por lo general el mejor desempeño tiene la LSTM ([Brownlee, 2017a](#)).

- **Modelo Secuencia a Secuencia (Sequence-to-Sequence Model o Seq2seq Model):** También conocida como *Encoder-Decoder* (Codificador-Decodificador por su traducción al español), es un tipo de generador de lenguaje natural (*Natural Language Generation* o NLG) usado comúnmente para traducción (por ejemplo, Google Traductor). Se basa en 2 capas LSTM, en donde la primera es usada para codificar la oración de entrada en un “vector de pensamiento”, y la otra para decodificar en una respuesta (ver Figura 2.45) (Deng & Liu, 2018).

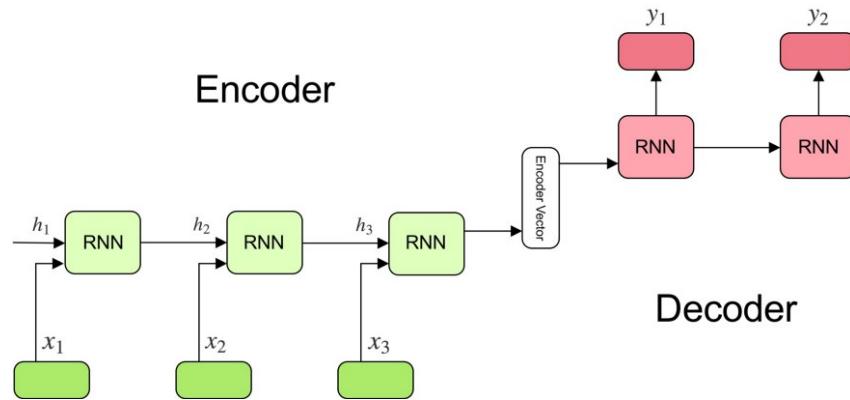


Figura 2.45: Arquitectura de un modelo Seq2seq. Fuente: Kostadinov, 2019

- **Aprendizaje Profundo Multimodal y Multitarea (Multimodal and Multitask Deep Learning):** Son un tipo de aprendizaje basado en la explotación de representaciones latentes en las redes neuronales profundas agrupando distintas modalidades (por ejemplo, audio, imágenes, texto, videos, etc) o múltiples tareas (predicción, clasificación, series de tiempo, entre otros) como en la Figura 2.46 (Deng & Liu, 2018)

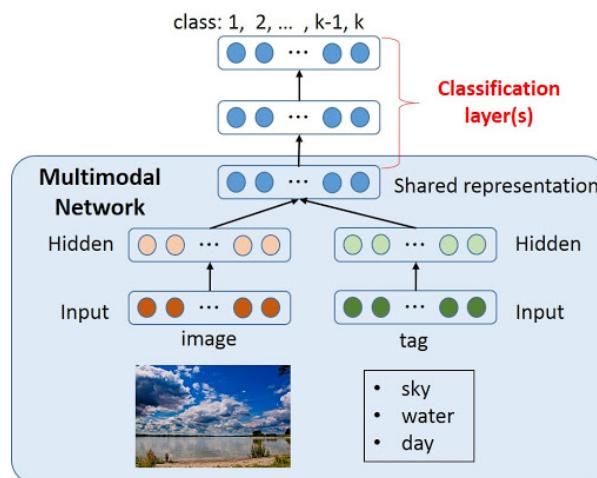


Figura 2.46: Arquitectura de un modelo multimodal de imágenes y texto. Fuente: Nishida y Nakayama, 2015

El contenido textual que se usa en los modelos anteriores debe ser pre-procesada previamente. El texto en sí no puede ser incluido tal cual en los modelos de Aprendizaje Automático o Aprendizaje Profundo sin antes ser limpiado ([Brownlee, 2017a](#)). Python ofrece una librería para trabajar y modelar texto llamada Natural Language Toolkit o NLTK. Algunas de las funciones disponibles se encuentran separación de texto en oraciones, separación en palabras o tokenización, eliminación de signos de puntuación, eliminación de palabras de parada (*stop words*), reducción de palabras hacia su forma raíz (*stemming*), retorno de palabras hacia su base o forma diccionario (*lemmatization*).

Suprimir contenido como signos de puntuación, caracteres especiales, palabras de parada, enlaces web, entre otros, ayuda a reducir la cantidad de vectores innecesarios para las representaciones de palabras que se utilizan en la fase de entrenamiento. Para ello, la secuencia de entrada debe dividirse en *tokens*, que pueden ser una palabra, oración, párrafo.

NLTK ofrece más de una alternativa para la tokenización, como por ejemplo, *WhitespaceTokenizer* (elimina espacios en blanco para separar palabras y signos de puntuación en una oración, estos últimos no son independientes), *TreebankWordTokenizer* (separa palabras y signos de puntuación en una oración de manera independiente) y *WordPunctTokenizer* (funciona igual que TreebankWordTokenizer pero no distingue contracciones en idiomas como el inglés). La elección de alguna de estas depende del objetivo que busque el usuario. Asimismo, la reducción de palabras hacia su raíz (llamada *stem*) permite suprimir los prefijos o sufijos agregados a una palabra. Sin embargo, presenta problemas en formas irregulares, generando “No palabras”. El retorno de palabras hacia su forma base (llamada *lemma*), por su lado, convertir palabras conjugadas en distintas variantes de tiempo. Aún así, no todas las formas pueden ser reducidas. La finalidad de estos 2 últimos es reducir una palabra a su forma más primitiva (expresiones regulares) para generar un diccionario homogéneo ([Zimovnov, 2018](#)).

Luego de la limpieza de texto, el nuevo conjunto de datos debe representarse como vectores. Dentro de las modalidades más usadas para representación de palabras se encuentran:

- **Incrustación de palabra (Word Embedding)**: Es una representación aprendida para texto en donde las palabras con el mismo significado tienen una similar representación. La principal ventaja que presenta es la baja dimensionalidad de vectores a nivel computacional, ya que una palabra individual se representa como vectores de valor real en un espacio vectorial predefinido, a diferencia de otros métodos de muy alta dimensionalidad como la codificación en caliente (*one hot encoding*) o la bolsa de palabras (*bag-of-words*), en donde distintas palabras presentan diferentes representaciones ([Brownlee, 2017a](#)).

Algunos de los algoritmos destacados de este grupo son:

- **Capa de incrustación (Embedding Layer):** Consiste en una incrustación de palabras que aprende con un modelo de red neuronal. En esta capa, se especifica el tamaño del espacio vectorial (dimensiones), donde los vectores inicializan con pequeños valores aleatorios. La capa se utiliza en el extremo frontal de la red, es decir, luego de la capa de entrada, y es ajustada de forma supervisada mediante el algoritmo de propagación hacia atrás. Puede recibir palabras codificadas, en donde cada una se representa por un código, o codificación en caliente, la cual presenta mayor dimensión. Si se utiliza un Perceptrón Multicapa (MLP), los vectores de palabras son concatenados antes de entrar al modelo. En el caso se use una RNN, cada palabra será tomada como una entrada en una secuencia (Brownlee, 2017a). Como se observa en la Figura 2.47, la capa de incrustación toma como entrada a la matriz de incrustación y la transforma en un vector tridimensional de n registros.

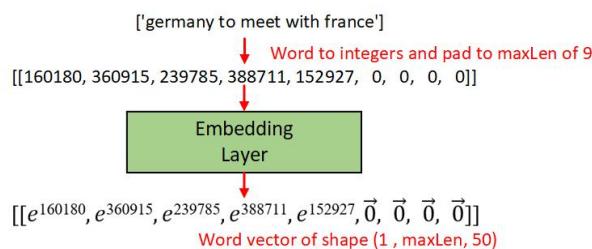


Figura 2.47: Ejemplo de funcionamiento de una capa de incrustación. Fuente: [Chengwei, 2018](#)

- **Palabra a vector (Word2Vec):** Se trata de un método estadístico, desarrollado en el 2013 por Tomas Mikolov, cuya finalidad es de aprender eficientemente una incrustación de palabras independientes de un corpus de texto, incluso si este y sus dimensiones son más grandes. En este trabajo se involucró el análisis de los vectores aprendidos y la exploración de la matemática vectorial para representar palabras. Para entender estos conceptos, se ilustra bajo el ejemplo de la analogía “Rey es a Reina como hombre es a mujer”, en donde se evidencia la relación sintáctica y semántica capturada por su proximidad vectorial (Brownlee, 2017a). La Figura 2.48 grafica las palabras relacionadas en un espacio vectorial gracias al algoritmo.

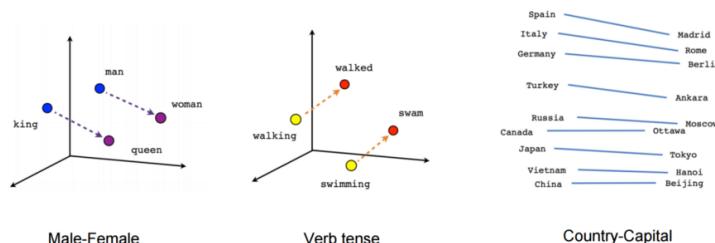


Figura 2.48: Incrustaciones de palabras por Word2Vec. Fuente: [Bujokas, 2020](#)

- Vectores globales para representación de palabra (GloVe):** Es un algoritmo de aprendizaje no supervisado para obtener representaciones vectoriales de palabras. Es una extensión del método Word2Vec, desarrollado en 2014 por Jeffrey Pennington como proyecto de código abierto en la Universidad Stanford. Las representaciones de palabras del modelo de espacio vectorial clásico se desarrollaron utilizando técnicas de factorización matricial como el Análisis semántico latente (LSA), con el cual combina sus características de aprendizaje adoptadas de Word2Vec, logrando un modelo de aprendizaje con mejor performance para incrustación de palabras (Pennington y col., 2014a). GloVe dispone en su web distintos vectores de palabras pre-entrenados, entre ellas, un vocabulario de más de 400 mil palabras, hasta 4 opciones de matrices con distintas dimensiones, y 6 billones de tokens extraídos de Wikipedia (2014) y Gigaword (5ta edición) de la Universidad de Pensilvania, así como datas públicas extraídas de mayor tamaño. La Figura 2.49 ilustra ejemplos de palabras relacionadas en un espacio vectorial gracias al algoritmo. La relación semántica (palabras clasificadas en un mismo grupo) entre ellas se aprecia por la cercanía en su eje Y. Mientras que aquellos señalados con líneas horizontales en su eje X se encuentran asociadas por analogías, como por ejemplo, género, geolocalización, compañía, grados del adjetivo, entre otros.

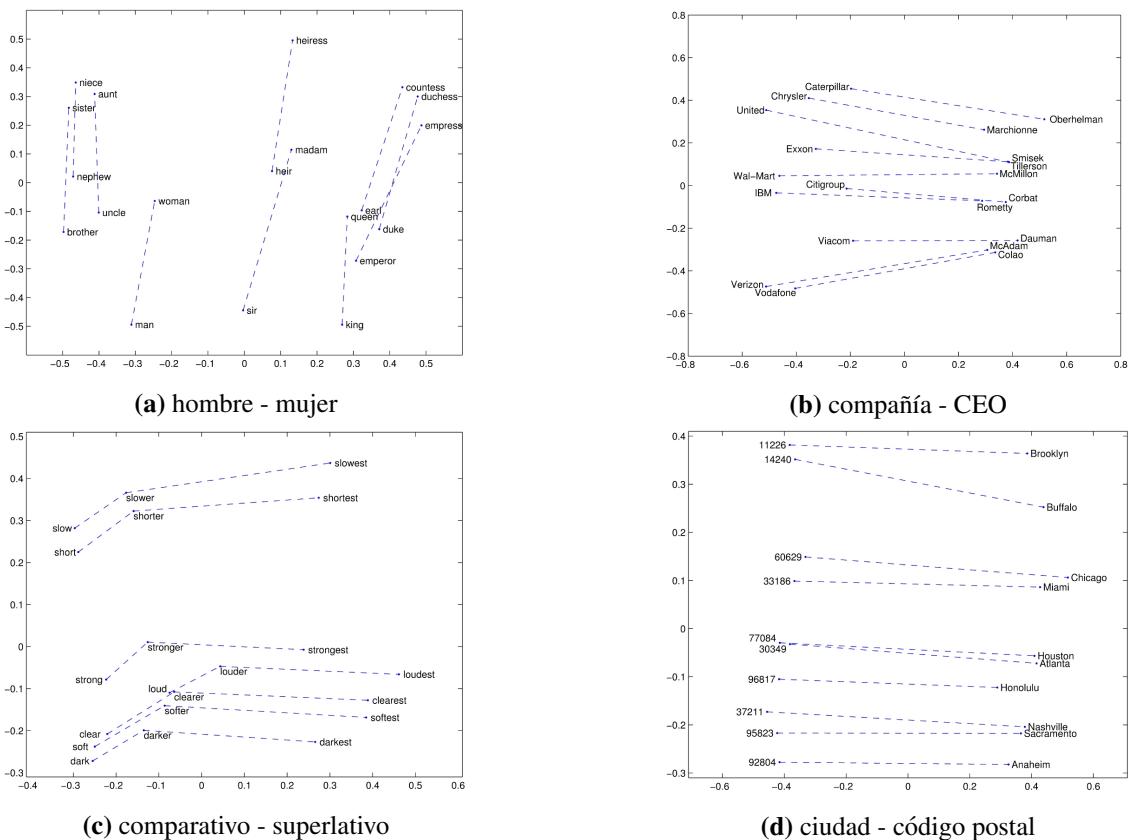


Figura 2.49: Incrustaciones de palabras por GloVe. Fuente: [Pennington y col., 2014b](#)

- **Bolsa de palabras (Bag-of-Words o BoW):** Consiste en una forma de extraer características de textos implicando un vocabulario de palabras conocidas y una métrica de presencia de éstas. Se le denomina Bolsa de palabras ya que no considera el orden o la estructura de las palabras dentro del documento, sólo se enfoca en conocer su presencia. El algoritmo relaciona la similitud entre documentos si tienen contenido similar ([Brownlee, 2017a](#)). Por ejemplo, en la Figura 2.50 el algoritmo cuantifica la frecuencia cada palabra individual dentro de un documento sin importar su orden. De esta manera, los relaciona a partir de su recurrencia en todos ellos

Document	the	cat	sat	in	hat	with
the cat sat	1	1	1	0	0	0
the cat sat in the hat	2	1	1	1	1	0
the cat with the hat	2	1	0	0	1	1

Figura 2.50: Ejemplo de funcionamiento de bolsa de palabras. Fuente: [V. Zhou, 2019](#)

- **Frecuencia del término - Frecuencia inversa del documento (Term Frequency - Inverse Document Frequency o TF-IDF):** Es un algoritmo que también contabiliza palabras pero que resaltan aquellas más significativas para calcular sus pesos correspondientes. Se compone por la frecuencia del término, que representa la frecuencia con la que una palabra determinada aparece en un documento, y la frecuencia inversa del documento, el cual reduce la escala de palabras que aparecen mucho en los documentos ([Brownlee, 2017a](#)). Su peso se calcula bajo la siguiente fórmula:

$$TF - IDF_{(n,d)} = TF_{(n,d)} * IDF_{(n)}$$

Ecuación 2.22: Fórmula de TF-IDF. Fuente: [Hamdaoui, 2019](#)

Donde:

$TF_{(n,d)}$ = número de veces que el término t aparece en un documento d

$IDF_{(n)} = \log \frac{1+n}{1+df_{(d,n)}} + 1$

n = número de documentos

$df_{(d,n)}$ = frecuencia de documentos d que contienen el término t

2.3. Marco Conceptual

2.3.1. Crowdfunding

El financiamiento colectivo o crowdfunding es un sistema que, utilizando internet como base de operaciones, busca generar una respuesta económica activa en el usuario ([López-Golán y col., 2017](#)).

Combinando ideas de microfinanzas y crowdsourcing, el crowdfunding es la práctica de financiar una empresa o un proyecto al recaudar muchas pequeñas cantidades de dinero de un gran número de personas. Este mecanismo de financiamiento ha sido recibido por los recaudadores de fondos, el público en general y los formuladores de políticas. La industria de crowdfunding ha crecido rápidamente en los últimos años, así como el crecimiento exponencial del número de plataformas de crowdfunding, el número de proyectos expuestos allí y el número de capital total recaudado en esos sitios web ([Xuefeng & Zhao, 2018](#)).

La cantidad de grupos de crowdfunding varía según distintos autores. Según Hollas, se clasifican en 4 modelos: basado en donaciones, recompensas, capital y deuda. Colgren reafirma lo anterior con la variante de crédito en vez de capital y por su lado, Collins, préstamo por deuda. Mientras que I. Lee y Shin solo consideran principalmente al crowdfunding basado en recompensas, donaciones y capital. El crowdfunding basado en capital social se encuentra actualmente limitado en los Estados Unidos debido a que la Regulación D de la Ley de Valores de 1933 prohíbe la participación de muchos potenciales inversionistas y los obliga a tener un ingreso anual mayor a \$ 200,000 o más de \$ 1 millón en patrimonio neto ([Lichtig, 2015](#)).

Además de los tipos mencionados en el párrafo anterior, Collins también afirma que el crowdfunding se puede dividir en 2 tipos de modelo de negocio ([H.-D. Lee, 2019](#)):

- **El modelo “Todo o Nada”** (*All-Or-Nothing* (AON) en inglés), el cual un proyecto logra ser financiado si es que alcanza o sobrepasa la meta durante el periodo de campaña. A partir de esto, el creador tiene como principal objetivo motivar a los patrocinadores. En caso de no lograrse el cometido, los montos aportados al proyecto serán devueltos, significando un riesgo menor para ellos. Esto, asimismo, permite evaluar a los dueños las variables de la campaña para apostar por un algún cambio o decisión que le permita encaminar al éxito del financiamiento. Hasta febrero del 2019, el ratio promedio de proyectos financiados exitosamente en Kickstarter (considerando todas las categorías de la plataforma) fue de 37 %. Sin embargo, aunque parezca un indicador con baja performance, según la misma compañía, el 82 % de los proyectos que alcanzan el 20 % de su meta son financiados exitosamente.

- El modelo “**Mantener todo**” (*Keep-In-All* (KIA) en inglés), muy similar al anterior, con la única diferencia que el creador del proyecto mantiene todo el dinero financiado después de acabar el plazo de días sin importar si alcanza o no la meta establecida. En contraste con el modelo AON, este presenta un ratio menor de éxito. Asimismo, la plataforma de crowdfunding de este tipo de modelo más popular es Indiegogo, que viene funcionando desde el 2008.

Algunos factores clave, relacionados a la eficiencia del proceso y la retención de clientes durante la campaña, que afectan a un crowdfunding exitoso son ([H.-D. Lee, 2019](#)):

- Seleccionar una plataforma correcta. Esta normalmente depende de los objetivos reales que busca el creador del proyecto durante la campaña. En el anterior punto se mencionaron las dos principales dirigidas a un tipo de modelo de negocio en particular cada una.
- Crear un proyecto atractivo. Las personas normalmente se motivan a contribuir económicamente por algún proyecto que sea de su interés. Para ello, deben ser fáciles de entender en general.
- Ofrecer diferentes opciones de recompensa. Estas pueden ir más allá de lo económico. Influirá mucho el nivel de creatividad que tenga el creador para llamar la atención del público.
- Promocionar un video. Además del contenido que pueda tener un proyecto, ayuda mucho el apoyo de material audiovisual que complemente o explique brevemente la descripción de este. Brinda una mejor impresión y además, para las generaciones más jóvenes, es más útil que leer un enunciado extenso.

2.3.2. Kickstarter

Kickstarter, desde su inicio en 2009, es una plataforma de financiamiento de proyectos creativos de todo tipo, los cuales incluyen películas, juegos, música, arte, diseño y tecnología. Actualmente, se han registrado más de 162 mil proyectos realizados, 16 millones de contribuyentes y 4,3 miles de millones de dólares fondeados ([Kickstarter, s.f.-a](#)). La plataforma utiliza un modelo de financiamiento llamado “Todo o Nada” (AON), el cual consiste en que, si un proyecto no alcanza su meta de financiamiento en un determinado plazo de tiempo, no se realiza ninguna transacción de fondos, como se explicó en el anterior punto ([Kickstarter, s.f.-d](#)). Si bien los patrocinadores apoyan estos proyectos por motivos personales y distintos para hacerlos

realidad, ellos no obtienen la propiedad o los ingresos de los proyectos que financian, sino que los creadores conservan la totalidad de su trabajo ([Kickstarter, s.f.-f](#)).

2.3.3. Proyecto

Un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único. La naturaleza temporal de los proyectos indica un principio y un final definidos, y que el propósito se alcanza cuando se logran los objetivos del proyecto o cuando este es terminado por no cumplir sus objetivos, o cuando ya no existe la necesidad inicial que dio origen al mismo ([Project Management Institute, 2017](#)).

A partir de este concepto enunciado por el PMI, se entiende que un proyecto parte de una idea que un individuo o conjunto de individuos tienen en mente para convertirla en realidad con el fin de responder a una necesidad.

En Kickstarter, los proyectos que pueden ser financiados deben pertenecer a las siguientes categorías: Arte, Cómics, Artesanías, Baile, Diseño, Moda, Cine y vídeo, Comida, Juegos, Periodismo, Música, Fotografía, Publicaciones, Tecnología, y Teatro. Cualquier tipo de persona puede contribuir al mismo desde ser patrocinadores hasta formar parte de las principales referencias ([Kickstarter, s.f.-c](#)).

Dentro de las normas internas de la plataforma para la creación de proyectos se especifica que cada proyecto debe resultar ser totalmente nuevo, original e innovador que pueda ser compartido con el público, así como haber sido presentados de forma honesta y clara. En adición a esto, también se menciona que las recaudaciones que se darán no podrán ser otorgadas a obras benéficas sino cumplir con el objetivo de llevar a cabo el proyecto planteado, al igual que está prohibido asimismo del ofrecimiento de incentivos financieros como resultado ([Kickstarter, s.f.-e](#)).

2.3.4. Campaña

Una campaña puede abarcar diversos términos si es que se busca su concepto en fuentes confiables como la Real Academia Española debido al alcance y empleabilidad del mismo. El significado más próximo que la RAE enuncia sobre campaña es la de un “conjunto de actos o esfuerzos de índole diversa que se aplican a conseguir un fin determinado” ([Real Academia Española, 2020](#)). Para el actual contexto, el término se refiere específicamente a la campaña publicitaria, la cual se define como una estrategia diseñada y ejecutada en diferentes medios para obtener objetivos de notoriedad, ventas y comunicación de una determinada marca o producto

a través del uso de la publicidad ([Cyberclic, s.f.](#)).

En la plataforma de Kickstarter, existen personas que tienen ideas y proyectos en mente, buscando financiarlos en el sitio web. Para lograr su objetivo y siguiendo la regla del “todo o nada”, estos individuos realizan eventos de promoción para atraer entre personas conocidas suyas y cibernautas en general. A esta serie de eventos de promoción se le conoce como campañas. Las campañas exitosas se basan en que un determinado proyecto alcanza a ser financiado en el tiempo estimado gracias a algunas de las siguientes características ([Kickstarter, s.f.-b](#)):

- Logran ser claros y concisos sobre lo que su proyecto busca alcanzar al ser financiado. Se definen bien las características del proyecto mediante detalles en la descripción, videos e imágenes precisas.
- Indican los beneficios y recompensas que los patrocinadores obtendrán si es que la campaña es exitosa y el proyecto se financia.
- Atrae e interactúa con una gran cantidad de público, manteniendo constante comunicación acerca de actualizaciones y novedades de la campaña y resolviendo dudas que puedan darse.
- Se estiman costos de manera eficiente que pueden ser solventados para cubrir más allá del proyecto una vez financiado, incluido los que se originan para la entrega del producto a los patrocinadores. Se logra convencerlos.

Adicional a estos puntos, se descubrió en una investigación con más de 27 mil proyectos en Kickstarter que, a pesar que el número de proyectos iniciados por varones es dos veces más que de mujeres, el ratio de éxito de financiamiento es mayor en ellas. Esto debido a que las creadoras, en general, establecen objetivos más realísticos y con un tiempo límite más bajo para cumplir sus objetivos, transmitiendo así calidad y confianza para que los inversores actúen más rápido ([Ullah & Zhou, 2020](#)).

Otro factor clave es el rol importante de los patrocinadores en el éxito de un proyecto financiado. Sus principales motivaciones vienen dadas a factores como las recompensas por el aporte económico al proyecto, el nivel de innovación que este ofrece, la participación social que se logre captar, entre otros. Podrían clasificarse estas motivaciones en 6 grupos: interés, diversión, filantropía, recompensa, relación y reconocimiento. El resultado, 4 tipos de patrocinadores: los angelicales (donantes tradicionales), cazadores de recompensas (inversores del mercado), ávidos fanáticos (apasionados como los miembros de una comunidad de marca) y ermitaños de buen gusto (fanáticos discretos pero entusiastas) ([Tung & Liu, 2018](#)).

Capítulo 3

METODOLOGÍA DE LA INVESTIGACIÓN

3.1. Diseño de la investigación

En esta sección del documento se explica cuál fue el diseño, el tipo y el enfoque del trabajo de investigación, así como también la población y la muestra.

3.1.1. Enfoque de la investigación

El presente trabajo tuvo un enfoque cuantitativo ya que se busca diseñar y desarrollar instrumentos, en este caso modelos predictivos, para responder al problema estudiado a partir de medición de datos históricos en la plataforma Kickstarter con herramientas basadas en la estadística y matemáticas que puedan ser interpretadas por cualquier investigador.

3.1.2. Alcance de la investigación

El alcance del presente trabajo fue descriptivo ya que se recolectaron datos en un determinado rango de tiempo (desde 2009 hasta el presente año 2019) para describir el comportamiento de las campañas de proyectos tecnológicos en Kickstarter a partir de las características de sus variables y con ello, pronosticar su posible éxito o fracaso antes de finalizar la campaña con un nivel óptimo de precisión.

3.1.3. Tipo de la investigación

Para determinar el tipo de la investigación, primero fue necesario definir el actual trabajo como Diseño Experimental ya que las variables que se tienen serán controladas, es decir, fueron agregadas o removidas en los modelos construidos en el experimento para analizar el impacto que estos tuvieron en los resultados obtenidos. Dentro de esta categoría se clasifica como Diseño Experimental Puro ya que se buscó medir la variable dependiente, en este caso *state* (el estado actual del proyecto en Kickstarter) a partir de la manipulación de las demás variables independientes agregando o desagregándolas para comparar los rendimientos obtenidos de los instrumentos de medición y determinar cuáles de ellas finalmente fueron tomadas en cuenta.

3.1.4. Descripción del prototipo de investigación

Teniendo como referencia investigaciones basadas en Aprendizaje Profundo Multimodal, es decir, que combinan distintas características de una campaña de crowdfunding ([R. S. Kamath y Kamat, 2018](#), décimo antecedente; [Jin y col., 2019](#), decimotercer antecedente; [Cheng y col., 2019](#), decimocuarto antecedente) usando Metainformación ([K. Chen y col., 2013](#), primer antecedente; [Mitra y Gilbert, 2014](#), segundo antecedente; [M. Zhou y col., 2015](#), tercer antecedente; [S.-Y. Chen y col., 2015](#), cuarto antecedente; [Beckwith, 2016](#), quinto antecedente; [Y. Li y col., 2016](#), sexto antecedente; [H. Yuan y col., 2016](#), séptimo antecedente; [Sawhney y col., 2016](#), octavo antecedente; [Kaur y Gera, 2017](#), noveno antecedente; [R. S. Kamath y Kamat, 2018](#), décimo antecedente; [P.-F. Yu y col., 2018](#), undécimo antecedente; [Jin y col., 2019](#), decimotercer antecedente; [Cheng y col., 2019](#), decimocuarto antecedente), descripción del proyecto ([Mitra y Gilbert, 2014](#), segundo antecedente; [M. Zhou y col., 2015](#), tercer antecedente; [H. Yuan y col., 2016](#), séptimo antecedente; [Sawhney y col., 2016](#), octavo antecedente; [S. Lee y col., 2018](#), duodécimo antecedente; [Jin y col., 2019](#), decimotercer antecedente; [Cheng y col., 2019](#), decimocuarto antecedente; [L.-S. Chen y Shen, 2019](#), decimoquinto antecedente; [Chaichi y Anderson, 2019](#), decimosexto antecedente) y comentarios de los patrocinadores acerca del mismo ([S.-Y. Chen y col., 2015](#), cuarto antecedente; [Y. Li y col., 2016](#), sexto antecedente; [S. Lee y col., 2018](#), duodécimo antecedente; [Jin y col., 2019](#), decimotercer antecedente; [Shafqat y Byun, 2019](#), decimoséptimo antecedente), la idea del prototipo final consistió en ensamblar estas 3 partes en un modelo apilado. Para ello, se representa cada una de las tres partes agrupadas en el marco de trabajo de la Figura 3.1.

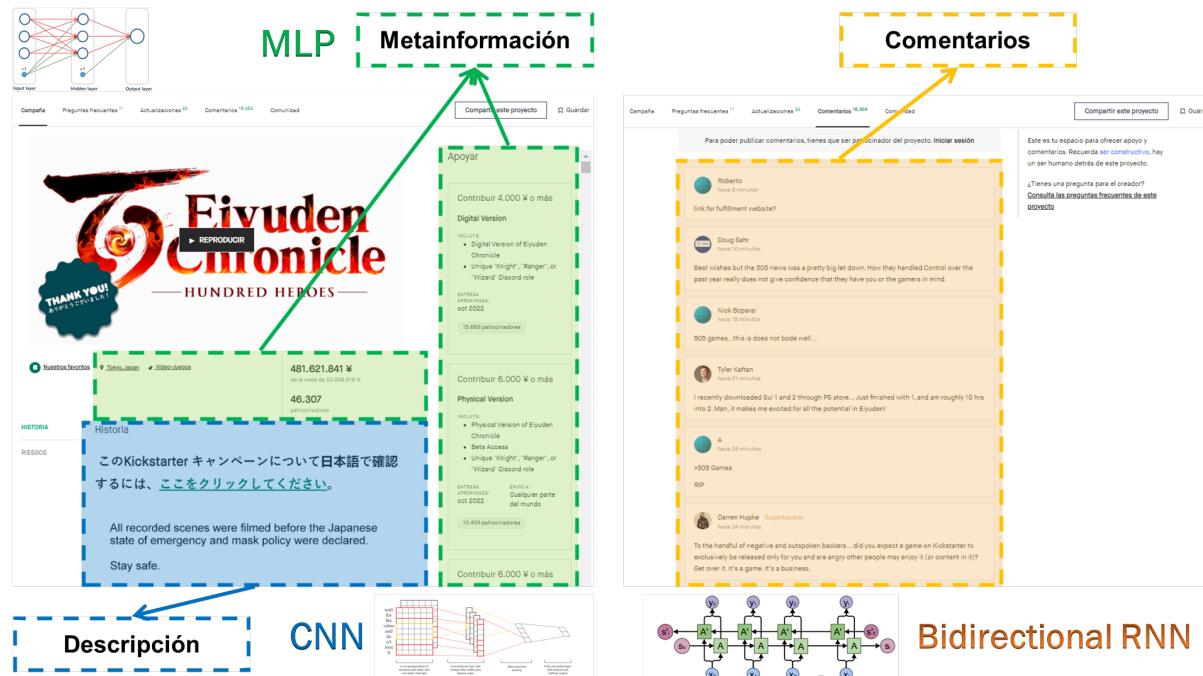


Figura 3.1: Marco de trabajo del prototipo final. Fuente: Elaboración propia

3.2. Población y muestra

3.2.1. Población

La población considerada para el presente trabajo fue de 27,251 proyectos en Kickstarter de la categoría tecnología incluyendo todas las subcategorías entre los períodos 2009-2019, en su mayoría del territorio de los Estados Unidos de América.

3.2.2. Muestra

Se distribuyeron los subconjuntos de entrenamiento y prueba en proporciones de 80 % (21,710 registros) y 20 % (5,541 registros) respectivamente ([H. Yuan y col., 2016](#), séptimo antecedente; [P.-F. Yu y col., 2018](#), undécimo antecedente; [L.-S. Chen y Shen, 2019](#), decimoquinto antecedente; [Mitra y Gilbert, 2014](#), segundo antecedente; [Sawhney y col., 2016](#), octavo antecedente).

3.2.3. Unidad de análisis

La unidad de análisis fue un proyecto en Kickstarter de la categoría tecnología de cualquier subcategoría entre los períodos 2009-2019 dentro del territorio de los Estados Unidos.

3.3. Operacionalización de Variables

En la Tabla 3.1 se presentan las variables usadas para el conjunto de datos final basado en contenido textual y metainformación. Estas fueron seleccionadas de acuerdo al Benchmarking aplicado a los 17 antecedentes en el Capítulo II.

Variable	Detalle	Tipo de dato
Variables independientes		
goal	Monto de la meta de financiamiento del proyecto.	float64
completeness	Porcentaje de financiamiento o completitud.	float64
duration	Duración de la campaña (en días).	int64
pledges_num	Cantidad de montos disponibles para contribuir.	int64
pledged	Monto contribuído en la campaña.	float64
pledges_median	Mediana de montos disponibles para contribuir.	float64
description	Descripción del proyecto.	object
comments	Comentarios de patrocinadores sobre el proyecto.	object
Variable dependiente		
state	Estado de financiamiento del proyecto.	object

Tabla 3.1: Diccionario de datos del conjunto final entrenado. Fuente: Elaboración propia.

Los autores citados por cada variable utilizada se mencionan a continuación:

- **goal:** [K. Chen y col., 2013](#) (primer antecedente), [Mitra y Gilbert, 2014](#) (segundo antecedente), [M. Zhou y col., 2015](#) (tercer antecedente), [S.-Y. Chen y col., 2015](#) (cuarto antecedente), [Y. Li y col., 2016](#) (sexto antecedente), [H. Yuan y col., 2016](#) (séptimo antecedente), [Sawhney y col., 2016](#) (octavo antecedente), [Kaur y Gera, 2017](#) (noveno antecedente), [R. S. Kamath y Kamat, 2018](#) (décimo antecedente), [P.-F. Yu y col., 2018](#) (undécimo antecedente), [Jin y col., 2019](#) (decimotercer antecedente), [Cheng y col., 2019](#) (decimocuarto antecedente).
- **completeness:** [S.-Y. Chen y col., 2015](#) (cuarto antecedente).

- **duration:** [Mitra y Gilbert, 2014](#) (segundo antecedente), [M. Zhou y col., 2015](#) (tercer antecedente), [Y. Li y col., 2016](#) (sexto antecedente), [Sawhney y col., 2016](#) (octavo antecedente), [Kaur y Gera, 2017](#) (noveno antecedente), [R. S. Kamath y Kamat, 2018](#) (décimo antecedente), [P.-F. Yu y col., 2018](#) (undécimo antecedente), [Jin y col., 2019](#) (decimotercer antecedente).
- **pledges_num:** [K. Chen y col., 2013](#) (primer antecedente), [Mitra y Gilbert, 2014](#) (segundo antecedente), [S.-Y. Chen y col., 2015](#) (cuarto antecedente), [H. Yuan y col., 2016](#) (séptimo antecedente), [Jin y col., 2019](#) (decimotercer antecedente).
- **pledged:** [K. Chen y col., 2013](#) (primer antecedente), [Y. Li y col., 2016](#) (sexto antecedente), [R. S. Kamath y Kamat, 2018](#) (décimo antecedente).
- **pledges_median:** [S.-Y. Chen y col., 2015](#) (cuarto antecedente)*, [Jin y col., 2019](#) (decimotercer antecedente)*.
- **description:** [Mitra y Gilbert, 2014](#) (segundo antecedente), [M. Zhou y col., 2015](#) (tercer antecedente), [H. Yuan y col., 2016](#) (séptimo antecedente), [Sawhney y col., 2016](#) (octavo antecedente), [R. S. Kamath y Kamat, 2018](#) (décimo antecedente), [S. Lee y col., 2018](#) (duodécimo antecedente), [Jin y col., 2019](#) (decimotercer antecedente), [Cheng y col., 2019](#) (decimocuarto antecedente), [L.-S. Chen y Shen, 2019](#) (decimoquinto antecedente), [Chai-chi y Anderson, 2019](#) (decimosexto antecedente).
- **comments:** [Y. Li y col., 2016](#) (sexto antecedente), [Kaur y Gera, 2017](#) (noveno antecedente), [S. Lee y col., 2018](#) (duodécimo antecedente), [Jin y col., 2019](#) (decimotercer antecedente).

Si bien en los respectivos antecedentes marcados en (*) figuran el promedio de los montos disponibles para patrocinar, se usó la mediana en vez de la media ya que presentó mejor performance en los experimentos.

3.4. Instrumentos de medida

Los instrumentos de medida que sirvieron para determinar la performance del o los modelos construidos en el experimento serán algunas de las métricas de clasificación de Machine Learning descritas en los papers tomados como antecedentes en el Capítulo II del presente trabajo y seleccionadas mediante Benchmarking considerando como criterio la similitud del

problema y propuestas de solución más aproximadas. Antes de proceder a explicar detalladamente cada una de ellas, es necesario conocer los conceptos de la Matriz de confusión, así como sus elementos que la componen.

- **Matriz de confusión:** Es una tabla de NxN que resume el nivel de éxito de las predicciones de un modelo de clasificación; es decir, la correlación que existe entre la etiqueta y la clasificación del modelo. Un eje de una matriz de confusión es la etiqueta que el modelo predijo; el otro es la etiqueta real. N representa el número de clases. Es un problema de clasificación binaria, N=2 ([Kohavi & Provost, 1998](#)). Su principal objetivo es describir el rendimiento de un modelo supervisado de Machine Learning en los datos de prueba, donde se desconocen los verdaderos valores. Se le llama “matriz de confusión” porque hace que sea fácil detectar dónde el sistema está confundiendo dos clases ([SitioBigData.com, 2019a](#)). Se representa de la siguiente manera:

		Valores Actuales	
		Positivos (1)	Negativos (0)
Valores Predichos	Positivos (1)	Verdaderos Positivos (VP)	Falsos Positivos (FP)
	Negativos (0)	Falsos Negativos (FN)	Verdaderos Negativos (VN)

Tabla 3.2: Matriz de confusión. Fuente: [Izco, 2018](#)

De la Tabla [3.2](#), existen 4 elementos clave:

- **Verdadero positivo** (TP o *True Positive*): Es el ejemplo en el que el modelo predijo de manera correcta la clase positiva. Por ejemplo, el modelo infirió correctamente que un paciente con determinadas características descritas en las variables sufre de cáncer ([Google Developers, 2018](#)).
- **Verdadero negativo** (TN o *True Negative*): Es el ejemplo en el que el modelo predijo de manera correcta la clase negativa. Por ejemplo, el modelo infirió correctamente que una determinada especie animal de acuerdo a sus características no era un mamífero ([Google Developers, 2018](#)).
- **Falso positivo** (FP, *False Positive* o Error del Tipo I): Es el ejemplo en el que el modelo predijo de manera incorrecta la clase positiva. Por ejemplo, el modelo infirió que un paciente varón presentaba embarazo (clase positiva) cuando en realidad no era así ([Google Developers, 2018](#)).
- **Falso negativo** (FN, *False Negative* o Error del Tipo II): Es el ejemplo en el que el modelo predijo de manera incorrecta la clase negativa. Por ejemplo, el modelo infirió

que un mensaje de correo electrónico en particular no era spam (clase negativa), pero ese mensaje en realidad sí era spam ([Google Developers, 2018](#)).

Explicado los conceptos anteriores, se derivan las siguientes métricas de clasificación usadas comúnmente, de las cuales serán usadas solo las 3 primeras tomando como referencia los papers de los antecedentes:

- **Exactitud (accuracy):** Representa la fracción de predicciones que se realizaron correctamente sobre el total de ejemplos en un modelo de clasificación. Se determina mediante la siguiente fórmula ([Kohavi & Provost, 1998](#)):

$$Exactitud = \frac{V.P. + V.N.}{V.P. + V.N. + F.P. + F.N.}$$

Ecuación 3.1: Fórmula para calcular la exactitud. Fuente: [Kohavi y Provost, 1998](#)

Esta métrica responde a la pregunta ¿Cuál es la proporción de predicciones que se realizaron correctamente? ([Izco, 2018](#))

- **Precisión (precision):** Representa el número de elementos identificados correctamente como positivo de un total de elementos identificados como positivos ([SitioBigData.com, 2019a](#)). Se calcula mediante la siguiente fórmula:

$$Precisión = \frac{V.P.}{V.P. + F.P.}$$

Ecuación 3.2: Fórmula para calcular la precisión. Fuente: [Kohavi y Provost, 1998](#)

Esta métrica responde a la pregunta ¿Qué proporción de predicciones positivas es correcta? ([Izco, 2018](#))

- **Área bajo la curva ROC (AUC):** Considera todos los umbrales de clasificación posibles. Representa la probabilidad de que un clasificador tenga más seguridad de que un ejemplo resulte ser un verdadero positivo con respecto a que sea un falso positivo ([Google Developers, 2018](#)). Para entender el concepto del área, se necesita entender qué es la curva ROC y para qué sirve en primer lugar.

La curva ROC permite cuantificar la performance de distinción entre dos cosas del modelo como, por ejemplo, si un paciente tiene cáncer o no.

Siguiendo el anterior ejemplo, se tiene un modelo que predice si un paciente sufre de cáncer o no, cuyo resultado es el siguiente (Figura 3.2)

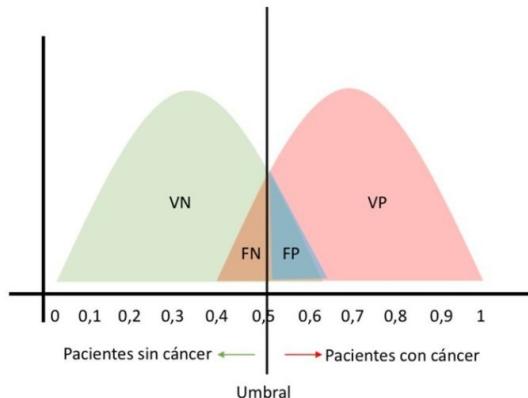


Figura 3.2: Descripción de resultados de modelo descriptivo de ejemplo. Fuente: [González, 2019](#)

En esta imagen, se puede observar que el área de borde verde (que contiene a los Falsos Positivos y el total de Negativos) representa a todos los pacientes que no tienen cáncer, mientras que el área de borde rojo (que contiene a los Falsos Negativos y el total de Positivos) representa a todos los pacientes que sí tienen cáncer. El umbral, que está establecido con valor 0.5, representa el punto de corte en el que el modelo clasificará a todos los pacientes por encima de ese valor como positivos, es decir, que sí tienen cáncer; mientras que aquellos por debajo del valor del umbral serán clasificados como negativos, es decir, que no tienen cáncer.

Cuando el umbral se desplaza hacia la izquierda, es decir, cuando la sensibilidad aumenta, la especificidad disminuirá. Por el contrario, cuando el umbral se desplaza hacia la derecha, la sensibilidad disminuirá y la especificidad aumentará. Se concluye entonces que existe una relación inversa entre la sensibilidad y la especificidad. En la curva ROC se representa la sensitividad (1-especificidad) ([González, 2019](#)).

Ahora bien, el área que se grafica bajo esta curva explicará qué tan bien funciona el modelo. Este tendrá un mejor desempeño si la curva se aleja de la diagonal principal como se observa en la Figura 3.3.

Una interpretación básica del área bajo la curva ROC respecto del poder discriminante del modelo se muestra a continuación ([Britos y col., 2006](#)):

- Si el área bajo la curva ROC = 0.5, entonces el poder discriminante del modelo es nulo.
- Si el área bajo la curva $0.5 < \text{ROC} < 0.7$, entonces el poder discriminante del modelo no es aceptable.
- Si el área bajo la curva $0.7 \leq \text{ROC} < 0.8$, entonces el poder discriminante del modelo es aceptable.

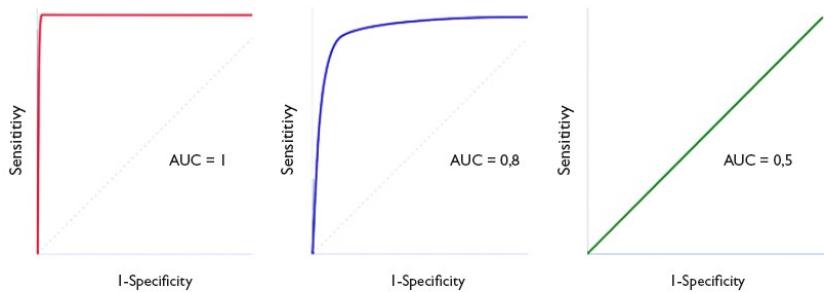


Figura 3.3: Comparación de tres resultados de la curva AUC en el modelo. Fuente: [Molina Arias y Ochoa Sangrador, 2017](#)

- Si el área bajo la curva $0.8 \leq \text{ROC} < 0.9$, entonces el poder discriminante del modelo es excelente.
- Si el área bajo la curva $\text{ROC} \geq 0.9$, entonces el poder discriminante del modelo es excepcionalmente bueno.
- **Sensibilidad (recall, sensitivity o True Positive Rate):** Representa el número de elementos correctamente identificados como positivos del total de positivos verdaderos ([SitioBigData.com, 2019a](#)). Se calcula mediante la siguiente fórmula:

$$\text{Sensibilidad} = \frac{\text{V.P.}}{\text{V.P.} + \text{F.N.}}$$

Ecuación 3.3: Fórmula para calcular la sensibilidad. Fuente: [Kohavi y Provost, 1998](#)

- **Puntaje F1 (F1-Score):** Representa la media armónica de la precisión y la sensibilidad. Normalmente, se usa cuando uno difiere mucho del otro y no es posible realizar una conclusión determinante ya que solo es posible predecir bien una clase ([SitioBigData.com, 2019a](#)). Se calcula mediante la siguiente fórmula:

$$\text{Puntaje F1} = \frac{2 * \text{Precisión} * \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}}$$

Ecuación 3.4: Fórmula para calcular el puntaje F1. Fuente: [Kohavi y Provost, 1998](#)

3.5. Técnicas de recolección de datos

Los conjuntos de datos recolectados para la investigación se componen de mezcla de observaciones cuantitativas (variables numéricas basadas en características del proyecto como la meta de financiamiento, montos prometidos, duración de la campaña y otros indicadores financieros) y cualitativas (descripción y comentarios del proyecto). Para obtener los 3 principales datasets, se siguió el flujo de la Figura 3.4. La base de datos de la metainformación, que comprende las variables cuantitativas y propaganda del proyecto, fue consolidada luego de descargar un histórico público de 10 años de la página web “Web Robots”, fundada por los ex corporativos de TI Tomás Vitulskis y Paulius Jonaitis, y posteriormente pre-procesarla. Mientras que por el lado de la descripción y comentarios de cada proyecto, se usaron técnicas y herramientas de *web scraping* a partir de los URLs. El detalle de cada paso del proceso se explica en la sección de Construcción de los conjuntos finales de datos del Capítulo IV.

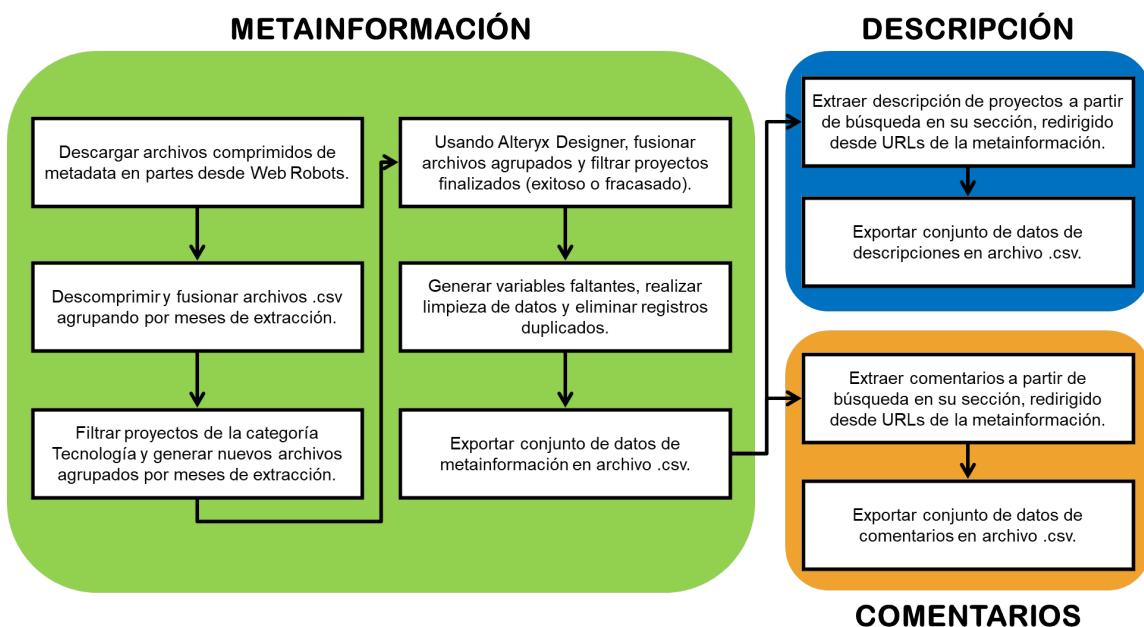


Figura 3.4: Flujograma de la recolección de conjuntos finales de datos. Fuente: Elaboración propia.

Asimismo, para encontrar algunos de los papers con la información requerida más cercana, se utilizaron keywords o palabras clave como *crowdfunding*, *Machine Learning*, *Deep Learning*, *prediction*, *Kickstarter*, *accuracy* y *projects*.

3.6. Técnicas para el procesamiento y análisis de la información

Como se explicó en el punto 2.2.6 del Marco Teórico del presente trabajo, dentro de los sistemas de analítica de negocio, Big Data y Minería de Datos, tres de las metodologías más usadas son CRISP-DM, SEMMA y KDD ([Braulio Gil & Curto Díaz, 2015](#)).

Después de evaluar y comparar las tres mencionadas, se concluyó que la mejor de ellas dependerá de la intención o finalidad que busca lograr el o los involucrados en el negocio tanto en objetivos como en resultados. Para la presente investigación, se siguió la metodología de CRISP-DM debido a la coyuntura en la que se desarrolló. Las razones de la elección de la misma fueron las siguientes:

- La metodología CRISP-DM contempla entre sus fases la comprensión del negocio además de la parte técnica que incluye el modelado y análisis de resultados. La comprensión del negocio tiene un rol importante ya que se define al inicio de todo el proceso para dar el alcance del proyecto y definir objetivos que se buscan a partir de la Minería de Datos y Big Data.
- Ayuda a la gerencia del proyecto en la planeación y toma de decisiones (fase Despliegue) a partir de los resultados obtenidos, reportándolos y convirtiéndolos en oportunidades a considerar en los objetivos del negocio.
- Evalúa en todo el proceso los datos y variables usadas con el fin de crear el mejor modelo. Las variables seleccionadas serán importantes para interpretar los resultados y tomar decisiones.
- Si se habla de las otras dos metodologías, empezando por KDD, si bien esta contempla 9 pasos durante su proceso, el objetivo de KDD en cada uno de estos resulta ser más técnico, es decir, trabajar, seleccionar e interpretar métricas, variables, modelos, entre otros para obtener los mejores resultados más allá de considerar el contexto y comprensión del negocio. De hecho, no existe alguna fase dedicada al entendimiento del mismo.
- Por otra parte, la metodología SEMMA se basa, como su nombre lo indica, en la selección, exploración y modelado de grandes cantidades de datos para descubrir patrones de negocio desconocidos. Sin embargo, al limitarse a 5 fases comenzando con la fase de muestreo, no hace hincapié en la comprensión del negocio, sino más bien comienza con el procesamiento de datos para la construcción del modelo.

Cada una de las fases de la metodología seleccionada se detalla a continuación.

3.6.1. Comprensión del negocio

La primera fase de la metodología CRISP-DM consiste en la definición del problema de Minería de Datos que se tiene, así como el entendimiento de los objetivos y requerimientos que se espera lograr en el proyecto.

Para empezar, el crowdfunding se basa, como su nombre en inglés lo indica, en lograr que un proyecto emprendedor sea llevado a cabo gracias al financiamiento colectivo. Existen 4 diferentes modelos de crowdfunding: basado en donaciones, basado en recompensas, basado en capital social y basado en deuda. El crowdfunding basado en capital social se encuentra actualmente limitado en los Estados Unidos debido a que la Regulación D de la Ley de Valores de 1933 prohíbe la participación de muchos potenciales inversionistas y los obliga a tener un ingreso anual mayor a \$200,000 o más de \$1 millón en patrimonio neto ([Lichtig, 2015](#)).

El modelo basado en donaciones se refiere a la recaudación de fondos a través de la Web 2.0, en el cual los patrocinadores no esperan recompensas materiales a cambio sino más bien una recompensa social. Por el contrario, el modelo basado en recompensas ofrece compensación tanto material como inmaterial y representa hoy en día el modelo de crowdfunding más frecuente. Los financiadores, por un lado, pueden verse beneficiados de la venta anticipada, recibiendo el proyecto o producto financiado antes de su publicación o entrada al mercado al mejor precio. Los proyectos que pertenecen a esta categoría a menudo son organizaciones sin fines de lucro ([Kraus y col., 2016](#)).

Kickstarter, la plataforma de crowdfunding más citada, analizada y una de las más grandes, es una comunidad basada en el crowdfunding por recompensas. A la fecha, 128 mil proyectos han sido financiados, 3 billones de dólares fueron prometidos en proyectos y 13 millones de patrocinadores han participado, del cual el 31 % ha colocado dinero en más de 1 proyecto.

La particularidad de los proyectos en esta plataforma es que, luego de cumplir un plazo de tiempo determinado, el monto prometido acumulado puede ser asignado a los proyectos que hayan alcanzado o superado la meta estipulada al inicio. Para ello, los dueños de los proyectos tienen que crear estrategias para lograr campañas exitosas y hacer realidad su producto. Un 75 % de los proyectos que fueron apoyados por al menos 25 patrocinadores han sido financiados exitosamente ([Kickstarter, s.f.-c](#)).

Para lograr que un proyecto en la plataforma logre impactar en la comunidad, debe contar con conexiones duraderas, retroalimentación del producto y ser tangible. Asimismo, algunos enfoques importantes se basan en probar nuevas ideas, aumentar la comunidad y cubrir costos de manufacturación.

Para empezar a realizar la campaña del proyecto, es necesario seguir las sugerencias basadas en los autores [A. Yu, 2017](#) y [Kickstarter, s.f.-b:](#)

- Identificar la meta óptima del proyecto. Se debe realizar una lista de todos los costos, así como el shipping que deben de cada país. También se debe contar con un buffer en caso ocurra un imprevisto. Se puede determinar la meta a través de la fórmula (Gastos + Buffer) x Alcance.
- Crear el video de la campaña en un presupuesto. El vídeo resulta ser un factor muy importante ya que aquellos que cuentan con un video de su proyecto tienen un 50% de chance de tener éxito. El promedio debería ser de 3.38 minutos, aunque el público presta más atención hacia aquellos que duran poco. El vídeo debe contener y responder las siguientes preguntas:
 - ¿Qué hace memorable al proyecto?
 - ¿Cómo se verá el proyecto?
 - ¿Por qué se está creando el producto?
 - ¿Por qué lo necesitas?
 - ¿Por qué es importante que tu proyecto exista?
 - ¿Por qué debes ser tú su creador?
 - ¿Por qué tu compañía?
 - ¿Cómo proveerás una solución?
 - ¿Cómo funciona la solución?

Es importante destacar las estadísticas del problema que se piensa resolver al inicio del video ya que el primer minuto resulta ser el lapso de tiempo que más capta la atención del público. Se recomienda también tener buen audio e iluminación.

- Escribir la descripción del proyecto. Otro factor clave en el desempeño de la campaña de un proyecto en Kickstarter resulta ser la descripción que representa un resumen de los puntos más relevantes del producto. Se debe mencionar una breve introducción del proyecto, cómo funciona, para quiénes son, el proceso que se siguió en el desarrollo, detalles y especificaciones, y un cronograma e hitos.
- Incluir imágenes sobre cómo lucirá el proyecto. Esta prueba convencerá a más de una persona en querer invertir en el proyecto. Las imágenes a veces atraen más a las personas que no gustan leer la descripción completa del proyecto.

- Determinar las recompensas del proyecto. Al basarse en un modelo de crowdfunding en recompensas, los patrocinadores esperan un beneficio por parte del proyecto una vez este logre alcanzar su financiamiento. El número ideal de recompensas puede ser entre 5 y 7. Una recompensa en promedio de \$100 es la que genera más dinero. Las recompensas pueden ser de cuatro tipos: el producto en sí mismo entregado hacia los patrocinadores antes que este salga al mercado, reconocimientos en la página web y acceso hacia actualizaciones, recuerdos y certificados de membresía, y nuevas experiencias como visita a los desarrolladores en sus estudios.

Todos los factores anteriormente mencionados determinarán en gran medida el nivel de éxito que tendrá un proyecto una vez su campaña sea lanzada en la plataforma.

3.6.2. Comprensión de los datos

Una vez comprendido el funcionamiento del negocio y sus objetivos que busca lograr, la siguiente fase fue entender el significado de los datos con los que se trabajaron. Estos incluyen valores y variables que determinaron la performance del modelo elaborado en las siguientes etapas.

Para ello, se analizaron estadísticamente las variables numéricas del dataset generado de la Metainformación (Tabla 4.1) luego del proceso de construcción bajo el flujo de la Figura 3.4 y que se explica en el subcapítulo 1 del Capítulo IV.

En el siguiente subcapítulo del mismo capítulo, se detalla el análisis exploratorio de cada variable, entre los cuales destaca el desbalance en las etiquetas de la variable dependiente a predecir (*state*).

3.6.3. Preparación de los datos

El análisis exploratorio de los datos fue determinante para la selección final de variables de la Metainformación, gracias al uso de herramientas estadísticas como el análisis de correlaciones para discriminar aquellas que superaron el límite fijado y que desembocó, luego del pre-procesamiento detallado en el subcapítulo 3 del mismo capítulo, en la Tabla 3.1.

3.6.4. Modelamiento

Luego del pre-procesamiento de los datos, en esta fase se procedió a crear modelos predictivos para cada modalidad considerada. Tomando de referencia a los antecedentes del Marco Teórico, los más usados para cada modalidad se detalla a continuación:

- **Metainformación:** Perceptrón Multicapa ([R. S. Kamath y Kamat, 2018](#), décimo antecedente; [P.-F. Yu y col., 2018](#), undécimo antecedente; [Cheng y col., 2019](#), decimocuarto antecedente), Máquina de Vectores de Soporte ([K. Chen y col., 2013](#), primer antecedente; [Beckwith, 2016](#), quinto antecedente; [Sawhney y col., 2016](#), octavo antecedente), Regresión Logística ([Mitra y Gilbert, 2014](#), segundo antecedente; [M. Zhou y col., 2015](#), tercer antecedente; [Beckwith, 2016](#), quinto antecedente; [Y. Li y col., 2016](#), sexto antecedente; [Kaur y Gera, 2017](#), noveno antecedente), Regresión Log-logística ([Y. Li y col., 2016](#), sexto antecedente), Bosques Aleatorios ([S.-Y. Chen y col., 2015](#), cuarto antecedente; [H. Yuan y col., 2016](#), séptimo antecedente; [R. S. Kamath y Kamat, 2018](#), décimo antecedente), Árboles de Decisión ([Beckwith, 2016](#), quinto antecedente; [R. S. Kamath y Kamat, 2018](#), décimo antecedente), Naïve Bayes ([Beckwith, 2016](#), quinto antecedente; [R. S. Kamath y Kamat, 2018](#), décimo antecedente), Modelo Seq2seq ([Jin y col., 2019](#), decimotercer antecedente).
- **Descripción:** CNN ([Cheng y col., 2019](#), decimocuarto antecedente), LSTM ([Jin y col., 2019](#), decimotercer antecedente), Modelo Seq2seq ([S. Lee y col., 2018](#), duodécimo antecedente), Máquina de Vectores de Soporte o variantes ([Sawhney y col., 2016](#), octavo antecedente; [L.-S. Chen y Shen, 2019](#), decimoquinto antecedente), Regresión Logística ([Mitra y Gilbert, 2014](#), segundo antecedente; [M. Zhou y col., 2015](#), tercer antecedente), LDA o variantes ([H. Yuan y col., 2016](#), séptimo antecedente; [Sawhney y col., 2016](#), octavo antecedente).
- **Comentarios:** LSTM ([Jin y col., 2019](#), decimotercer antecedente; [Shafqat y Byun, 2019](#), decimoséptimo antecedente), LDA o variantes ([Shafqat y Byun, 2019](#), decimoséptimo antecedente), Modelo Seq2seq ([S. Lee y col., 2018](#), duodécimo antecedente; [Jin y col., 2019](#), decimotercer antecedente), HAN ([S. Lee y col., 2018](#), duodécimo antecedente), Regresión Logística ([Y. Li y col., 2016](#), sexto antecedente; [Kaur y Gera, 2017](#), noveno antecedente), Regresión Log-logística ([Y. Li y col., 2016](#), sexto antecedente).

De todos los modelos mencionados, los más predominantes de Aprendizaje Automático fueron las Máquinas de Vectores de Soporte y los de Regresión Logística, mientras que por el lado de Aprendizaje Profundo, fueron los Perceptrones Multicapa aunque las otras redes enunciadas derivan de este tipo.

Según los anteriores autores, salvo en investigaciones donde se usaron 1 sola modalidad, como por ejemplo la Metainformación, los modelos de Aprendizaje Profundo tuvieron mejor desempeño que los tradicionales de Aprendizaje Automático. Además, dado que la propuesta de esta investigación es ensamblar múltiples modelos, considerar un mix de ambos tipos de arquitectura resulta inviable por la complejidad de transformación de sus salidas de distintos tamaños.

Por ello, se optó por usar arquitecturas de Redes Neuronales para cada modalidad con el fin de poder ensamblar y comparar sus resultados, tanto de manera individual como en el modelo apilado, con la literatura.

3.6.5. Evaluación

De las métricas de clasificación de Aprendizaje Automático explicadas en el cuarto subcapítulo del presente capítulo, las usadas por los autores de la literatura fueron las siguientes:

- **Exactitud:** [K. Chen y col., 2013](#) (primer antecedente); [S.-Y. Chen y col., 2015](#) (cuarto antecedente); [Beckwith, 2016](#) (quinto antecedente); [H. Yuan y col., 2016](#) (séptimo antecedente); [Sawhney y col., 2016](#) (octavo antecedente); [Kaur y Gera, 2017](#) (noveno antecedente); [R. S. Kamath y Kamat, 2018](#) (décimo antecedente); [P.-F. Yu y col., 2018](#) (undécimo antecedente); [S. Lee y col., 2018](#) (duodécimo antecedente); [Cheng y col., 2019](#) (decimocuarto antecedente); [L.-S. Chen y Shen, 2019](#) (decimoquinto antecedente); [Shafqat y Byun, 2019](#) (decimoséptimo antecedente).
- **Precisión:** [Beckwith, 2016](#) (quinto antecedente); [H. Yuan y col., 2016](#) (séptimo antecedente); [Kaur y Gera, 2017](#) (noveno antecedente); [Cheng y col., 2019](#) (decimocuarto antecedente).
- **Sensibilidad:** [Beckwith, 2016](#) (quinto antecedente); [H. Yuan y col., 2016](#) (séptimo antecedente); [Kaur y Gera, 2017](#) (noveno antecedente); [Cheng y col., 2019](#) (decimocuarto antecedente); [L.-S. Chen y Shen, 2019](#) (decimoquinto antecedente).
- **Especificidad:** [L.-S. Chen y Shen, 2019](#) (decimoquinto antecedente).
- **Ratio de Falsa Alarma:** [Kaur y Gera, 2017](#) (noveno antecedente).
- **Media Geométrica (G-Mean):** [L.-S. Chen y Shen, 2019](#) (decimoquinto antecedente).
- **Puntaje F1:** [M. Zhou y col., 2015](#) (tercer antecedente); [Beckwith, 2016](#) (quinto antecedente); [H. Yuan y col., 2016](#) (séptimo antecedente); [Kaur y Gera, 2017](#) (noveno antecedente).

tecedente); [Cheng y col., 2019](#) (decimocuarto antecedente); [L.-S. Chen y Shen, 2019](#) (decimoquinto antecedente).

- **Curva ROC:** [M. Zhou y col., 2015](#) (tercer antecedente); [Beckwith, 2016](#) (quinto antecedente).
- **Área bajo la Curva ROC (AUC):** [Beckwith, 2016](#) (quinto antecedente); [Y. Li y col., 2016](#) (sexto antecedente); [Kaur y Gera, 2017](#) (noveno antecedente); [P.-F. Yu y col., 2018](#) (undécimo antecedente); [Cheng y col., 2019](#) (decimocuarto antecedente).
- **Área bajo la Curva Precisión-Sensibilidad (PRC):** [Kaur y Gera, 2017](#) (noveno antecedente).
- **Error de Validación Cruzada:** [Mitra y Gilbert, 2014](#) (segundo antecedente).
- **Coeficiente de Correlación de Matthew (MCC):** [Kaur y Gera, 2017](#) (noveno antecedente).
- **Divergencia de Kullback-Leibler (KL):** [Jin y col., 2019](#) (decimotercer antecedente).
- **Raíz del Error Cuadrático Medio (RMSE):** [Jin y col., 2019](#) (decimotercer antecedente).
- **Error Absoluto Medio (MAE):** [Jin y col., 2019](#) (decimotercer antecedente).
- **Índice de Concordancia (CI):** [Jin y col., 2019](#) (decimotercer antecedente).

En el Capítulo 5 se explica cuáles fueron seleccionadas para evaluar cada modelo.

3.6.6. Despliegue

Luego de analizar el desempeño de cada modelo, tanto individual como el modelo final, se compararon los resultados con los antecedentes y se desarrolló una prueba piloto en la cual el modelo apilado entrenado fue ejecutado usando como entrada la URL de un proyecto aleatorio de Kickstarter para, luego de obtener sus características, predecir su estado de financiamiento. Esta acción también se encuentra detallada en el Capítulo 5.

3.7. Cronograma de actividades y presupuesto

Se elaboró un cronograma de actividades de toda la investigación, mostrada en la Figura 3.5, contemplando desde el inicio de la misma, desarrollo, evaluación de resultados y sustentación.

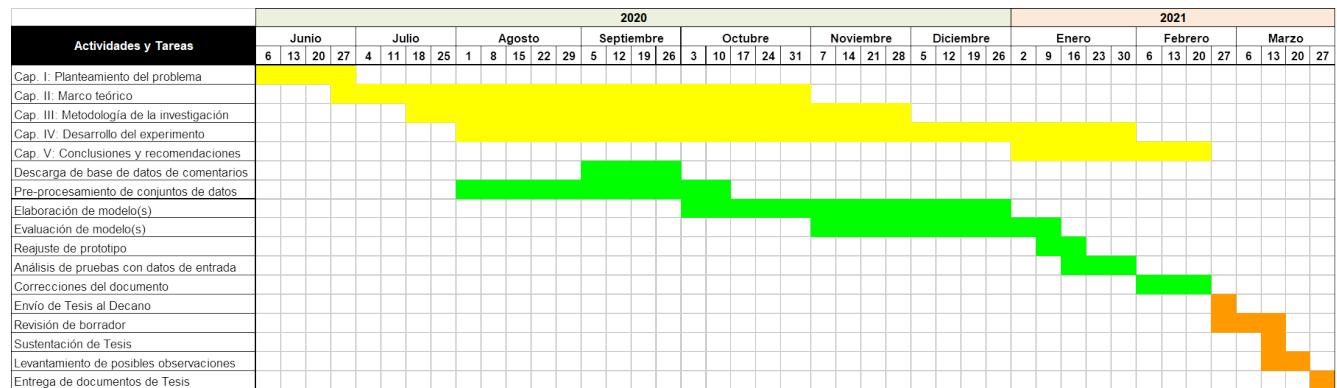


Figura 3.5: Cronograma de actividades de la investigación. Fuente: Elaboración propia

La partida presupuestal de todo el proyecto se divide en dos partes: los costos personales del autor y los costos de las herramientas para la operación del proyecto.

Los costos personales del autor del trabajo de tesis se muestran en la Tabla 3.3. Estos incluyen las herramientas adquiridas antes del inicio de la investigación como la laptop, así como los pagos de servicios generales y del trámite de elaboración y sustentación pública de Tesis. Se menciona la cantidad de horas utilizadas por el autor para el desarrollo del trabajo; sin embargo no se considera un valor monetario por cada unidad al ser un valor incalculable.

Asimismo, también se contemplan los costos por uso de servicios en la nube como parte de la extracción de comentarios desde servidores en Google Cloud Platform (GCP) y desarrollo de modelos en Google Colab Pro en la Tabla 3.4.

Los montos totales por el uso de instancias creadas en GCP son descontados de los \$300.00 en crédito gratuito válido por 12 meses (a la fecha en que se desarrolló la investigación) (Google Cloud, s.f.), recibidos inicialmente desde el 12 de agosto del 2020 (fecha de inscripción) para poder ser usados como versión de prueba. A partir del siguiente mes, se mejoró el plan para habilitar más servicios (por ejemplo, máquina virtual en Windows) y comenzó a facturarse \$5.22 mensualmente.

Item	Tiempo usado (horas)	Costo (soles)	Subtotal
Recursos materiales			
Laptop Lenovo ideapad 330 Core i7 8va Gen		S/.4,500.00	S/.4,500.00
Pagos del trámite de elaboración y sustentación pública de Tesis			
Derecho de inscripción de tema de investigación		S/.800.00	S/.800.00
Reserva del tema de tesis		S/.2,700.00	S/.2,700.00
Derecho de sustentación		S/.1,500.00	S/.1,500.00
Recursos humanos			
Avance de tesis	900	Incalculable	-
Servicios generales			
Internet + luz (7 meses)	110	S/.80.00	S/.560.00
Total			S/.10,060.00

Tabla 3.3: Presupuesto de los costos personales del autor. Fuente: Elaboración propia.

Item	Unidades	Costo (dólares)	Horas	Subtotal
Google Cloud Platform				
Instancia VM Ubuntu (g1-small, 12GB RAM)	1	\$0.021	310	\$6.51
Imagen de instancia Ubuntu	8	\$0.02	306	\$48.96
Instancia VM de Windows Server 2019 (4GB RAM)	1	\$0.086	300	\$25.80
Costo por instancias encendidas	10			\$3.41
Uso de instancias posteriormente eliminadas				\$95.88
Pago mensual (luego de aceptar mejora de plan)	4	\$5.22		\$20.88
Crédito de \$300.00 por 12 meses	1	-\$300.00		-\$300.00
Google Colab Pro				
Pago mensual (luego de aceptar mejora de plan)	5	\$9.99		\$49.95
Total a pagar				\$49.95

Tabla 3.4: Presupuesto de los costos de las herramientas para el proyecto. Fuente: Elaboración propia.

Capítulo 4

DESARROLLO DEL EXPERIMENTO

4.1. Construcción de los conjuntos finales de datos

El conjunto total de datos utilizado para la investigación consistió en la recolección de 27,251 proyectos tecnológicos de Kickstarter comprendidos entre los años 2009 y 2019, en 3 partes: metainformación, descripción y comentarios (excluyendo los del creador) del proyectos.

4.1.1. Metainformación

Como se mencionó en las Técnicas de recolección de datos del Capítulo III, el punto de partida para la construcción de los conjuntos de datos fue consolidar las bases públicas, archivos de valores separados por comas (.csv) comprimidos, de la página Web Robots (<https://webrbots.io/kickstarter-datasets/>), fraccionadas por fechas mensuales de captura de información comenzadas desde noviembre del 2015 hasta agosto del 2019 (Figura 4.1), y posteriormente pre-procesada con la ayuda del software Alteryx Designer. De acuerdo con la información de los creadores del sitio Web Robots, se ejecutan robots en dos servidores en la nube encargados de recolectar en un determinado punto del día y una vez al mes información de las campañas que aparecen en Kickstarter ([Web Robots, 2019](#)).

Cada archivo descargado por fecha de captura mensual es descomprimido, y las particiones en formato .csv que contienen son juntadas mediante el siguiente proceso:

- Descargar librerías correspondientes de Python (pandas, numpy, glob, math).
- Dirigirse a la ruta de destino.
- Crear carpetas por mes para almacenar archivos particionados.

Kickstarter Datasets

We have a scraper robot which crawls all [Kickstarter](#) projects and collects data in CSV and JSON formats. From March 2016 we run this data crawl once a month. Datasets are available from the following scrape dates:

2019

- 2019-08-15 [\[JSON\]](#) – [\[CSV\]](#)
- 2019-07-18 [\[JSON\]](#) – [\[CSV\]](#)
- 2019-06-13 [\[JSON\]](#) – [\[CSV\]](#)
- 2019-05-16 [\[JSON\]](#) – [\[CSV\]](#)
- 2019-04-18 [\[JSON\]](#) – [\[CSV\]](#)
- 2019-03-14 [\[JSON\]](#) – [\[CSV\]](#)
- 2019-02-14 [\[JSON\]](#) – [\[CSV\]](#)
- 2019-01-17 [\[JSON\]](#) – [\[CSV\]](#)

2018

- 2018-12-13 [\[JSON\]](#) – [\[CSV\]](#)
- 2018-11-15 [\[JSON\]](#) – [\[CSV\]](#)
- 2018-10-18 [\[JSON\]](#) – [\[CSV\]](#)
- 2018-09-13 [\[JSON\]](#) – [\[CSV\]](#)
- 2018-08-16 [\[JSON\]](#) – [\[CSV\]](#)
- 2018-07-12 [\[JSON\]](#) – [\[CSV\]](#)
- 2018-06-14 [\[JSON\]](#) – [\[CSV\]](#)
- 2018-05-17 [\[JSON\]](#) – [\[CSV\]](#)
- 2018-04-12 [\[JSON\]](#) – [\[CSV\]](#)
- 2018-03-15 [\[JSON\]](#) – [\[CSV\]](#)
- 2018-02-15 [\[JSON\]](#) – [\[CSV\]](#)
- 2018-01-12 [\[JSON\]](#) – [\[CSV\]](#)

Figura 4.1: Vista (agosto 2019) del website Web Robots. Fuente: [Web Robots, 2019](#)

- Ejecutar algoritmo para unir archivos particionados csv dentro de una misma carpeta.

Cuando el algoritmo finaliza su ejecución, se crea un nuevo archivo separado por comas en cada carpeta por mes de extracción, cuyo tamaño oscila entre 1 y 5 gigabytes (GB) en total por cada uno. Con el fin de ahorrar espacio en memoria dentro de la computadora, los archivos particionados son eliminados y solamente se almacenan los nuevos generados.

Por ejemplo, en la Figura 4.2 se detalla el tamaño del conjunto de datos total al corte del periodo de captura de información Julio 2019, aproximadamente más de 212 mil proyectos de todas las categorías y 37 columnas de variables.

```
In [7]: data_combinada_201907.shape    ##Originalmente había 212,378 registros de todos los proyectos en Kickstarter
Out[7]: (212378, 37)

In [8]: data_combinada_201907.columns
Out[8]: Index(['backers_count', 'blurb', 'category', 'converted_pledged_amount',
       'country', 'created_at', 'creator', 'currency', 'currency_symbol',
       'currency_trailing_code', 'current_currency', 'deadline',
       'disable_communication', 'friends', 'fx_rate', 'goal', 'id',
       'is_backing', 'is_starrable', 'is_starred', 'launched_at', 'location',
       'name', 'permissions', 'photo', 'pledged', 'profile', 'slug',
       'source_url', 'spotlight', 'staff_pick', 'state', 'state_changed_at',
       'static_usd_rate', 'urls', 'usd_pledged', 'usd_type'],
      dtype='object')
```

Figura 4.2: Tamaño de conjunto de datos al corte de Julio 2019. Fuente: Elaboración propia.

A cada conjunto de datos generado se filtran los que pertenecen a la categoría **Technology**. Debido a que no se cuenta con la información de la columna *main_category*, este proceso de filtración se logra utilizando la variable *source_url* al seleccionar aquellos registros que con-

tengan la cadena de caracteres “<https://www.kickstarter.com/discover/categories/technology>” . En la Figura 4.3 se aprecia cómo queda el conjunto filtrado por categoría Techonlogy al corte de Julio 2019.

```
In [40]: df_201907_filtrada.shape  
Out[40]: (21398, 37)
```

Figura 4.3: Tamaño de conjunto de datos filtrado del corte de Julio 2019. Fuente: Elaboración propia.

Cuando se repitió este procedimiento con cada conjunto de datos, se notó que la proporción de proyectos tecnológicos en Kickstarter representa el 10 % del total de proyectos. Esto se calculó al comparar los tamaños (cantidad de registros por fila) del conjunto de datos inicial y del filtrado.

La siguiente fase del pre-procesamiento fue la de seleccionar las variables de los nuevos conjuntos de datos filtrados que se usarán para generar el dataset final. Para ello, se usó el software Alteryx Designer, el cual permite desarrollar flujos de trabajo para preparar, unir y analizar volúmenes de datos complejos de distintas fuentes.

Con los conjuntos de datos filtrados, se creó el flujo de trabajo representado en la Figura 4.4.

El flujo comienza con la carga de los datos de entrada, los cuales son los 45 archivos separados por coma por cada captura mensual desde noviembre del 2015 hasta agosto del 2019. Luego, se realizaron dos uniones en vez de una sola, esto debido a que, a partir de marzo del 2018, algunas de las variables y valores de estas son diferentes a la de sus predecesoras. Sin embargo, esto no afectará al proceso más adelante ya que ambas uniones fueron juntadas por las mismas variables.

En cada una de las dos uniones, se seleccionaron las variables de la Tabla N° 3, se realizó limpieza de datos para las variables *category*, *location*, *photo* y *urls*, y se transformaron las variables numéricas en milisegundos *created_at*, *launched_at* y *deadline* a variables de fecha. Esto último permitió crear la variable *duration* para determinar la duración de la campaña (en días) de un proyecto calculando la diferencia entre la fecha de culminación (*deadline*) y la fecha de lanzamiento (*launched_at*). Luego se excluyeron los proyectos en proceso de la variable *state* para conservar los culminados, es decir, aquellos cuyo valor sea “successful” o “failed” ya que se analizarán solamente los proyectos que han sido exitosos o fracasados. Por último, la variable *urls* contiene los enlaces directos de los proyectos.

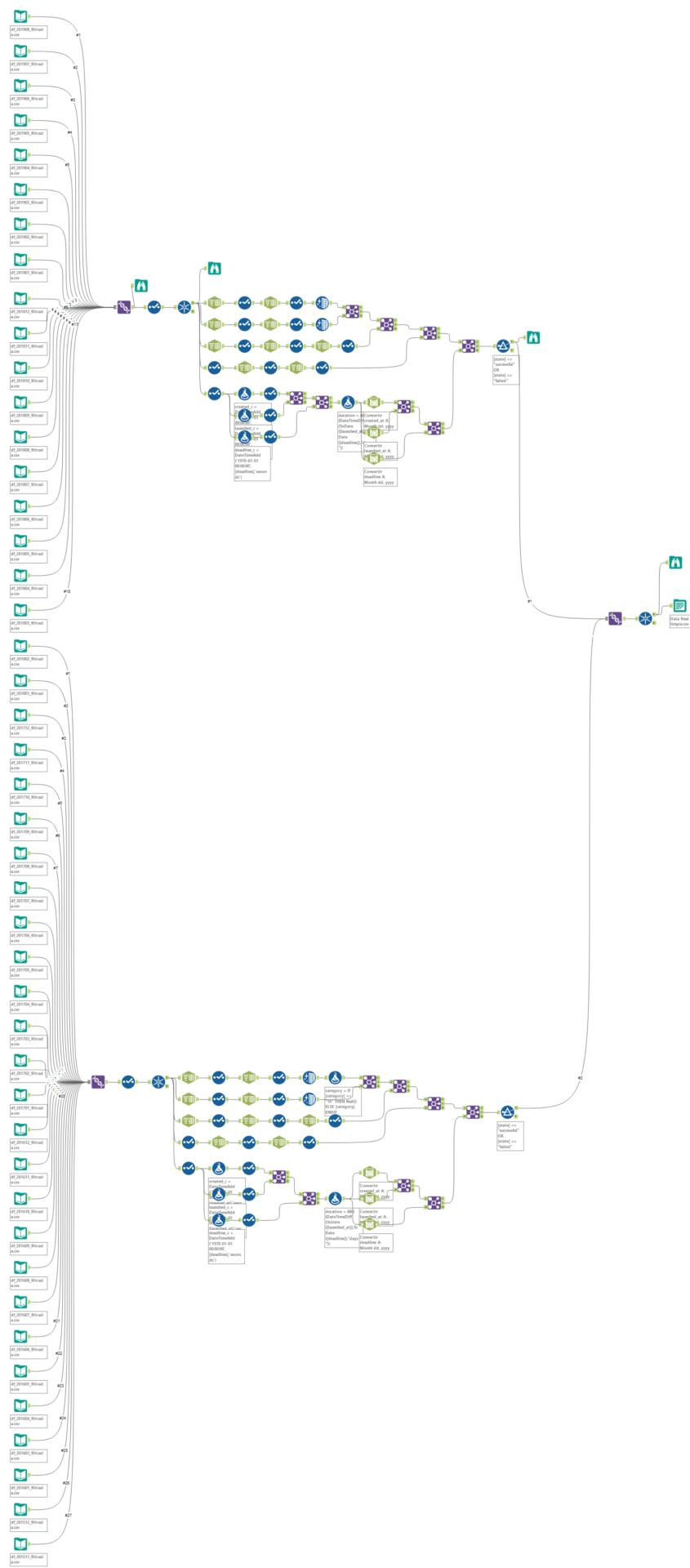


Figura 4.4: Flujo de trabajo de la data final en Alteryx Designer. Fuente: Elaboración propia.

El flujograma culmina con la generación de un archivo de valores separados por coma (.csv) guardado en memoria local. Posteriormente, el archivo generado fue subido a la plataforma Kaggle de manera pública (Figura 4.5) para que pueda ser descargada.

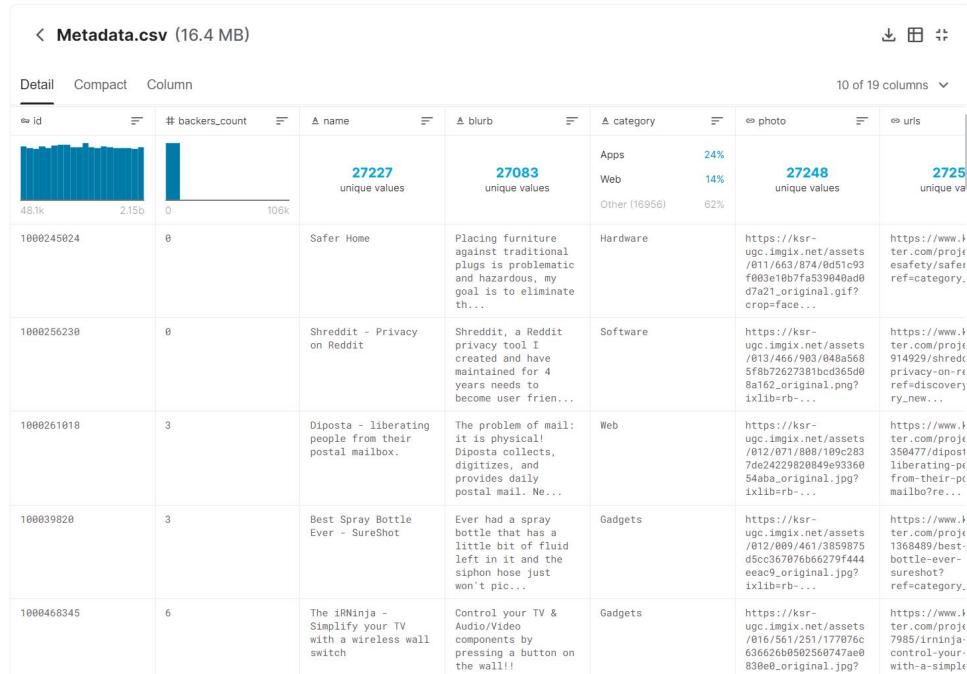


Figura 4.5: Visualización del archivo de metainformación subido a Kaggle. Fuente: Elaboración propia.

La Tabla 4.1 detalla cada variable del conjunto final, adicionando la variable *pledge_amounts* obtenida a partir de web scraping.

4.1.2. Descripción

La descripción (variable *description*) contiene todas las características del producto o servicio del proyecto ofrecido al público, así como otros datos importantes de la campaña. Al proveer gran cantidad de información, en los antecedentes de la investigación se confirma la relación que presenta esta observación con la performance de la campaña y, por ende, el resultado final del financiamiento del proyecto.

La variable se obtuvo utilizando web scraping en cada proyecto gracias a la variable *urls*. Para ello, y como se describe en la Figura 4.6, se elaboró un algoritmo usando la librería BeautifulSoup que, mediante la navegación al contenido de estas páginas a través de un agente falso, se dirigió a las descripciones de los proyectos identificando las etiquetas con clase llamada “**rte_content js-full-description responsive-media**” y las almacenó en un vector vacío,

Variable	Detalle	Tipo de dato
id	Identificador del proyecto.	number
backers_count	Número de patrocinadores de la campaña del proyecto.	number
name	Nombre del proyecto.	string
blurb	Propaganda del proyecto.	string
category	Categoría (dentro de categoría principal) del proyecto.	string
photo	Dirección de enlace de la foto del proyecto.	string
urls	Dirección de la página de la campaña del proyecto.	string
city	Ciudad del creador del proyecto.	string
country	Código de país del creador del proyecto.	string
goal	Monto de la meta de financiamiento del proyecto.	float
pledge_amounts	Montos disponibles para patrocinar la campaña.	string
pledged	Monto final patrocinado de la campaña.	float
currency	Divisa del monto final patrocinado.	string
usd_pledged	Monto final patrocinado de la campaña (en USD).	float
created_at	Fecha de creación de la campaña.	date
launched_at	Fecha de lanzamiento de la campaña.	date
deadline	Fecha de culminación de la campaña.	date
duration	Duración de la campaña (en días).	number
state	Estado de financiamiento del proyecto.	string

Tabla 4.1: Diccionario de datos del dataset de metainformación generado. Fuente: Elaboración propia.

uniendo previamente los párrafos y eliminando caracteres especiales, para posteriormente asignarle la id correspondiente y guardarlo en un archivo de extensión .csv. En caso el algoritmo no encuentre esta clase dentro de las páginas (IndexError), el vector almacenaba un registro nulo.

Debido a la gran cantidad de memoria y tiempo que iba a presentar este proceso, se determinó fraccionar los 27,251 proyectos en tres partes y repetir el mismo en cada uno de ellos. El tiempo aproximado de descarga de cada fracción fue de 6 horas.

Finalmente, las tres partes fueron unidas, se reemplazaron los valores nulos por espacios en blanco y se guardó como un nuevo archivo de valores separados por coma (.csv) en código Unicode UTF-8 para la lectura de caracteres no alfabéticos.

El archivo generado fue subido a la plataforma Kaggle de manera pública (Figura 4.7) para que pueda ser descargada a través del API de la web.

```

def getPageText(url):
    # Crear agente falso para scrapear
    ua = UserAgent()
    user_agent = ua.chrome
    # Solicitar uso de librería urllib con agente falso
    request = urllib.request.Request(url, headers = {"User-Agent":user_agent})
    # Obtener contenido de urls mediante librería urllib
    data = urllib.request.urlopen(request)
    # parse as html structured document
    bs = BeautifulSoup(data, "html.parser")
    # Buscar todos los tags div con una determinada clase
    # A partir de acá, se usa un "try" y un "except" porque hay páginas que no muestran su contenido
    # Para evitar que salga el mensaje de error, se llenarán como null los contenidos que no se puedan descargar
    # El try contiene el algoritmo para descargar la descripción de un proyecto
    try:
        description = bs.find_all("div", {"class":"rte__content js-full-description responsive-media"})
        # Encontrar todos los párrafos
        description = description[0].find_all("p")
        # Crear array vacío donde se almacenará el contenido descargado
        project_description = []
        # Iteración para agregando en lista cada descripción descargada
        for link in description:
            project_description.append(link.text)
        # Juntar párrafos separados en un solo vector, separándolos con un espacio
        project_description = ' '.join(project_description)
        # Eliminar caracteres especiales
        project_description = project_description.replace(u'\xa0', u' ')
        project_description = project_description.replace(u'\n', u' ')
    # Y el except sirve para llenar valores nulos en el array en caso de error
    except (IndexError, ValueError):
        project_description = 'null'
    # Eliminar saltos de línea
    return Newlines.sub("\n", project_description)

```

Figura 4.6: Función del algoritmo web scraping de la descripción de proyectos. Fuente: Elaboración propia.

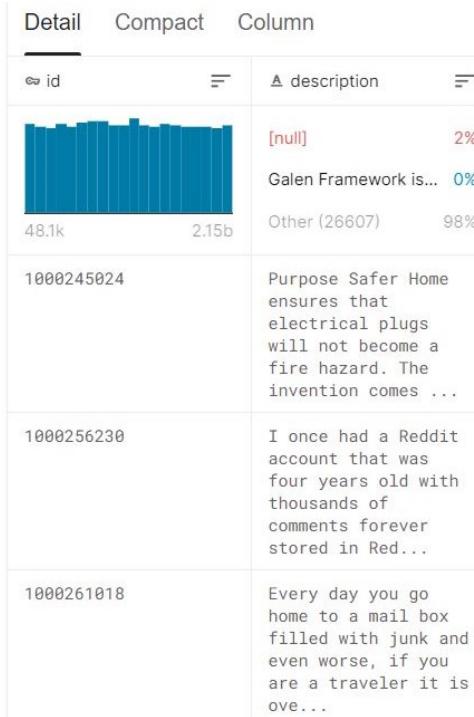


Figura 4.7: Visualización del archivo de descripción subido a Kaggle. Fuente: Elaboración propia.

4.1.3. Comentarios

Al igual que la descripción, los comentarios se obtuvieron utilizando la variable *urls* para web scraping, pero reemplazando caracteres que contengan desde “?ref=” en adelante, por “/comments” para redireccionarse a la sección de comentarios de cada proyecto.

Los comentarios, al ser dinámicos, no podían extraerse mediante la librería BeautifulSoup-Soup como en el caso de las descripciones, por lo que se utilizó la librería Selenium para extraerlos al iniciar una sesión desde Google Chrome, como se observa en el algoritmo de la Figura 4.8.

```
def getPageText(url):
    driver = webdriver.Chrome(options=options)
    driver.set_page_load_timeout(30)
    driver.get(url)
    time.sleep(10)
    count = 0
    while (count<13):
        try:
            loadMoreButton = driver.find_element_by_xpath('//*[@id="react-project-comments"]/div/button')
            time.sleep(2)
            loadMoreButton.click()
            time.sleep(5)
            count += 1
        except (NoSuchElementException, StaleElementReferenceException, TimeoutException):
            break
    time.sleep(5)
    comments = driver.find_elements_by_css_selector('span.bg-ksr-green-700.white.px1.type-14.mr1, div.w100p')
    project_paragraphs = []
    for paragraph in comments:
        project_paragraphs.append(paragraph.text)
    idx_comment_creador = [i for i in range(len(project_paragraphs)) if project_paragraphs[i] == "Creator"]
    idx_comment_creador = [i+1 for i in idx_comment_creador]
    for index in sorted(idx_comment_creador, reverse=True):
        del project_paragraphs[index]
    idx_comment_creador = [i for i in range(len(project_paragraphs)) if project_paragraphs[i] == "Creator"]
    for index in sorted(idx_comment_creador, reverse=True):
        del project_paragraphs[index]
    project_paragraphs = [x for x in project_paragraphs if ("This comment" not in x)]
    project_paragraphs = [x for x in project_paragraphs if ("0:00" not in x)]
    project_paragraphs = [x for x in project_paragraphs if ("Showing " not in x)]
    project_paragraphs = [x for x in project_paragraphs if x]
    project_comments = []
    for project_text in project_paragraphs:
        project_text = project_text.replace(u'\xa0', u' ')
        project_text = project_text.replace(u'\n', u' ')
        project_text = re.sub(useless_characters, '', project_text)
        project_comments.append(project_text)
    del(project_paragraphs)
    driver.quit()
    time.sleep(randint(1,10))
    return project_comments
print("Scraping...")
```

Figura 4.8: Función del algoritmo web scraping de los comentarios. Fuente: Elaboración propia.

La función consiste en redireccionarse a la sección de comentarios del proyecto, esperar un máximo de 30 segundos de carga de la página, hacer clic en el botón “Load More” (Cargar más) hasta un límite de no más de 13 veces y almacenar en un vector cada comentario que no pertenezca al creador (se puede diferenciar gracias a una etiqueta en el extremo superior derecho del recuadro del comentario) de manera independiente. El siguiente paso implica la eliminación de ciertos caracteres especiales, emojis y comentarios completos de un patrocinador que no se encuentren en inglés. Finalmente, estos registros de vectores de comentarios con el id del proyecto correspondiente se almacenaron en un archivo .csv por cada fila terminada, el cual se encuentra disponible públicamente en Kaggle (Figura 4.9) así como los anteriores conjuntos comentados para poder ser descargados.

Detail		Compact	Column
	id	comments	
		71%	[@Creator: What wi... 0%
	48.1k	2.15b	Other (7857)
	1000245024	[]	
	1000256230	[]	
	1000261018	[]	
	100039820	[]	
	1000468345	['Electrical Solutions Group installed the iRNinja at my parents house and they love it! They use to...]	

Figura 4.9: Visualización del archivo de comentarios subido a Kaggle. Fuente: Elaboración propia.

Para optimizar la descarga, se crearon 8 instancias en Google Cloud (Figura 4.10), en donde cada una contenía dos copias del algoritmo, con la cantidad de proyectos fraccionada en 16 partes para que sean ejecutados en paralelo. Si bien el tiempo total de la consolidación de esta base de información duró aproximadamente un mes debido a percances de la conexión interna de las instancias y algunos problemas de ineficiencia de la primera versión del algoritmo, durante el transcurso dentro de este lapso de tiempo fueron solucionados hasta lograr optimizar el algoritmo de web scraping y tener el conjunto final de datos tomó menos de 48 horas.

Compute Engine		Instanc... de VM	+    
 Instancias de VM		Nombre	Zona Recomendación Usada por
 Grupos de instancias		<input type="checkbox"/> Instance-1	us-central1-a
 Plantillas de instancias		<input checked="" type="checkbox"/> scraping-01	us-central1-a
 Nodos de único cliente		<input checked="" type="checkbox"/> scraping-02	us-central1-a
 Imágenes de la máquina		<input checked="" type="checkbox"/> scraping-03	us-central1-a
 Discos		<input checked="" type="checkbox"/> scraping-04	us-central1-a
 Capturas		<input checked="" type="checkbox"/> scraping-05	us-central1-a
		<input checked="" type="checkbox"/> scraping-06	us-central1-a
		<input checked="" type="checkbox"/> scraping-07	us-central1-a
		<input checked="" type="checkbox"/> scraping-08	us-central1-a

Figura 4.10: Instancias lanzadas en paralelo para la extracción de comentarios. Fuente: Elaboración propia.

4.2. Análisis Exploratorios de los datos

Según su estado de financiamiento, los 27,251 proyectos se distribuyen mediante el gráfico de pie de la Figura 4.11. De esta, se observa que más del 70 % de estos fracasaron.

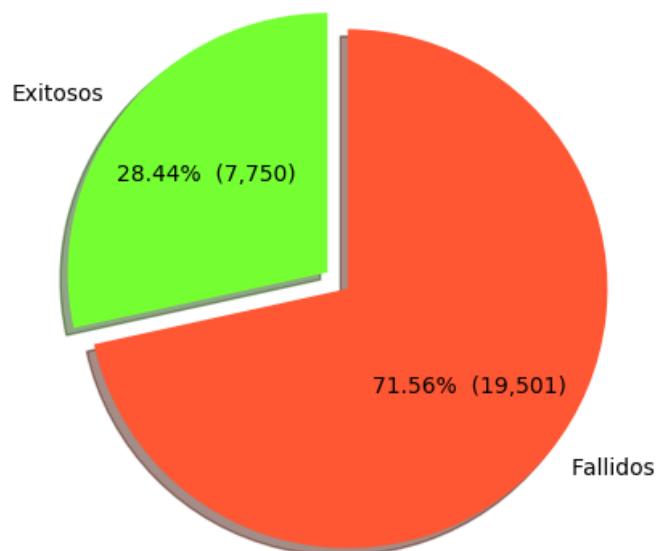


Figura 4.11: Distribución de proyectos tecnológicos según su estado. Fuente: Elaboración propia.

Al visualizarlos por año (Figura 4.12), se puede ver que el histórico comprende entre los períodos 2009 y 2019, siendo la gran mayoría perteneciente al año 2015.

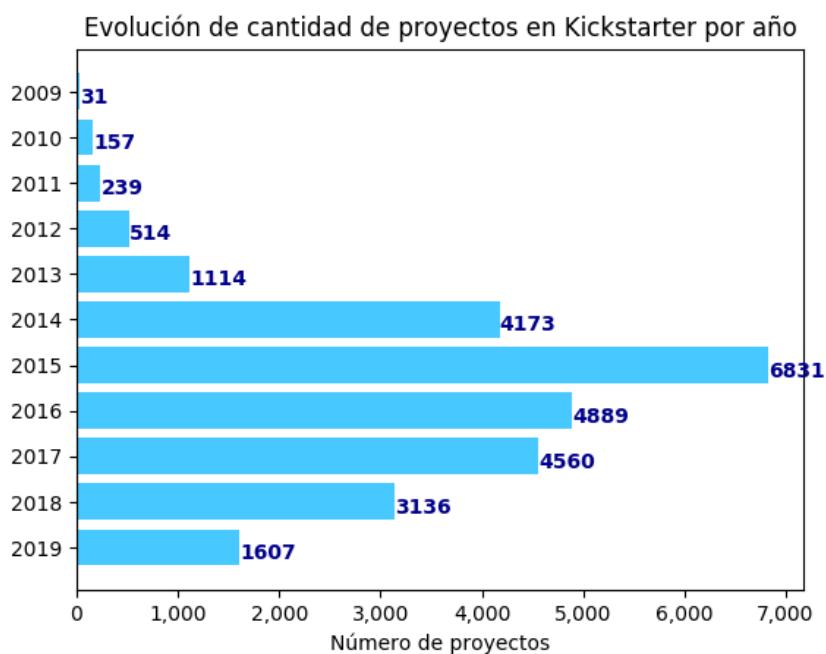


Figura 4.12: Evolución de cantidad de proyectos tecnológicos por año. Fuente: Elaboración propia.

La evolución histórica detallada por su estado se observa en la Figura 4.13.

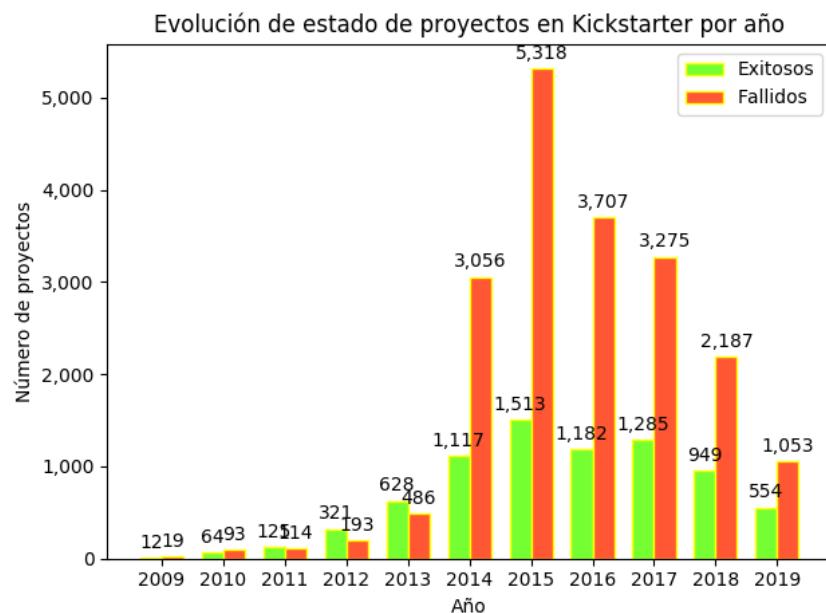


Figura 4.13: Evolución de proyectos tecnológicos, por su estado y año. Fuente: Elaboración propia.

4.2.1. Metainformación

De las 19 variables de la Tabla 4.1, se consideraron como potenciales variables independientes a 5 numéricas (*backers_count*, *goal*, *pledged*, *usd_pledged* y *duration*), 3 categóricas (*category*, *country* y *currency*) y 1 del tipo lista compuesta por números (*pledge_amounts*).

Para las variables categóricas, se realizaron gráficos de pastel para observar la distribución de sus valores. Así tenemos la Figura 4.14 para las categorías de tecnología, Figura 4.15 para los países (por código) y la Figura 4.16 para las divisas del monto final patrocinado (por código). De estos tres gráficos, se observa que más de la mitad de los patrocinadores provienen de los Estados Unidos (US) e invierten en dólares (USD). El segundo y tercer lugar los constituyen personas de Gran Bretaña (GB) y Canadá (CA), invirtiendo en sus monedas locales respectivas (GBP y CAD). Por el lado de las categorías, la más recurrente son de Apps, representando casi la cuarta parte del universo observado.

Para las variables numéricas, se calcularon sus datos estadísticos con ayuda de diagramas de caja y bigote:

- Número de patrocinadores de la campaña (*backers_count*):

- Rango de valores: [0; 105,857]

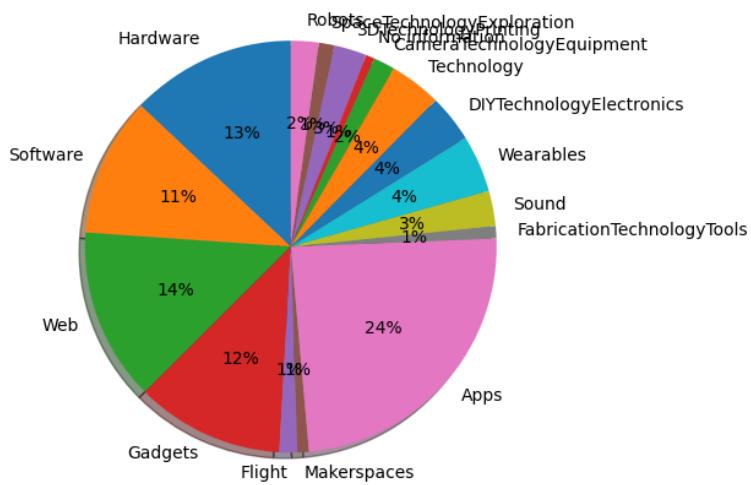


Figura 4.14: Distribución de categorías de tecnología. Fuente: Elaboración propia.

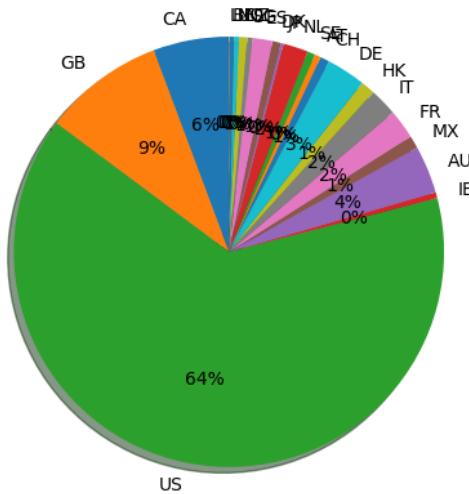


Figura 4.15: Distribución de países. Fuente: Elaboración propia.

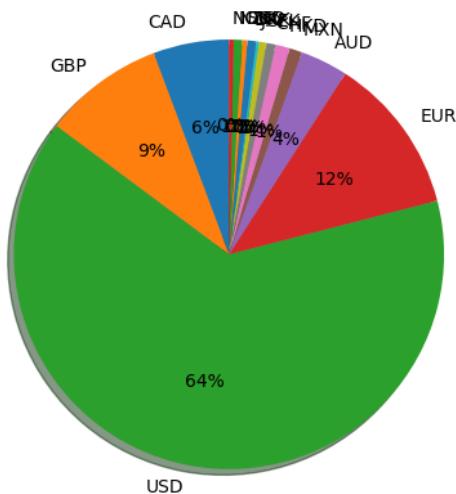


Figura 4.16: Distribución de divisa del monto patrocinado. Fuente: Elaboración propia

- Media: 208.710469340575
- Mediana: 9.487
- Desviación estándar: 1,179.68237749203
- Varianza: 1,391,650.51176525



Figura 4.17: Diagrama de caja y bigote de patrocinadores. Fuente: Elaboración propia.

■ Monto meta de la campaña (*goal*):

- Rango de valores: [1; 100,000,000]
- Media: 91,263.9666162825
- Mediana: 15,762.614
- Desviación estándar: 1,259,282.1587922
- Varianza: 1,585,791,555,452.35

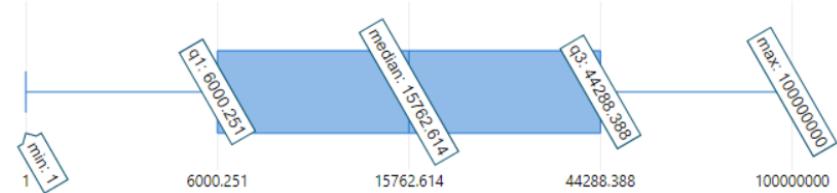


Figura 4.18: Diagrama de caja y bigote de meta. Fuente: Elaboración propia.

■ Monto patrocinado al final de la campaña (*pledged*):

- Rango de valores: [0; 17,406,300]
- Media: 34,668.5134710787
- Mediana: 1,382.933
- Desviación estándar: 226,763.900313481
- Varianza: 51,421,866,485.3822

■ Duración de la campaña (*duration*):

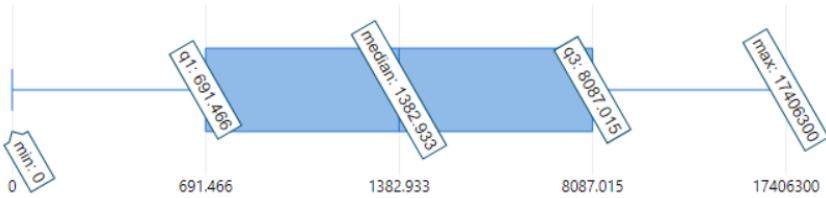


Figura 4.19: Diagrama de caja y bigote de monto patrocinado. Fuente: Elaboración propia.

- Rango de valores: [1; 92]
- Media: 35.4654141132436
- Mediana: 30
- Desviación estándar: 11.845708629999998
- Varianza: 140.320812946853

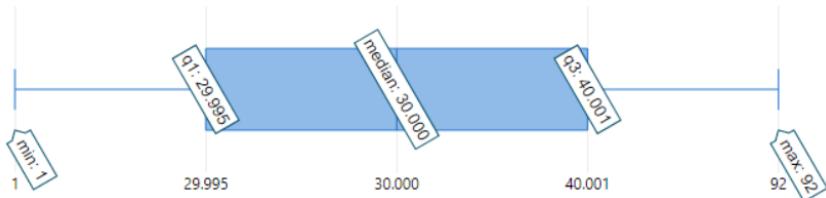


Figura 4.20: Diagrama de caja y bigote de duración. Fuente: Elaboración propia.

Posterior a este entendimiento de datos, se elaboró una matriz de correlaciones (Figura 4.21) para encontrar correlaciones entre ellas y determinar la existencia de alguna variable redundante y descartarla para no afectar el rendimiento del modelo.

Como se puede apreciar en la figura anterior, la variable *usd_pledged* está altamente correlacionada con las variables *backers_count* y *pledged* (ambas con un aproximado de 70%). Esto quiere decir que dicha variable no es significativa porque explicaría de manera muy similar a las otras dos.

Asimismo, si se observan los registros desde una matriz que contiene, además de gráficos de dispersión de las correlaciones, histogramas de las variables independientes como en la Figura 4.22, se confirma y concluye no utilizar las observaciones comentadas.

El mismo gráfico, segmentado por las dos clases del estado de financiamiento, se aprecia en la Figura 4.23.

Se concluye entonces que la variable *duration* es la única que sigue una distribución normal.

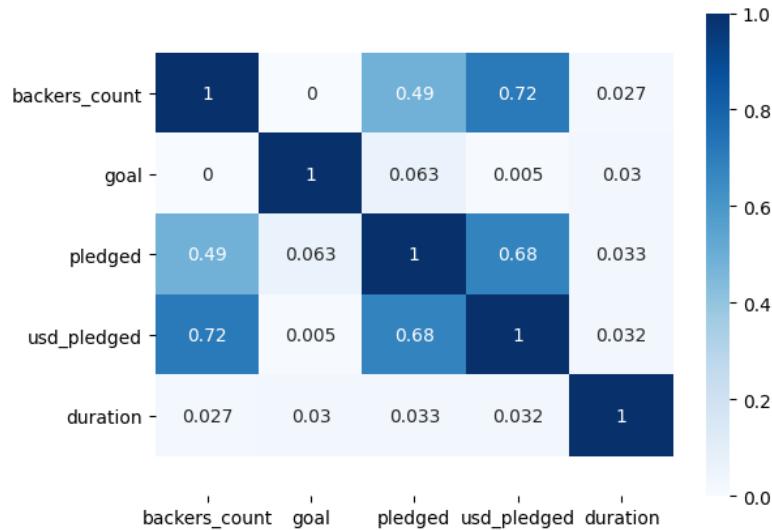


Figura 4.21: Matriz de correlaciones entre variables independientes. Fuente: Elaboración propia.

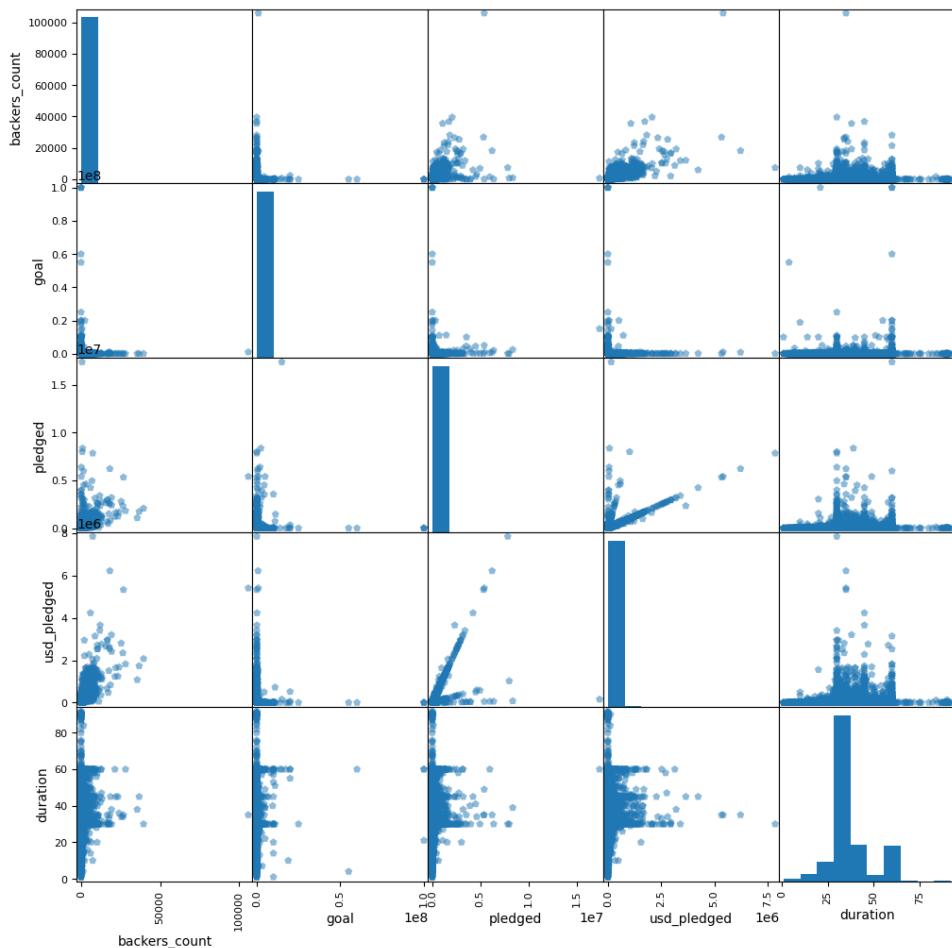


Figura 4.22: Gráfico de dispersión de correlaciones entre variables independientes. Fuente: Elaboración propia.

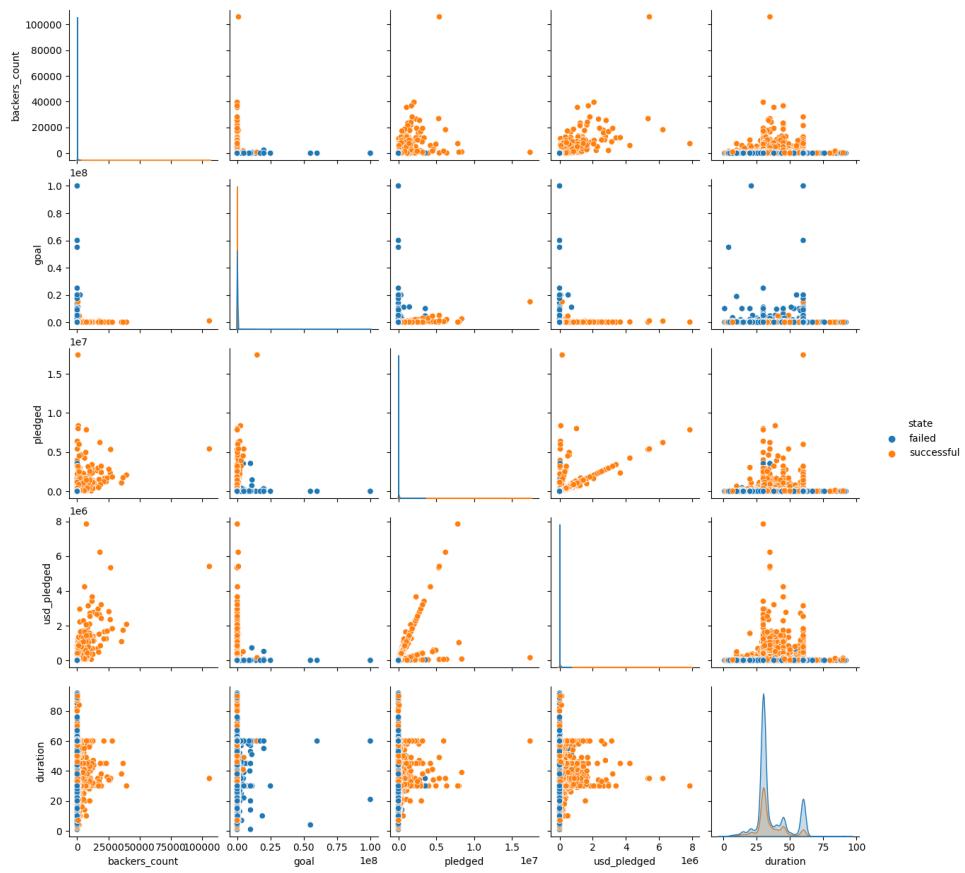


Figura 4.23: Gráfico alterno de dispersión de correlaciones entre variables independientes. Fuente: Elaboración propia.

4.2.2. Descripción

Del conjunto de datos, solamente 640 proyectos (aproximadamente 2 % de la población) no presentaron descripciones por razones externas durante el proceso de extracción de datos en el Web Scraping. De esta cantidad, 512 (80 % de proyectos sin descripciones) no fueron financiados al final de su campaña, es decir, su estado fue fracasado. Mientras que del subconjunto de proyectos con descripciones, casi el 30 % fueron exitosos (Figura 4.24).

En este universo, el registro con descripción de mayor longitud presentó 5,152 palabras y, a nivel total de proyectos, el vocabulario fue de 165,683 palabras. La Figura 4.25 representa la nube de palabras del contenido de cada descripción, donde las más destacadas son aquellas de mayor dimensión.

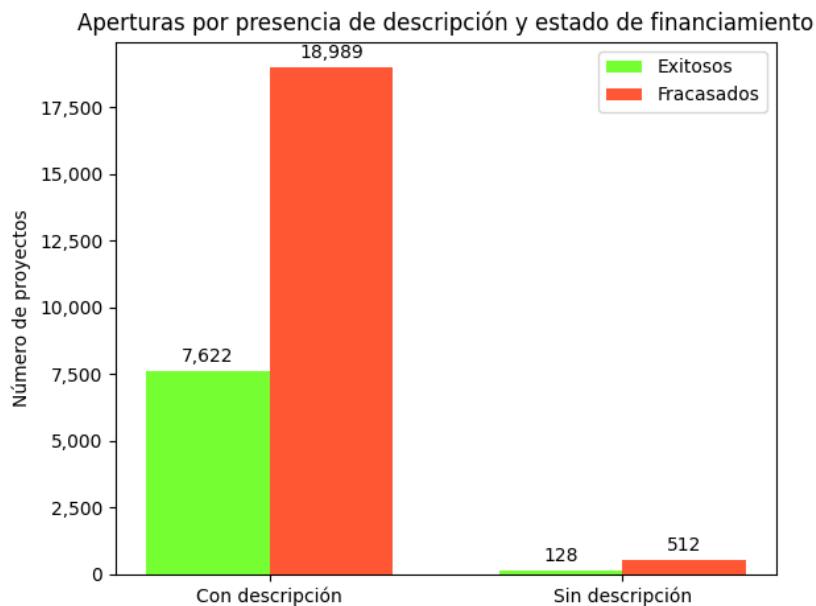


Figura 4.24: Aperturas de proyectos por presencia de comentarios y estado de financiamiento.
Fuente: Elaboración propia.

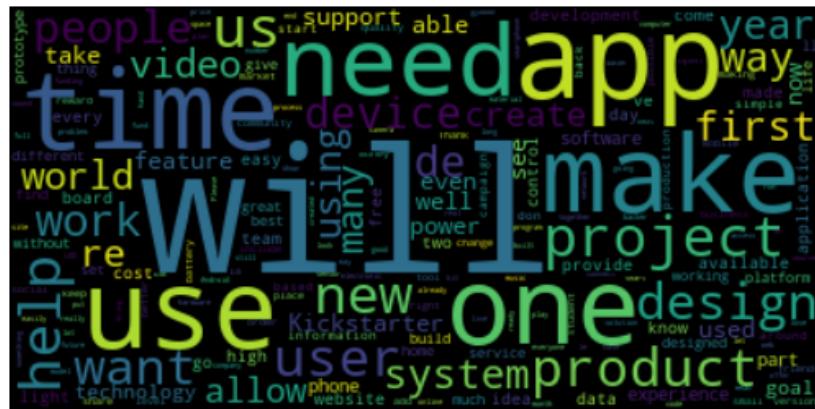


Figura 4.25: Nube de palabras de descripciones. Fuente: Elaboración propia.

4.2.3. Comentarios

Al analizar los proyectos exitosos y fracasados de acuerdo a la presencia de comentarios (Figura 4.26), se observa que hay una sólida relación entre aquellos que fracasaron y los que no cuentan con comentarios, mientras que para los proyectos exitosos, la asociación es menos contundente ya que la diferencia entre aquellos que presentan comentarios y los que no es menor.

Ordenando la distribución por presencia de comentarios (Figura 4.27), se observa que

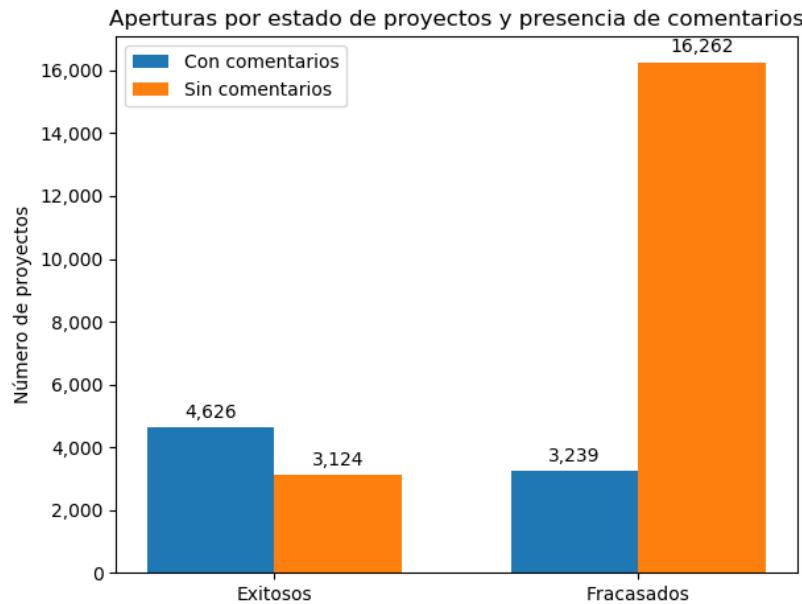


Figura 4.26: Aperturas de proyectos por estado de financiamiento y presencia de comentarios.
Fuente: Elaboración propia.

del total de los proyectos con comentarios, el 60% de ellos (4,626 registros) fueron exitosos, encontrándose casi balanceado este subconjunto.

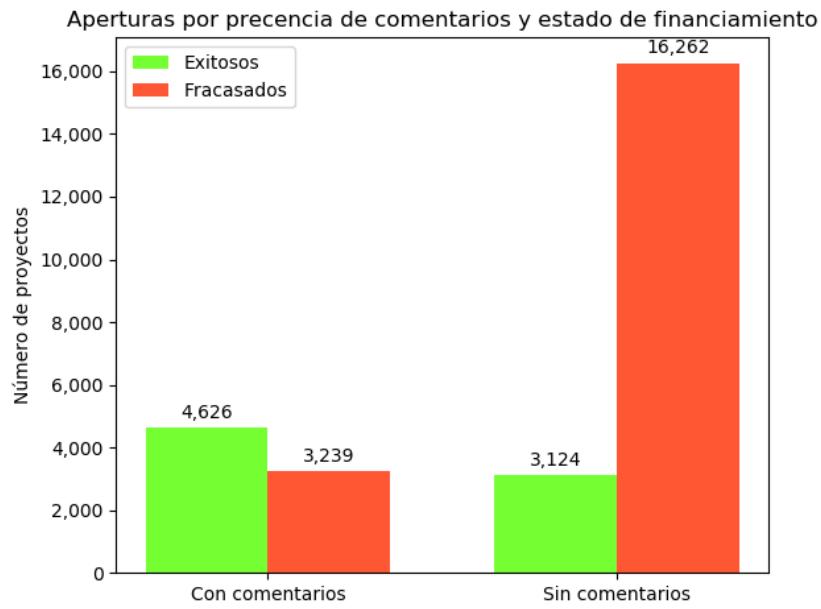


Figura 4.27: Aperturas de proyectos por presencia de comentarios y estado de financiamiento.
Fuente: Elaboración propia.

Por último, del universo de proyectos tecnológicos que presentan comentarios, los más

de 4 mil proyectos exitosos contienen casi la totalidad de comentarios registrados (97 %), representando más de 475 mil como se aprecia en la Figura 4.28. Estos datos confirman la correlación existente entre la presencia de un volumen considerable de comentarios y su éxito de financiación.



Figura 4.28: Distribución de comentarios en proyectos exitosos y fracasados. Fuente: Elaboración propia.

Respecto al contenido, en la Figura 4.29 se ilustra la nube de palabras más frecuentes, respectivamente, luego de quitar URLs, emoticonos y términos en idioma distinto al inglés.

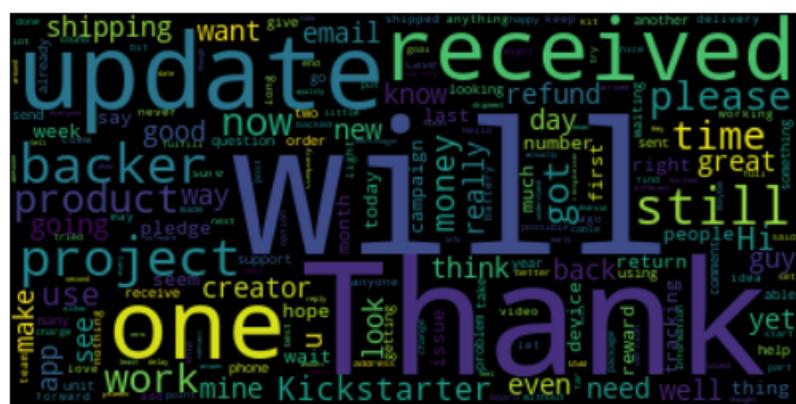


Figura 4.29: Nube de palabras de comentarios más frecuentes. Fuente: Elaboración propia.

4.3. Pre-procesamiento de los conjuntos de datos

4.3.1. Metainformación

De acuerdo a los autores [K. Chen y col., 2013](#) (primer antecedente), [S.-Y. Chen y col., 2015](#) (cuarto antecedente) y [Jin y col., 2019](#) (decimotercer antecedente), a las 5 potenciales variables numéricas se adicionaron 7 variables basadas en el mecanismo financiero (media (pledges_median), promedio (pledges_mean), valor máximo (pledges_max), valor mínimo (pledges_min), variación estándar (pledges_std) y cantidad de montos disponibles para contribuir (pledges_num)) y efecto progresión (porcentaje de financiamiento o completitud (completeness) del monto prometido). Esta última se calcula dividiendo el monto alcanzado en el tiempo t, sobre la meta de la campaña, multiplicado por 100 %. La nueva matriz de correlación se observa en la Figura 4.30.

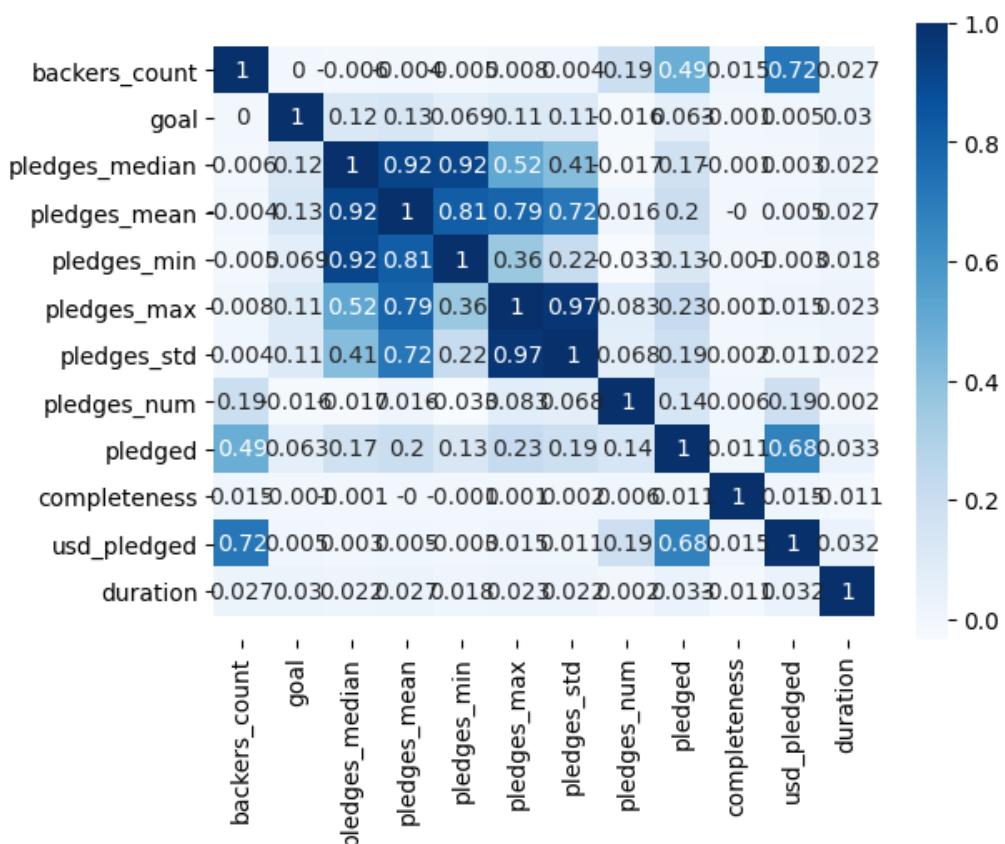


Figura 4.30: Matriz de correlaciones entre variables independientes considerando adicionales.
 Fuente: Elaboración propia.

Para la selección de variables, se consideró a aquellas con correlación clasificada como insignificante, es decir, menor o igual a 0.30 ([Mukaka, 2012](#)). Las únicas que cumplen son

goal, pledges_num, completeness y *duration*. Sin embargo, algunas de las restantes pueden ser consideradas condicionándose a excluir otras. En la Tabla 4.2 se listan las 8 combinatorias posibles de variables que se pueden formar.

Combinación 1	Combinación 2	Combinación 3	Combinación 4
goal	goal	goal	goal
completeness	completeness	completeness	completeness
duration	duration	duration	duration
pledges_num	pledges_num	pledges_num	pledges_num
backers_count	backers_count	backers_count	backers_count
pledges_median	pledges_mean	pledges_min	pledges_max
		pledges_std	

Combinación 5	Combinación 6	Combinación 7	Combinación 8
goal	goal	goal	goal
completeness	completeness	completeness	completeness
duration	duration	duration	duration
pledges_num	pledges_num	pledges_num	pledges_num
pledged	pledged	pledged	pledged
pledges_median	pledges_mean	pledges_min	pledges_max
		pledges_std	

Tabla 4.2: Combinatorias posibles de variables de metadata. Fuente: Elaboración propia.

4.3.2. Descripción

Para poder entrenar un modelo de clasificación binaria basado en texto, se debe preprocesar siguiendo el fluograma de Figura 4.31, que complementa la limpieza de texto referidos previamente ([Mitra y Gilbert, 2014](#), segundo antecedente; [H. Yuan y col., 2016](#), séptimo antecedente; y [L.-S. Chen y Shen, 2019](#), decimoquinto antecedente) con el proceso dictado en el curso de Procesamiento de Lenguaje Natural en la Escuela Superior de Economía de la Universidad Nacional de Investigación, Rusia ([Zimovnov, 2018](#)). Antes de ejecutar cada paso, los registros de proyectos sin descripciones, es decir, con valor nulo (*NaN*), se convirtieron en cadena (*string*) para evitar problemas de procesamiento de texto. Se remueven las contracciones, caracteres especiales, enlaces externos y contenidos en otros idiomas. Este resultado será tokenizado a palabras individuales con la intención de eliminar palabras de parada en inglés, lematizar las restantes y finalmente juntarlas en una lista, separadas por su proyecto correspondiente.



Figura 4.31: Flujograma de limpieza de conjunto de datos de descripciones. Fuente: Elaboración propia.

Para seguir con cada paso del flujo, se usaron elementos de la biblioteca para procesamiento de lenguaje natural Natural Language Toolkit (NLTK), como por ejemplo *word_tokenize*, *stopwords* y *WordNetLemmatizer*. Como resultado de la secuencia, la descripción de mayor longitud pasó a presentar 3,671 palabras y a nivel general de proyectos, el nuevo vocabulario tuvo 165,526 palabras.

Las nubes de palabras reflejan las palabras más frecuentes dentro de un conjunto de datos. La Figura 4.32 representa aquellas palabras que más aparecen en las descripciones de proyectos de tecnología en Kickstarter.

4.3.3. Comentarios

La base de datos de comentarios está conformada a nivel de 1 proyecto con una lista de comentarios, separados en sublistas y diferenciados entre ellos por autor (patrocinador). De los

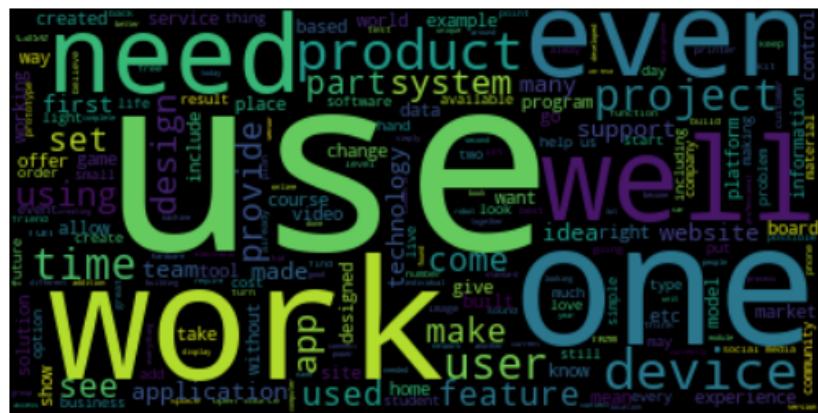


Figura 4.32: Nube de palabras de descripciones posterior al pre-procesamiento. Fuente: Elaboración propia.

7,750 proyectos con comentarios (de los cuales, 4,626 fueron exitosos), se removieron aquellos que presentaron términos en idioma distinto al inglés, así como URLs, emoticonos, emojis, números y caracteres especiales, resultando finalmente en 7,658 proyectos con comentarios (4,574 exitosos).

Debido a la gran cantidad de registros carecientes de comentarios, se propuso rellenarlos con un término aleatorio no relacionado con las temáticas principales: *kuwagatabaizan*. Posteriormente, se repitió el mismo flujograma de la Figura 4.31 para las descripciones. Sin considerar este nuevo término, la nube de palabras se representa en la Figura 4.33.



Figura 4.33: Nube de palabras de comentarios posterior al pre-procesamiento. Fuente: Elaboración propia.

4.4. Creación de los modelos predictivos

Una vez generadas las variables independientes (observaciones, X) y dependiente (estado de financiamiento, Y), el conjunto de datos es separado en subconjuntos de entrenamiento y prueba, con proporciones de 80 % y 20 % respectivamente (H. Yuan y col., séptimo antecedente; P.-F. Yu y col., undécimo antecedente; L.-S. Chen y Shen, decimoquinto antecedente; Mitra y Gilbert, segundo antecedente; Sawhney y col., octavo antecedente) y se fija un valor de aleatoriedad. Dentro de los parámetros de separación, se establece el argumento de estratificación según la variable Y dentro de la función *train_test_split*, de la siguiente manera:

```
train_test_split(X, Y, test_size = 0.20, stratify = Y, random_state=0)
```

Antes de crear los modelos correspondientes, y después de definir los valores de entrada y parámetros (las subsecciones que se detallarán a continuación), se asigna una semilla inicial con un valor fijado por el usuario con el fin de evitar resultados aleatorios para futuras iteraciones. Se establece, además, una ruta local en donde se almacena cada punto de control basado en la mejora de la pérdida del subconjunto de validación con respecto a su iteración anterior. En caso de un estancamiento de esta última durante 10 épocas, es decir, si el valor de la pérdida no decrementa, el modelo dejará de entrenar. A esta regla se le añade la reducción de la tasa de aprendizaje luego de 5 épocas en caso el valor de la exactitud del subconjunto de validación no refleje un incremento. El objetivo de estas condiciones es evitar el sobreajuste en los modelos.

Por último, es importante asignar un peso distinto para cada una de las dos clases de la variable dependiente *state*. Con el fin de evitar un mal entrenamiento, los pesos de ambas clases se balancean y se almacenan en un diccionario con su etiqueta correspondiente.

4.4.1. Metainformación

Se diseñó el modelo de descripciones basada en un Perceptrón Multicapa (MLP por sus siglas en inglés) bajo la arquitectura de la Figura 4.34 y teniendo como referencia a los autores P.-F. Yu y col. (undécimo antecedente). Se asignaron 100 épocas y el número de lotes fue 32.

La arquitectura comienza con la capa de entrada alimentadas por las 6 variables consideradas, la cual asimismo representará la cantidad de neuronas, tanto de entrada como de salida. Si bien no existe alguna regla general para definir el número de capas óptimas, así como los hiperparámetros que se deben configurar en ellas, se puede utilizar como referencia algunas metodologías como las Reglas del Pulgar ([Ranjan, 2019](#)).

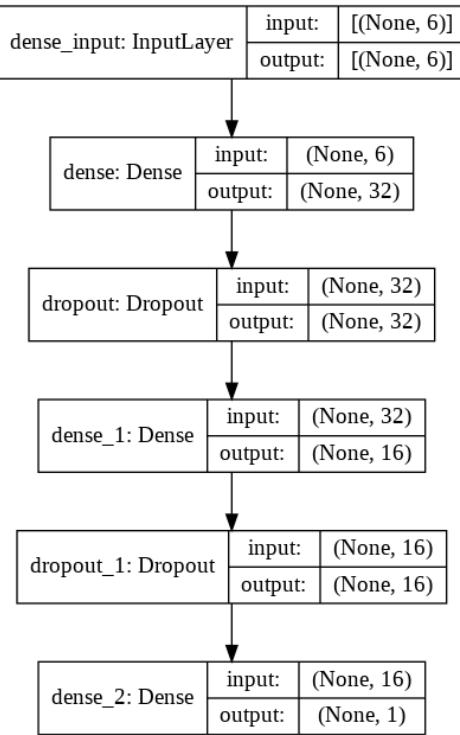


Figura 4.34: Arquitectura de modelo MLP para la metadata. Fuente: Elaboración propia.

De acuerdo a algunas de estas, el número de capas ocultas comienza con 2 sin contar la última. La primera capa densa continúa a la capa de entrada, mientras que la segunda aparece después de la primera capa de desactivación.

Otro punto considerado fue el número de nodos o neuronas de las capas intermedias. Estas deben seguir una progresión geométrica de 2, donde la primera capa debe ser la mitad del número de variables en la capa de entrada. Dado que la mitad de 6 es un valor que no cumple, un número potencial puede ser 4.

El autor de algunas de estas reglas también menciona tener en consideración utilizar la función de activación **relu** para las capas intermedias, una tasa de abandono de por lo menos 0.5 para las capas de desactivación, tamaño de salida de 1 neurona y función de activación **sigmoide** por tratarse de un problema de clasificación binaria, utilizar el optimizador **adam**, comenzar con 20 épocas en adelante de acuerdo al progreso de los resultados y fijar un tamaño de lote bajo progresión geométrica de 2; además de otros requerimientos previamente establecidos como la ponderación de clases para la variable dependiente en caso de datos desbalanceados y escalado de datos antes del entrenamiento.

Todas estas opciones fueron probadas en el modelo y evaluadas con las métricas correspondientes. Sin embargo, al calibrar el modelo y comparar distintos resultados, se obtuvo que la mejor cantidad de neuronas para la primera capa densa era de 32. De este modo, la siguiente

capa intermedia se le asignó la mitad (16). La función de activación **tanh** para la segunda capa oculta presentó mejores resultados, así como tasas de abandono entre 0.25 y 0.3 para las capas de desactivación. Por último, además de *adam*, se realizó experimentos con otros optimizadores como por ejemplo *RMSprop* siendo este el resultado más cercano. Al final, *adam* fue escogido pero con una tasa de aprendizaje baja como 0.005 debido a que el modelo tenía a aprender muy rápido durante el transcurso de las épocas. En conjunto, el ratio de decaimiento se asignó un valor más bajo, 0.00005, y para evaluar los subconjuntos de entrenamiento y prueba se eligió precisión como métrica.

4.4.2. Descripción

Para crear la capa de incrustación de palabras, se usó la función *Tokenizer* de la librería **tensorflow.keras.preprocessing.text**, creando una función para originar un diccionario de palabras a índice al subconjunto de entrenamiento, asignándose a cada una un código. La función creada asimismo permite colocar un valor (en este caso, el término *<OOV>*) a aquellos términos no entrenados y serán parte de la data de prueba (Figura 4.35). A continuación, los arreglos de estos textos codificados son llenados con ceros a la derecha, hasta uniformizar con la longitud de la descripción más larga, ya que todas presentan distintos tamaños (Figura 4.36).

```
oov_tok = "<OOV>"  
  
# fit a tokenizer  
def create_tokenizer(lines):  
    tokenizer = Tokenizer(oov_token=oov_tok)  
    tokenizer.fit_on_texts(lines)  
    return tokenizer  
  
tokenizer = create_tokenizer(training_sentences)
```

Figura 4.35: Función de tokenización de palabras de descripciones. Fuente: Elaboración propia.

```
# encode a list of lines  
def encode_text(tokenizer, lines, length):  
    # integer encode  
    encoded = tokenizer.texts_to_sequences(lines)  
    # pad encoded sequences  
    padded = pad_sequences(encoded, maxlen=length, padding='post')  
    return padded  
  
# encode data  
training_sentences = encode_text(tokenizer, training_sentences, length_long_sentence)  
testing_sentences = encode_text(tokenizer, testing_sentences, length_long_sentence)
```

Figura 4.36: Función de codificación de palabras y relleno de arreglos. Fuente: Elaboración propia.

Una vez obtenido el vocabulario de palabras únicas y asignado el tamaño de cada arreglo (en este caso, se asignó el de la descripción de mayor longitud), se procedió a elaborar la matriz de características usando incrustaciones de GloVe (Figura 4.37). Para la presente investigación, se seleccionó la opción Wikipedia 2014 + Gigaword 5, con una matriz de 100 columnas, donde cada una contendrá las incrustaciones de palabras GloVe para las palabras del corpus, cuyos índices se corresponderán con cada número de fila (Malik, 2019).

```

embeddings_dictionary = dict()
glove_file = open('Glove/glove.6B.100d.txt', encoding="utf8")

for line in glove_file:
    records = line.split()
    word = records[0]
    vector_dimensions = asarray(records[1:], dtype='float32')
    embeddings_dictionary [word] = vector_dimensions

glove_file.close()

embedding_matrix = zeros((vocab_size, embedding_dim))
for word, index in tokenizer.word_index.items():
    embedding_vector = embeddings_dictionary.get(word)
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector

```

Figura 4.37: Elaboración de matriz de incrustaciones de palabras. Fuente: Elaboración propia.

Se diseñó el modelo de descripciones basada en una Red Neuronal Convolutacional bajo la arquitectura de la Figura 4.38, así como de referencia un trabajo de análisis de sentimientos de películas (Malik, 2019). Se asignaron 100 épocas para entrenar y número de lotes de 128.

Esta red se compone de una capa de incrustación de palabras o *Embedding* alimentada por los datos de entrada en la primera capa *InputLayer* de dimensión de 3,671 vectores de palabras (la mayor longitud de palabras de todas las descripciones), la cual genera como salida una matriz de 3,671 por 100 (número de columnas de incrustaciones de GloVe). Para esta capa se entrenarán 14,827,000 parámetros como resultado del producto de las 100 columnas mencionadas y 148,270 como el tamaño del vocabulario entrenado.

La siguiente capa es la Convolución en 1 dimensión o *Conv1D* (usado frecuentemente para extraer características de datos de textos por ser unidimensionales) que, con 128 características, 5 de tamaño de kernel y función de activación **relu**, generó una salida de 3,667 por 128; así como 64,128 parámetros entrenables.

A continuación, le sigue la capa de reducción *GlobalMaxPooling*. Al igual que en la convolución, esta también fue unidimensional y se caracteriza por realizar agrupamiento global basado en el valor máximo de los bloques seleccionados para reducir el tamaño del vector.

Esta nueva salida pasa por la capa de aplanamiento o *Flatten*, en donde se multiplican

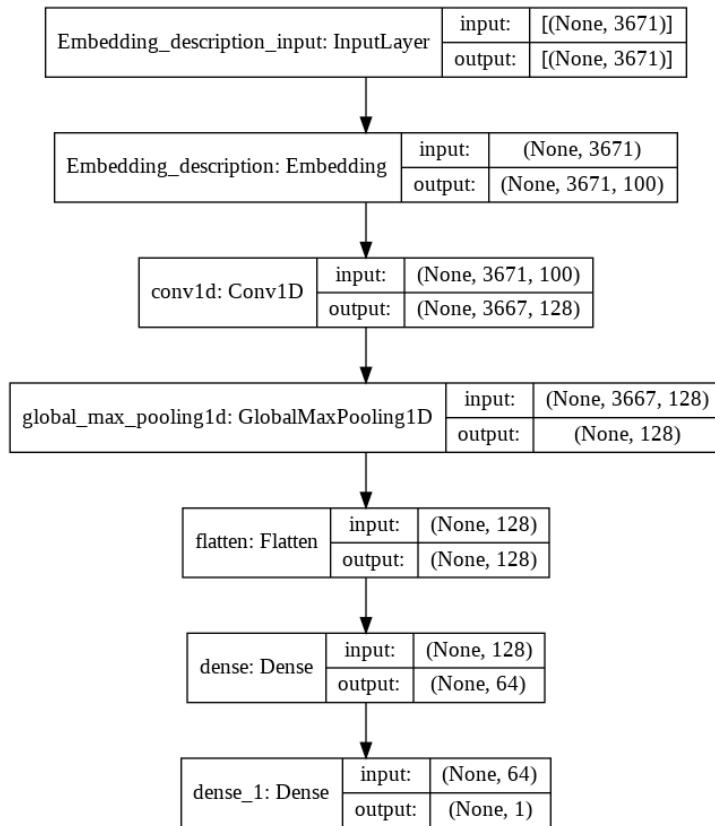


Figura 4.38: Arquitectura de modelo CNN para las descripciones. Fuente: Elaboración propia.

las filas y columnas y tener un solo vector. Esta sirve para conectar con las 64 neuronas de la nueva capa densa y función de activación **tanh**, agregada con el fin de mejorar la performance del modelo. El criterio para considerar una función *tanh* obedece al mejor desempeño durante el entrenamiento que la función *relu*, evitando el sobreajuste durante mayores épocas ocasionado por gradientes que desaparecen al propagar hacia atrás a mayor cantidad de capas. El uso de la función tangente hiperbólica en capas ocultas resultó una buena práctica durante las décadas de 1990 y 2000, teniendo mejor rendimiento que la función logística (Brownlee, 2019). Aún así, ambas son dos opciones válidas para problemas de redes neuronales profundas.

Finalmente, la arquitectura culmina con la última capa con función de activación **sigmoid** para regular el valor de salida entre 0 y 1, ya que se trata de un problema de clasificación binaria (por ello, también cuenta con solo 1 neurona y su parámetro de pérdida es “*binary_crossentropy*”).

4.4.3. Comentarios

Al igual que en el modelo de descripciones de proyectos, se creó un diccionario de palabras con la data luego de tokenizar y codificarlas con las mismas funciones y librerías. Sin embargo, a diferencia del anterior modelo, la longitud de la matriz para el relleno de ceros con el fin de homogenizar el tamaño de cada vector de incrustaciones se limitó a las 5,000 últimas palabras de una oración (parámetro **padding='post'** de la función *pad_sequences*) en lugar de la longitud máxima dado que ésta representa una cantidad considerable (30,072) como para utilizar todos los recursos del entorno de ejecución.

De igual manera, se creó una matriz de incrustaciones de palabras usando GloVe y se diseñó una arquitectura basada en una Red Neuronal Recurrente (RNN por sus siglas en inglés) ilustrada en la Figura 4.39.

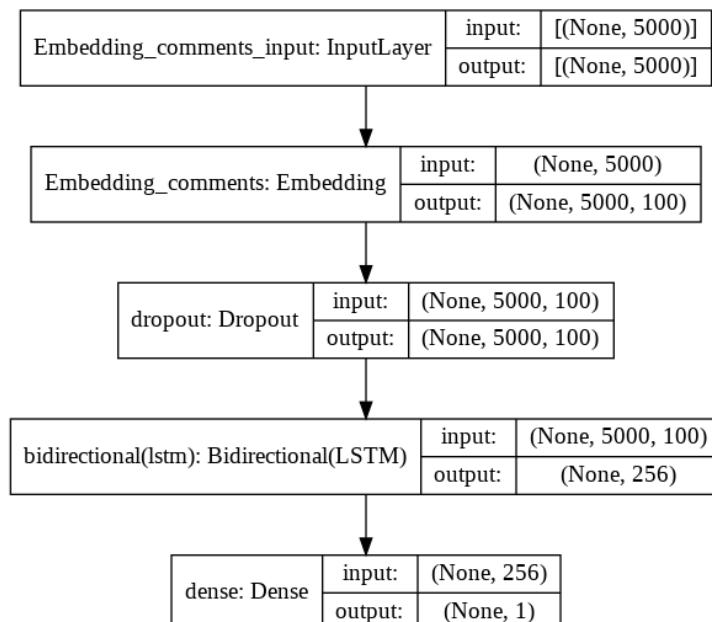


Figura 4.39: Arquitectura de modelo RNN para los comentarios. Fuente: Elaboración propia.

Existe una similitud de estructura en las 2 primeras capas con las del modelo de descripción. Luego de la capa de incrustaciones o *Embedding*, para reducir las conexiones entre neuronas se añadió 1 capa de desactivación o *Dropout*.

A continuación, los vectores de palabras ingresan a la capa de la red neuronal recurrente *LSTM*. Para este caso, se consideró envolverla dentro de una Red Bidireccional de 128 neuronas, ya que como se explicó en el Marco Teórico del Capítulo II sobre las RNN Bidireccionales, este tipo es una mejora de la LSTM tradicional al entrenar 2 juntas (la segunda representa una copia invertida de la secuencia de entrada) en donde cada capa ahora puede considerar también

información de las capas siguientes junto con la información de las previas que ya tenía en cuenta, es decir, toma información de 2 direcciones ([Brownlee, 2017b](#)).

Finalmente, el modelo culmina con una capa densa en donde recibe 256 valores de entrada ($2*128$ de la capa Bidireccional LSTM) y utiliza la función de activación *sigmoid* para transformar el valor final entre 0 y 1.

4.4.4. Modelo apilado

Una vez construidos los modelos para cada modalidad (metainformación, descripción y comentarios), se construyó un modelo de apilamiento integrado ilustrado en la Figura 4.40.

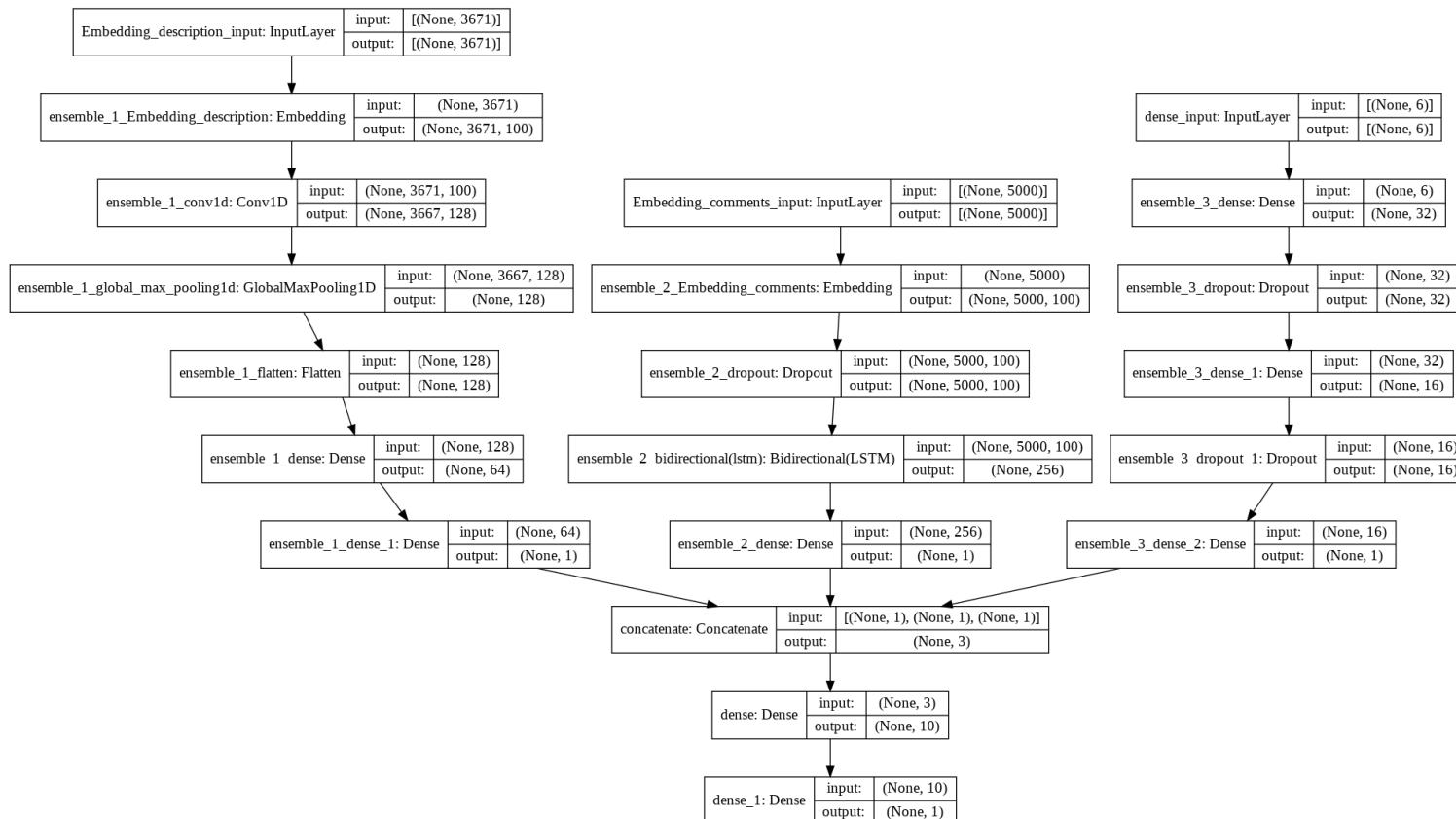


Figura 4.40: Arquitectura del modelo apilado final. Fuente: Elaboración propia.

La finalidad de esta red neuronal de múltiples cabezas es aprender la mejor manera de combinar las predicciones de cada modelo para obtener mejor performance que cada uno individualmente.

A este modelo se le denominó “*The Hydra*” (La Hidra por su traducción al español) en referencia al monstruo mitológico del lago de Lerna, con 7 cabezas que renacían a medida que se cortaban ([Real Academia Española, s.f.](#)).

Las salidas de cada modelo se concatenaron en una capa debajo de estos, generando 3 valores de entrada para una penúltima capa densa con 10 neuronas de salida y una función de activación ReLu. Finalmente, el modelo apilado culmina con una capa densa de 1 salida y asignándose la función Sigmoide para generar probabilidades entre 0 y 1, los valores de Fracasado o Exitoso respectivamente.

Previo a la compilación del modelo final, se repitió el ejercicio de cada modelo cargado asignar los parámetro de pérdida *binary_crossentropy* para la clasificación binaria, *accuracy* (exactitud) para la métrica del entrenamiento, pesos balanceados para las clases de la variable *state* (0.6987077585764833 para 0 y 1.7581290322580645 para 1), y optimizador *Adam* con la variante de asignarle el ratio de aprendizaje y también de decaimiento de 0.00005.

Capítulo 5

ANÁLISIS Y DISCUSIÓN DE RESULTADOS

Como parte de la aplicación de la metodología CRISP-DM, explicada en el sexto subcapítulo del Capítulo III, se mencionaron las métricas usadas en la literatura. La más recurrente fue la exactitud. Dado que la librería Scikit-learn cuenta con un reporte de clasificación con esta métrica y otras 4 más como la precisión, sensibilidad, puntaje F1 y AUC, además que la distribución de proyectos por su estado de financiamiento es desbalanceada y se necesita más de un indicador para poder evaluar y comparar, se decidió usar estas 5 teniendo como referencias a los autores Beckwith (quinto antecedente), H. Yuan y col.* (séptimo antecedente), Kaur y Gera (noveno antecedente), Cheng y col. (decimocuarto antecedente), y L.-S. Chen y Shen** (decimoquinto antecedente).

En el antecedente marcado en (*), los modelos no fueron evaluados por AUC; mientras que en (**), las métricas precisión y AUC no fueron tomadas en cuenta.

5.1. Metainformación

Luego de 19 épocas, con un promedio de 3 segundos de entrenamiento cada una, el modelo dejó de entrenar dado que durante 7 épocas no registró una reducción en el valor de la pérdida del subconjunto de validación, a pesar de que hace 3 épocas se redujo su tasa de aprendizaje.

Así, de acuerdo a la Figura 5.1, en la época 11 se registran los mejores valores de exactitud y pérdida para el subconjunto de validación, alcanzando 0.9523 y 0.1246 respectivamente.

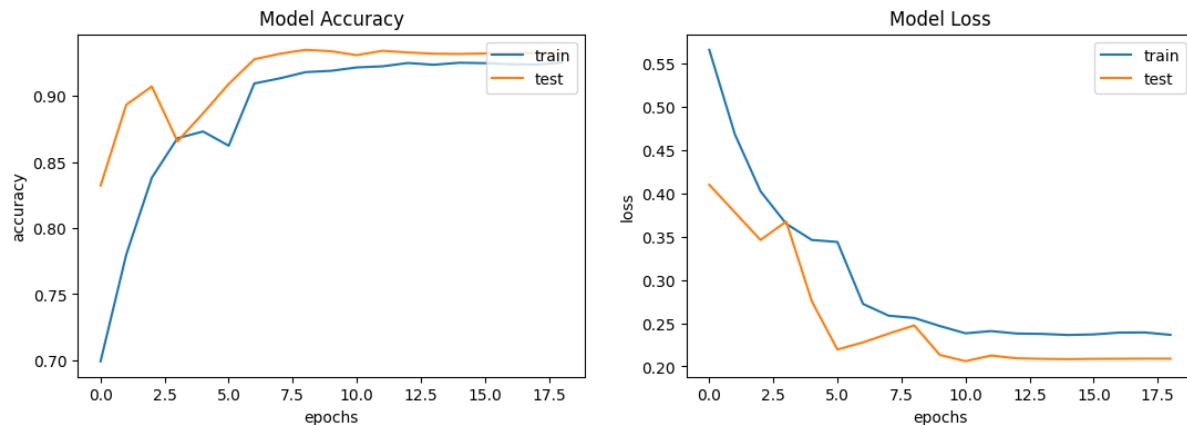


Figura 5.1: Exactitud y pérdida respectivamente de los subconjuntos de entrenamiento y validación para el modelo MLP de metadata con 100 épocas. Fuente: Elaboración propia.

La matriz de confusión resultante se representa en la Figura 5.2.

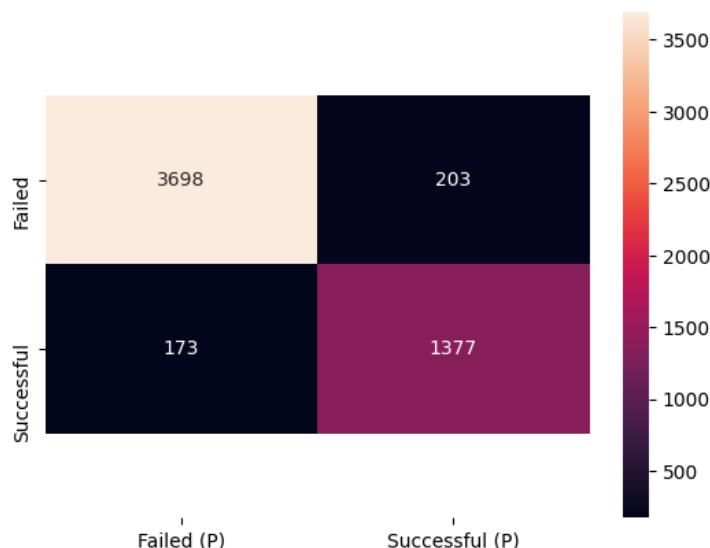


Figura 5.2: Matriz de confusión para el modelo de metadata. Fuente: Elaboración propia.

De esta matriz, se derivan los resultados de la Tabla 5.1 y el AUC en la Figura 5.3.

- El ratio de exactitud se interpreta como: El 93 % de los proyectos de la muestra fueron predichos correctamente.
- El ratio de precisión para los positivos (*Successful*) se interpreta como: El 87 % de los proyectos exitosos predichos de la muestra fueron clasificados correctamente.
- El ratio de sensibilidad para los positivos (*Successful*) se interpreta como: El 89 % de los proyectos exitosos reales de la muestra fueron clasificados correctamente.

Valor	Precisión	Sensibilidad	Puntaje F1	Muestras
Fracasado	0.96	0.95	0.95	3,901
Exito	0.87	0.89	0.88	1,550
Exactitud			0.93	5,451
Promedio macro	0.91	0.92	0.92	5,451
Promedio ponderado	0.93	0.93	0.93	5,451

Tabla 5.1: Informe de clasificación para el modelo de metadata. Fuente: Elaboración propia.

- El ratio de Puntaje F1 para los positivos (*Successful*) representa un balance entre las 2 métricas anteriores. En la tabla se observa que su valor es de 88 %, lo cual indica que en general, el modelo mantiene un alto rendimiento.

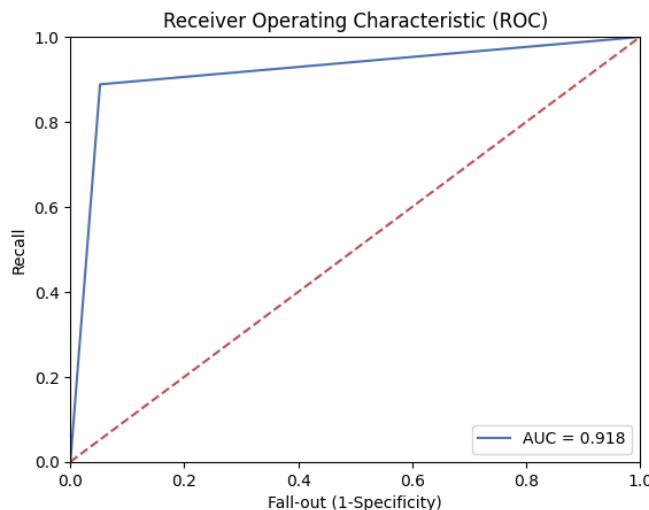


Figura 5.3: Área bajo la curva ROC de modelo de metadata. Fuente: Elaboración propia.

El área bajo la Curva ROC presenta un valor de aproximadamente 92 %, del cual se observa en el gráfico que su sensibilidad es muy alta y el ratio de Falsa Alarma o Falsos Positivos es casi nulo. Esto significa que el poder discriminante del modelo es excelente (Britos y col., 2006).

5.2. Descripción

Luego de 82 épocas, con un promedio de 106 segundos de entrenamiento cada una, el modelo dejó de entrenar dado que durante 10 épocas no registró una reducción en el valor de

la pérdida del subconjunto de validación, a pesar de que hace 5 épocas se redujo su tasa de aprendizaje.

Así, de acuerdo a la Figura 5.4, en la época 72 se registran los mejores valores de exactitud y pérdida para el subconjunto de validación, alcanzando 0.7683 y 0.4901 respectivamente.

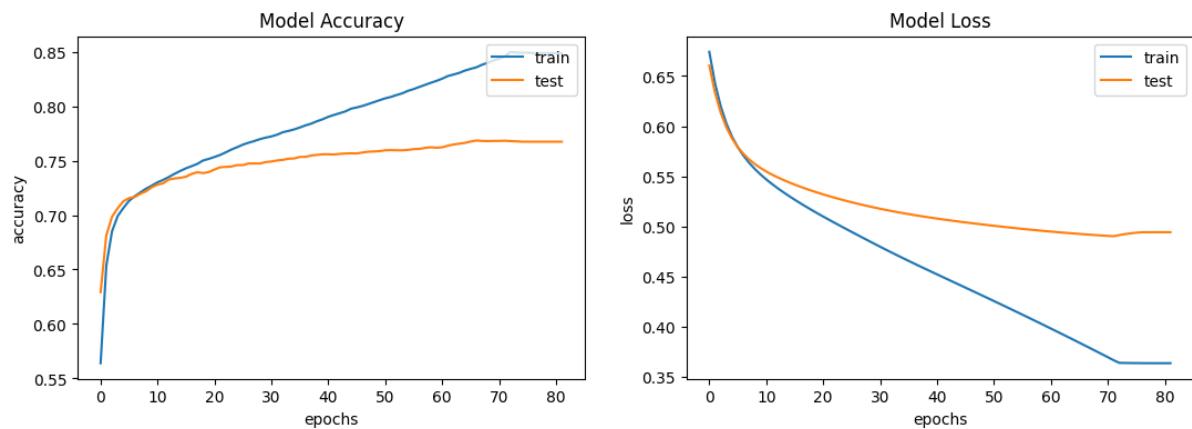


Figura 5.4: Exactitud y pérdida respectivamente de los subconjuntos de entrenamiento y validación para el modelo CNN de descripciones con 100 épocas. Fuente: Elaboración propia.

La matriz de confusión resultante se representa en la Figura 5.5.

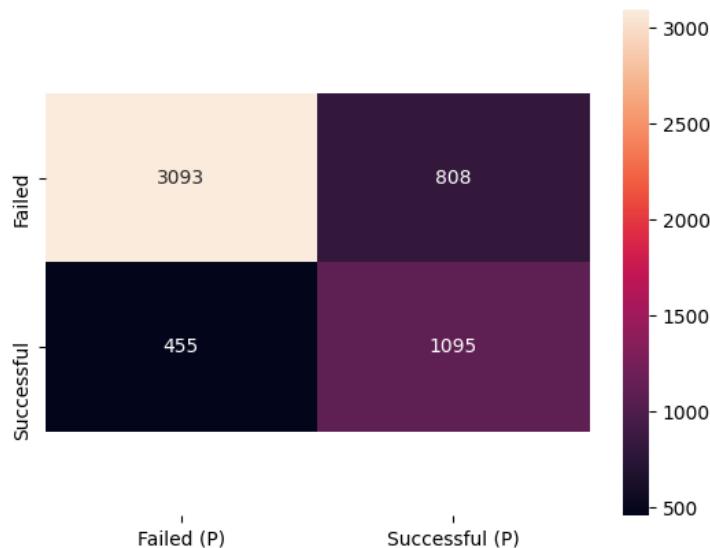


Figura 5.5: Matriz de confusión para el modelo de descripciones. Fuente: Elaboración propia.

De esta matriz, se derivan los resultados de la Tabla 5.2 y el AUC en la Figura 5.6.

- El ratio de exactitud se interpreta como: El 77 % de los proyectos de la muestra fueron predichos correctamente.

Valor	Precisión	Sensibilidad	Puntaje F1	Muestras
Fracasado	0.87	0.79	0.83	3,901
Exito	0.58	0.71	0.63	1,550
Exactitud			0.77	5,451
Promedio macro	0.72	0.75	0.73	5,451
Promedio ponderado	0.79	0.77	0.77	5,451

Tabla 5.2: Informe de clasificación para el modelo de descripciones. Fuente: Elaboración propia.

- El ratio de precisión para los positivos (*Successful*) se interpreta como: El 58 % de los proyectos exitosos predichos de la muestra fueron clasificados correctamente.
- El ratio de sensibilidad para los positivos (*Successful*) se interpreta como: El 71 % de los proyectos exitosos reales de la muestra fueron clasificados correctamente.
- El ratio de Puntaje F1 para los positivos (*Successful*) representa un balance entre las 2 métricas anteriores. En la tabla se observa que su valor es de 63 %, lo cual indica que en general, el modelo presenta un rendimiento regular.

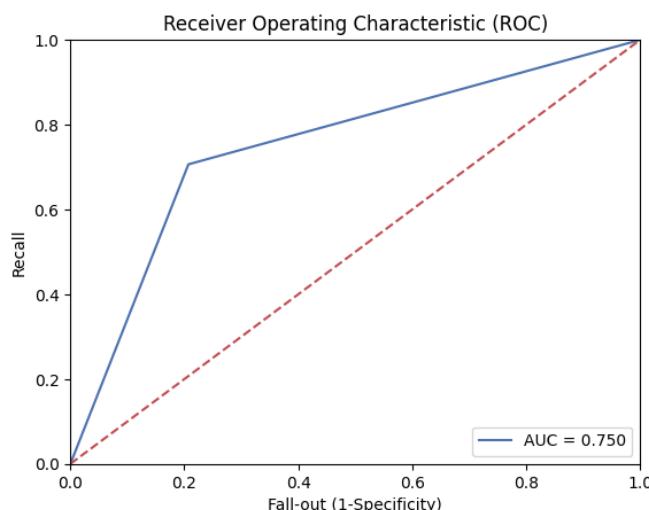


Figura 5.6: Área bajo la curva ROC de modelo de descripciones. Fuente: Elaboración propia.

El área bajo la Curva ROC presenta un valor de aproximadamente 75 %, del cual se observa en el gráfico que su sensibilidad es medianamente alta y el ratio de Falsa Alarma es medianamente baja. Esto significa que el poder discriminante del modelo es aceptable ([Britos y col., 2006](#)).

5.3. Comentarios

Luego de 43 épocas, con 77 segundos en promedio de entrenamiento cada una, el modelo dejó de entrenar dado que durante 10 épocas no registró una reducción en el valor de la pérdida del subconjunto de validación, por más que 1 época antes se había reducido su tasa de aprendizaje.

Así, de acuerdo a la Figura 5.7, en la época 33 se registran los mejores valores de exactitud y pérdida para el subconjunto de validación, alcanzando 0.8510 y 0.4472 respectivamente.

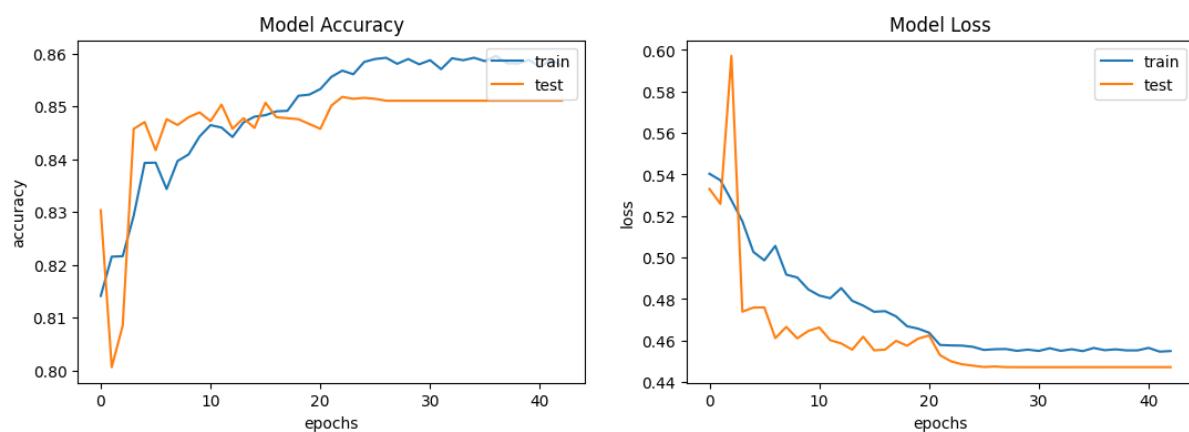


Figura 5.7: Exactitud y pérdida respectivamente de los subconjuntos de entrenamiento y validación para el modelo RNN de comentarios con 50 épocas. Fuente: Elaboración propia.

La matriz de confusión resultante se representa en la Figura 5.8.



Figura 5.8: Matriz de confusión para el modelo de comentarios. Fuente: Elaboración propia.

De esta matriz, se derivan los resultados de la Tabla 5.3 y el AUC en la Figura 5.9.

Valor	Precisión	Sensibilidad	Puntaje F1	Muestras
Fracasado	0.84	0.98	0.90	3,901
Exitoso	0.91	0.53	0.67	1,550
Exactitud			0.85	5,451
Promedio macro	0.87	0.75	0.79	5,451
Promedio ponderado	0.86	0.85	0.84	5,451

Tabla 5.3: Informe de clasificación para el modelo de comentarios. Fuente: Elaboración propia.

- El ratio de exactitud se interpreta como: El 85 % de los proyectos de la muestra fueron predichos correctamente.
- El ratio de precisión para los positivos (*Successful*) se interpreta como: El 91 % de los proyectos exitosos predichos de la muestra fueron clasificados correctamente.
- El ratio de sensibilidad para los positivos (*Successful*) se interpreta como: El 53 % de los proyectos exitosos reales de la muestra fueron clasificados correctamente.
- El ratio de Puntaje F1 para los positivos (*Successful*) representa un balance entre las 2 métricas anteriores. En la tabla se observa que su valor es de 67 %, lo cual indica que en general, el modelo presenta un rendimiento regular.

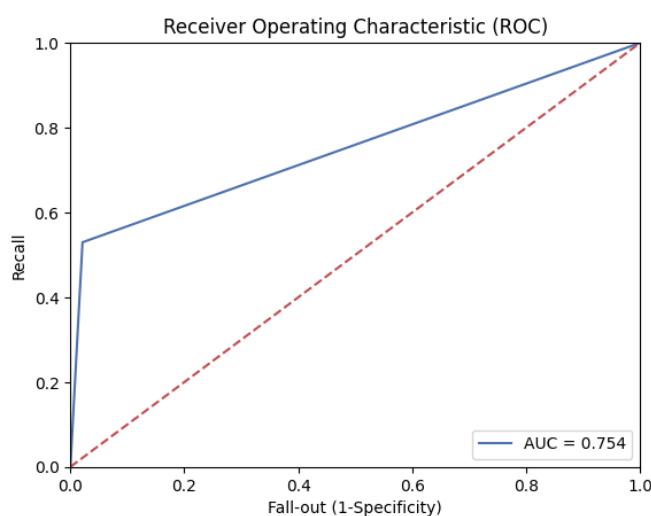


Figura 5.9: Área bajo la curva de modelo de comentarios. Fuente: Elaboración propia.

El área bajo la Curva ROC presenta un valor de aproximadamente 75 %, del cual se observa en el gráfico que su sensibilidad es baja pero su ratio de Falsa Alarma es casi nulo. Esto significa que el poder discriminante del modelo es aceptable ([Britos y col., 2006](#)).

5.4. Modelo apilado

Luego de 13 épocas, con 152 segundos de entrenamientos cada una, el modelo dejó de entrenar dado que durante 10 épocas no registró una reducción en el valor de la pérdida del subconjunto de validación, a pesar de que hace 2 épocas se redujo su tasa de aprendizaje.

Así, de acuerdo a la Figura 5.10, en la época 3 se registran los mejores valores de exactitud y pérdida para el subconjunto de validación, alcanzando 0.9336 y 0.1810 respectivamente.

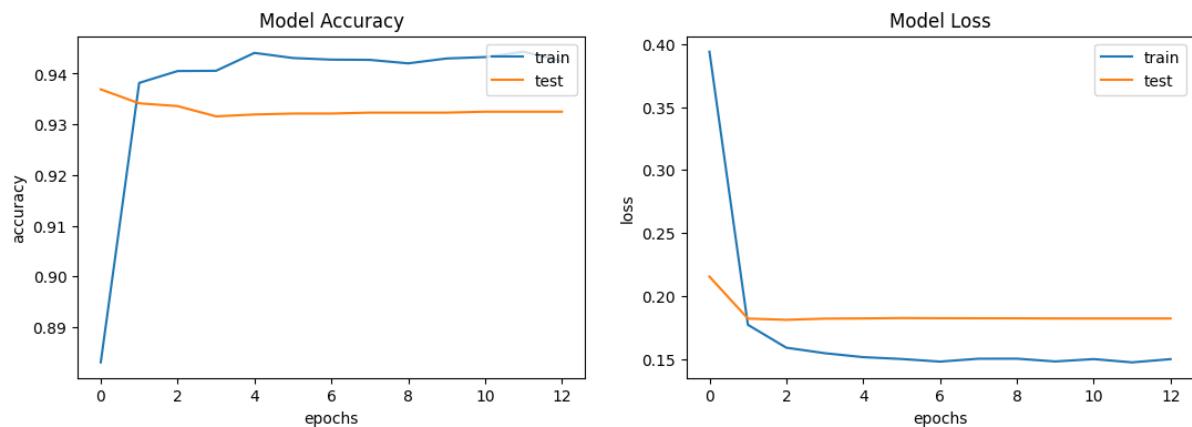


Figura 5.10: Exactitud y pérdida respectivamente de los subconjuntos de entrenamiento y validación para el modelo apilado con 200 épocas. Fuente: Elaboración propia.

La matriz de confusión resultante se representa en la Figura 5.11.

De esta matriz, se derivan los resultados de la Tabla 5.4 y el AUC en la Figura 5.12.

Valor	Precisión	Sensibilidad	Puntaje F1	Muestras
Fracasado	0.96	0.94	0.95	3,901
Exitoso	0.87	0.91	0.89	1,550
Exactitud			0.93	5,451
Promedio macro	0.91	0.93	0.92	5,451
Promedio ponderado	0.93	0.93	0.93	5,451

Tabla 5.4: Informe de clasificación para el modelo apilado. Fuente: Elaboración propia.

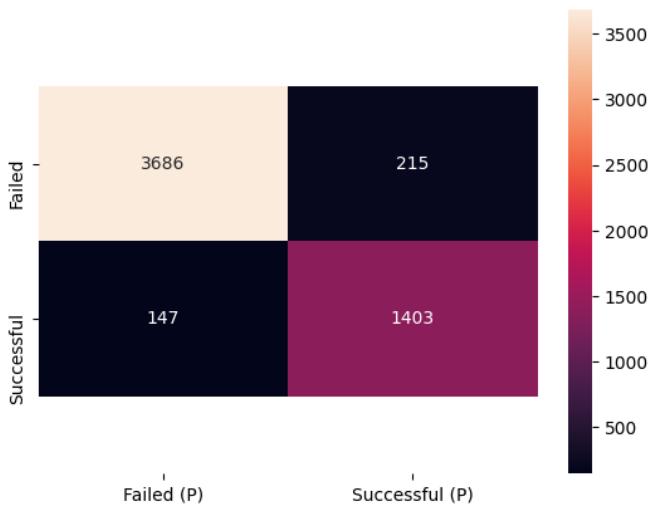


Figura 5.11: Matriz de confusión para el modelo apilado. Fuente: Elaboración propia.

- El ratio de exactitud se interpreta como: El 93 % de los proyectos de la muestra fueron predichos correctamente.
- El ratio de precisión para los positivos (*Successful*) se interpreta como: El 87 % de los proyectos exitosos predichos de la muestra fueron clasificados correctamente.
- El ratio de sensibilidad para los positivos (*Successful*) se interpreta como: El 91 % de los proyectos exitosos reales de la muestra fueron clasificados correctamente.
- El ratio de Puntaje F1 para los positivos (*Successful*) representa un balance entre las 2 métricas anteriores. En la tabla se observa que su valor es de 89 %, lo cual indica que en general, el modelo presenta un rendimiento regular.

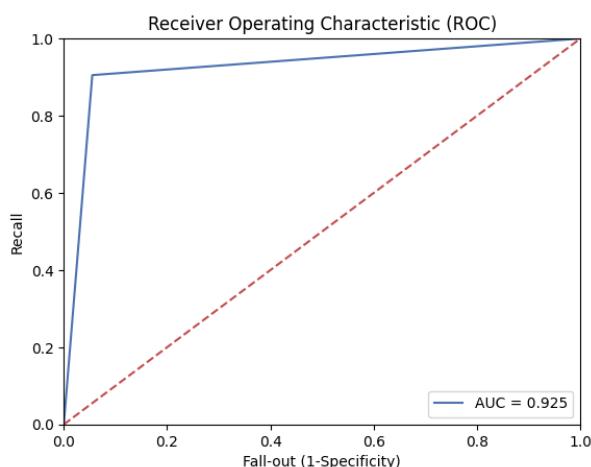


Figura 5.12: Área bajo la curva de modelo apilado. Fuente: Elaboración propia.

El área bajo la Curva ROC presenta un valor aproximado de 93 %, del cual se observa en el gráfico que su sensibilidad es muy alta y su ratio de Falsa Alarma es casi nulo. Esto significa que el poder discriminante del modelo es excepcionalmente bueno ([Britos y col., 2006](#)).

5.5. Comparación de resultados

Considerando los modelos independientes para cada modalidad, así como un trabajo previo del autor de la presente investigación cuyo trabajo sirvió de base ([Puente, 2019](#)), se armó el cuadro comparativo de la Tabla 5.5.

Modelos	Exactitud	Precisión	Sensibilidad	Puntaje F1	AUC
Tesis de pregrado - Metainformación	0.89	0.86	0.72	0.75	0.84
Tesis de pregrado - Descripción	0.75	0.59	0.53	0.35	0.68
Propuesta - Metainformación	0.93	0.91	0.92	0.92	0.92
Propuesta - Descripción	0.77	0.72	0.75	0.73	0.75
Propuesta - Comentarios	0.85	0.87	0.75	0.79	0.75
Propuesta - The Hydra	0.93	0.91	0.93	0.92	0.93

Tabla 5.5: Comparación de resultados de modelos propuestos con antecedentes. Fuente: Elaboración propia.

Los modelos citados de la Tesis de pregrado para Metainformación y Descripción utilizaron una Máquina de Vectores de Soporte (SVM) y una SVM entrenada con el algoritmo TF-IDF, respectivamente.

Para comparar ambas investigaciones, se usaron las mismas bases de datos para las 2 modalidades, con proyectos tecnológicos de Kickstarter finalizados entre 2009 y 2019, así como también las mismas métricas para evaluar cada modelo. El tiempo de entrenamiento en el antecedente mencionado fue mayor (aproximadamente 16 segundos para la metainformación y 4 horas para la descripción), en contraste con los elaborados en este trabajo (aproximadamente 38 segundos para la metainformación y 2 horas y media para la descripción).

5.6. Demostración del modelo final

Para culminar con la investigación, se diseñó un sistema piloto que logra predecir el estado de financiamiento de un determinado proyecto siguiendo el flujo de la Figura 5.13.

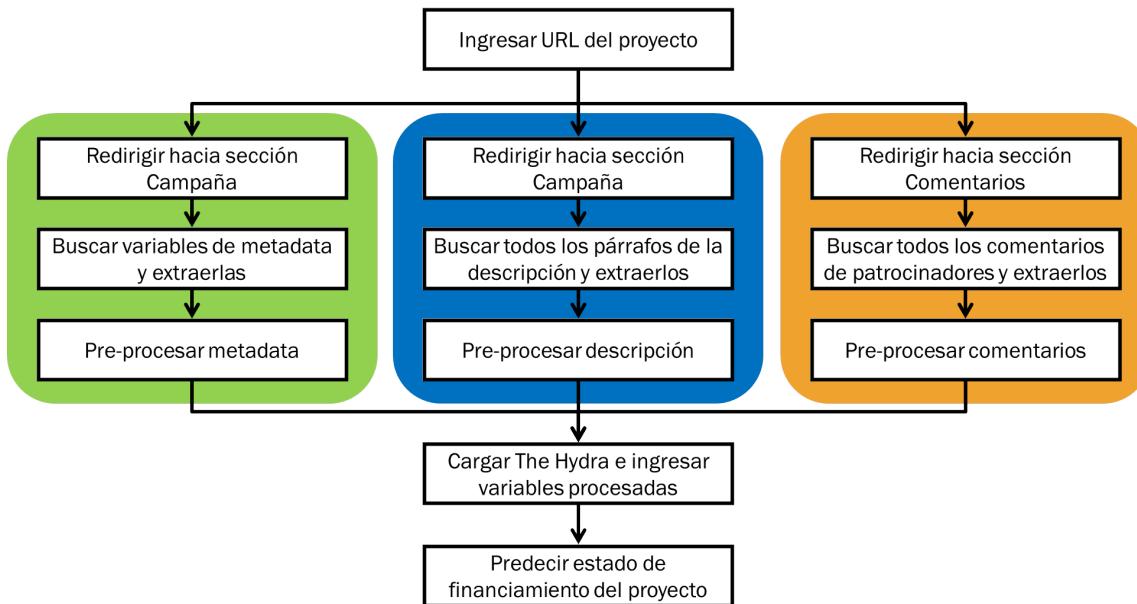


Figura 5.13: Flujograma del sistema piloto. Fuente: Elaboración propia.

Como se observa en la anterior imagen, el software ejecutado localmente desde Jupyter Notebook recibe de entrada el enlace web de un proyecto no finalizado de Kickstarter. Uno de los experimentos hechos el 23 de enero del 2021 se realizó con la campaña de la Figura 5.14.

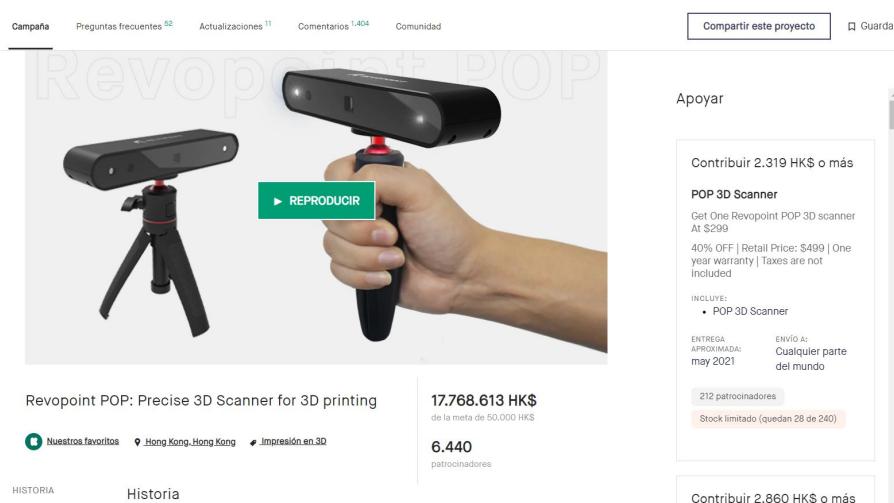


Figura 5.14: Proyecto usado para la demostración. Captura de pantalla: 15/02/21. Fuente: Kickstarter, 2021

La primera acción hecha por el sistema fue extraer la metainformación, descripción y comentarios del proyecto desde el ingreso al URL por un navegador. Debido al cambio de políticas de acceso a la plataforma en 2020, Kickstarter detecta la presencia de bots y restringe la navegación usando CAPTCHAs para evitar su accionar. Algunas veces fue detectado el bot del sistema. Ante ello, la única acción manual por parte del usuario en el sistema se da en este paso presionando por 5 segundos el botón de “*I'm a human*” (Soy humano por su traducción al español) que se muestra en la ventana. Desde este punto, luego de ingresar al enlace de la campaña del proyecto, el sistema primero se redirige a la sección de la metainformación y extrae las variables usadas para el entrenamiento del modelo. Del mismo modo, repite esta secuencia para la descripción y los comentarios.

Los datos extraídos de cada modalidad se muestran en la Figura 5.15 respectivamente.



Figura 5.15: Variables extraídas del proyecto. Fuente: Elaboración propia.

Esta información es pre-procesada de la misma manera que la data usada para entrenar cada modelo. A continuación, el modelo cargado The Hydra recibe los datos procesados y concatenados para realizar la predicción. Si el umbral es por lo menos 0.50, el resultado será **EXITOSO** (*Successful* en inglés) como en la Figura 5.16.

Predicción de probabilidad de financiamiento: 99.86 %
Estado final de la campaña será: EXITOSO

Figura 5.16: Resultado de predicción de The Hydra para el proyecto consultado. Fuente: Elaboración propia.

Capítulo 6

CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

De acuerdo a los resultados mostrados en la Tabla 5.5, el modelo propuesto The Hydra presenta mejor rendimiento que los modelos que lo componen individualmente bajo las 5 métricas evaluadas. Esto confirma que un modelo apilado mejora su capacidad predictiva alimentándose de otros entrenados previamente ([Brownlee, 2018](#)).

Asimismo, a nivel general, la performance también fue mejor que los modelos referenciados en los antecedentes. El concepto (con otros modelos y variantes en el desarrollo) utilizado en la Tesis de pregrado del autor de la presente investigación ([Puente, 2019](#)) se basó en el Antecedente 14 ([Cheng y col., 2019](#)), el cual utilizó un marco de trabajo de Aprendizaje Profundo Multimodal (*Multimodal Deep Learning* en inglés), donde se combinan las características de metainformación, descripción e imagen principal del proyecto en la capa totalmente conectada. Sin embargo, en dicha ocasión no se alcanzó lograr los objetivos dado que el modelo de contenido visual presentó problemas para clasificar adecuadamente un proyecto según su estado. Esto se dio en parte a la variedad de imágenes dentro de la misma categoría Tecnología, que contiene asimismo 16 subcategorías, lo cual dificultó en su momento a la red a encontrar patrones a partir de su características. Se decidió, entonces, cambiar el criterio de reemplazar el contenido visual por otra modalidad respaldada por varios antecedentes (enumerados en la descripción del prototipo de investigación del Capítulo III) y que no fue tomada en cuenta en su momento, los comentarios realizados durante la campaña por los patrocinadores del proyecto.

Algunas propuestas de los antecedentes del Capítulo II que consideraron a los comentarios fueron modelos Seq2seq o LDA. Particularmente, el Antecedente 17 planteó una arquitectura de este último tipo para resolver el problema de clasificación que encajaba con el marco de trabajo de la actual tesis de investigación, aplicando segmentación de comentarios según el tema de su contenido para luego alimentar a su sistema de recomendación de proyectos. Sin embargo, como se especifica en las especificaciones de requerimientos para el desarrollo de la experimentación en su Capítulo 5, se necesitó tener al menos una memoria RAM de 32 GB y una GPU potente como Nvidia GForce 1080 para llevar a cabo los experimentos con más de 504 mil comentarios filtrados provenientes de 600 proyectos de Kickstarter ([Shafqat & Byun, 2019](#)). Esto resultó inviable para las condiciones presentes en el entorno ya que, si bien la suscripción a Google Colab Pro permite utilizar GPU con hasta aproximadamente 26 GB de memoria, el conjunto recolectado de comentarios representó más de 10 veces (7,865 proyectos con comentarios) la cantidad mencionada con un total de más de 494 mil comentarios, como se detalla en el segundo subcapítulo del Capítulo IV. Ante este escenario, se optó por la opción de un modelo LSTM Bidireccional, acortando el número de palabras del total de comentarios por proyecto a un valor estándar para poder diseñar la capa de incrustación correspondiente.

A nivel individual, The Hydra bajo cada métrica (desde las más usadas como la exactitud hasta aquellas más recomendadas para problemas con data desbalanceada como el puntaje F1) mantuvo niveles parejos y llevó sin problemas su entrenamiento, pese a que tanto el modelo de descripción como de comentarios se obstaculizaron con el sobreajuste luego de muchas épocas. Entre una de las razones por las cuales ambos modelos no progresaban luego de una avanzada cantidad de épocas se encontró en el contenido textual, en especial, el de comentarios ya que la interacción social muchas veces no está sujeta a estrictas normativas de la gramática hacia los usuarios que expresan libremente su opinión. Por lo tanto, algunas palabras incorrectamente redactadas no pudieron ser lematizadas al 100% por la librería NLTK. El modelo de metainformación, en cambio, ayudó a mejorar el rendimiento del modelo apilado considerablemente, el cual se ve reflejado en la aproximación de sus resultados con este último.

A pesar de presentarse una data desbalanceada (72% proyectos fracasados y 28% exitosos), separar en subconjuntos de entrenamiento y pruebas estratificadas, y luego previo a la creación de cada modelo balancear los pesos de las clases (0.6987077585764833 para proyectos fracasados y 1.7581290322580645 para exitosos) fueron también determinantes para que los modelos no solo caigan en sobreajuste tempranamente, sino también presenten niveles de exactitud y pérdida en la validación muy similares al entrenamiento.

6.2. Recomendaciones

El próximo paso será crear una plataforma web que integre tanto el sistema prototipo descrito en la última sección del Capítulo V, que contiene al modelo The Hydra, como una interfaz más amigable y sencilla de utilizar para el usuario que requiera evaluar en tiempo real su proyecto desde esta solución. Esta idea fue motivada por la extensión en Google Chrome que los autores del Antecedente 1 realizaron ([K. Chen y col., 2013](#)).

Para afinar el desarrollo y los resultados de esta investigación, se recomienda comenzar con continuar ajustando los hiperparámetros de los modelos de contenido textual para progresar en la etapa de entrenamiento, buscando otras alternativas de optimizadores (por ejemplo, *RMS-Prop*, *SGD*) o alterar más parámetros de la opción usada *Adam*, usando otros inicializadores de kernel, entre otros.

En caso se cuente con herramientas tecnológicas más potentes de hardware y software para el desarrollo de modelos predictivos más profundos como modelos Seq2seq, multimodales o híbridos del tipo DC-LDA, se sugiere de todas maneras limitar la extensión de palabras a un valor no mayor a la cantidad de comentarios del Antecedente 17 ([Shafqat & Byun, 2019](#)).

Como trabajos a futuro se plantea construir modelos, tanto de Aprendizaje Profundo como de Aprendizaje Automático, considerando otras variables como actualizaciones del proyecto (contenido textual) e interacción social externa entre el creador y la comunidad que menciona o toma interés al proyecto en redes sociales. Otras alternativas potenciales también figuran la de predicción de las mejores opciones de valores que deberían contener las variables (tanto cuantitativas como cualitativas) para un proyecto antes del lanzamiento de su campaña. Este enfoque permitiría evaluar indefinidamente la información que un creador asigne a su campaña hasta encontrar la mejor combinación gracias a un modelo de recomendación.

Finalmente, como en varias referencias y libros sobre Aprendizaje Automático y Aprendizaje Profundo que se pueden encontrar, no existe una regla definida para asegurar el rendimiento excelente de cualquier modelo. El factor del logro de objetivos principalmente se debe a la continua experimentación y uso de distintas técnicas para alcanzar la performance esperada.

BIBLIOGRAFÍA

- A Not So Random Walk. (2019). Backpropagation Example With Numbers Step by Step. <https://www.anotsorandomwalk.com/backpropagation-example-with-numbers-step-by-step/>
- Alpaydin, E. (2014). *Introduction to Machine Learning* (Tercera edición). MIT Press.
- Asociación de Emprendedores de Perú. (2018). Avances y limitaciones del emprendimiento peruano. <https://asep.pe/index.php/avances-limitaciones-emprendimiento-peruano/>
- BBVA OpenMind. (2019). ¿Qué es el aprendizaje Profundo? (A. Banafa, Ed.). <https://www.bbvaopenmind.com/tecnologia/mundo-digital/que-es-el-aprendizaje-profundo/>
- Beckwith, J. (2016). Predicting Success in Equity Crowdfunding. *Joseph Wharton Scholars*. http://repository.upenn.edu/joseph_wharton_scholars/25
- Bertona, L. F. (2005). *Entrenamiento de Redes Neuronales basado en Algoritmos Evolutivos* (Tesis de grado). Universidad de Buenos Aires. Buenos Aires, Argentina. <http://laboratorios.fi.uba.ar/lisi/bertona-tesisingenieriainformatica.pdf>
- Betancourt, G. A. (2005). Las Máquinas de Soporte Vectorial (SVMs). *Scientia et Technica*. <https://revistas.utp.edu.co/index.php/revistaciencia/article/view/6895/4139>
- Braulio Gil, N. & Curto Díaz, J. (2015). *Customer Analytics: Mejorando la inteligencia del cliente a través de los datos* (Reporte técnico). Universitat Oberta de Catalunya. Barcelona, España. <https://docplayer.es/17897069-Customer-analytics-mejorando-la-inteligencia-del-cliente-a-traves-de-los-datos-jordi-conesa-i-caralt-coordinador-nuria-braulio-gil-josep-curto-diaz.html>
- Britos, P. V., García Martínez, R., Hossian, A. & Sierra, E. (2006). *Minería de Datos*. Nueva Librería.
- Brownlee, J. (2017a). *Deep Learning for Natural Language Processing: Develop Deep Learning Models for your Natural Language Problems*. Machine Learning Mastery. http://ling.snu.ac.kr/class/AI_Agent/deep_learning_for_nlp.pdf
- Brownlee, J. (2017b). *How to Develop a Bidirectional LSTM For Sequence Classification in Python with Keras*. <https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/>

- Brownlee, J. (2018). *Stacking Ensemble for Deep Learning Neural Networks in Python*. <https://machinelearningmastery.com/stacking-ensemble-for-deep-learning-neural-networks/>
- Brownlee, J. (2019). *How to Fix the Vanishing Gradients Problem Using the ReLU*. <https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/>
- Bujokas, E. (2020). *Creating Word Embeddings: Coding the Word2Vec Algorithm in Python using Deep Learning*. <https://towardsdatascience.com/creating-word-embeddings-coding-the-word2vec-algorithm-in-python-using-deep-learning-b337d0ba17a8>
- Calvo, D. (2017). Clasificación de redes neuronales artificiales. <http://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/>
- Calvo, D. (2018a). Definición de Red Neuronal Recurrente. <http://www.diegocalvo.es/red-neuronal-recurrente/>
- Calvo, D. (2018b). Función de activación - Redes neuronales. <http://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/>
- Chaichi, N. & Anderson, T. (2019). Deploying Natural Language Processing to Extract Key Product Features of Crowdfunding Campaigns: The Case of 3D Printing Technologies on Kickstarter, 1-9. <https://doi.org/10.23919/PICMET.2019.8893839>
- Chen, K., Jones, B., Kim, I. & Schlamp, B. (2013). *KickPredict: Predicting Kickstarter Success* (Reporte técnico). Instituto de Tecnología de California. California, Estados Unidos de América. <http://courses.cms.caltech.edu/cs145/2013/blue.pdf>
- Chen, L.-S. & Shen, E.-L. (2019). Finding the Keywords Affecting the Success of Crowdfunding Projects, 567-571. <https://doi.org/10.1109/IEA.2019.8714815>
- Chen, S.-Y., Chen, C.-N., Chen, Y.-R., Yang, C.-W. & Lin, W.-C. (2015). Will Your Project Get the Green Light? Predicting the Success of Crowdfunding Campaigns. <http://aisel.aisnet.org/pacis2015/79>
- Cheng, C., Tan, F., Hou, X. & Wei, Z. (2019). Success Prediction on Crowdfunding with Multimodal Deep Learning, 2158-2164. <https://www.ijcai.org/proceedings/2019/0299.pdf>
- Chengwei. (2018). *Simple Stock Sentiment Analysis with news data in Keras*. <https://www.dlogy.com/blog/simple-stock-sentiment-analysis-with-news-data-in-keras/>
- Colgren, D. (2014). The rise of crowdfunding: Social media, big data, cloud technologies. *Strategic Finance*, 95(10), 56.
- Collins, L. (2014). *Crowdfunding: Innovative access to finance and regulatory challenges*.
- Cyberclic. (s.f.). ¿Qué es una campaña publicitaria? <https://www.cyberclick.es/publicidad/campana-publicitaria>
- Deng, L. & Liu, Y. (2018). *Deep Learning in Natural Language Processing*. Springer. <https://doi.org/10.1007/978-981-10-5209-5>

- Dorofki, M., Elshafie, A. H., Jaafar, O., Karim, O. A. & Mastura, S. (2012). Comparison of Artificial Neural Network Transfer Functions Abilities to Simulate Extreme Runoff Data. En IPCBEE (Ed.). IACSIT Press. <http://www.ipcbee.com/vol33/008-ICEEB2012-B021.pdf>
- Fernández-Bedoya, V. H., Gago-Chávez, J. d. J. S., Meneses-la-Riva, M. E. & Suyo-Vega, J. A. (2020). Collaborative Economy in Peru: Past, Present and Future. *Path of Science*, 6(5), 7001-7006. <https://doi.org/10.22178/pos.58-5>
- Figueroa M., G. (s.f.). Convolución y transformadas. *Revista digital Matemática*. <https://tecdigital.tec.ac.cr/revistamatematica/cursos-linea/EcuacionesDiferenciales/EDO-Geo/edo-cap5-geo/laplace/node7.html>
- Gandhi, R. (2018). *Support Vector Machine — Introduction to Machine Learning Algorithms*. <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>
- Gartner. (2019a). Gartner IT Glossary - Machine Learning. <https://www.gartner.com/it-glossary/machine-learning/>
- Gartner. (2019b). Gartner IT Glossary - Predictive Modeling. <https://www.gartner.com/it-glossary/predictive-modeling/>
- Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing* (G. Hirst, Ed.). Morgan & Claypool Publishers. <https://doi.org/10.2200/S00762ED1V01Y201703HLT037>
- González, L. (2019). Curvas ROC y Área bajo la curva (AUC). <http://ligdigonzalez.com/curvas-roc-y-area-bajo-la-curva-auc-machine-learning/>
- Google Cloud. (s.f.). Programa gratuito de Google Cloud. <https://cloud.google.com/free/docs/gcp-free-tier>
- Google Developers. (2018). Glosario sobre aprendizaje automático. <https://developers.google.com/machine-learning/glossary/?hl=es-419>
- Gupta, P. (2017). *Decision Trees in Machine Learning*. <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>
- Hamdaoui, Y. (2019). *TF(Term Frequency)-IDF(Inverse Document Frequency) from scratch in python*. <https://towardsdatascience.com/tf-term-frequency-idf-inverse-document-frequency-from-scratch-in-python-6c2b61b78558>
- Hollas, J. (2013). Is crowdfunding now a threat to traditional finance? *Corporate Finance Review*, 18(1), 27.
- House of Bots. (2018). Most Popular 20 Free Online Courses to Learn Deep Learning (K. Cook, Ed.). <https://www.houseofbots.com/news-detail/3620-4-most-popular-20-free-online-courses-to-learn-deep-learning>
- IArtificial.net. (2019a). Matemáticas de la Regresión Logística. <https://iartificial.net/regresion-logistica-para-clasificacion/>

- IArtificial.net. (2019b). Método del Gradiente Descendiente. <https://iartificial.net/gradiente-descendiente-para-aprendizaje-automatico/>
- IBM. (2019). Regresión Logística. https://www.ibm.com/support/knowledgecenter/es/SSLVMB_sub/statistics_mainhelp_ddita/spss/regression/idh_lreg.html
- Inzaugarat, E. (2018). *Understanding Neural Networks: What, How and Why?* <https://towardsdatascience.com/understanding-neural-networks-what-how-and-why-18ec703ebd31>
- Izco, F. (2018). Base de Datos Corporativa. https://bookdown.org/f_izco/BDC-POC/metricas.html
- Jin, B., Zhao, H., Chen, E., Liu, Q. & Ge, Y. (2019). Estimating the Days to Success of Campaigns in Crowdfunding: A Deep Survival Perspective. http://staff.ustc.edu.cn/~cheneh/paper_pdf/2019/Binbin-Jin-AAAI.pdf
- Kamath, R. S. & Kamat, R. K. (2018). Supervised Learning Model For Kickstarter Campaigns With R Mining. *International Journal of Information Technology, Modeling and Computing (IJITMC)*, 4(1). <https://doi.org/10.5281/zenodo.1228716>
- Kamath, U., Liu, J. & Whitaker, J. (2019). *Deep Learning for NLP and Speech Recognition*. Springer.
- Kaur, H. & Gera, J. (2017). Effect of Social Media Connectivity on Success of Crowdfunding Campaigns. *Procedia Computer Science*, 122, 767-774. <https://doi.org/10.1016/j.procs.2017.11.435>
- Kickstarter. (s.f.-a). Acerca de nosotros: Kickstarter. <https://www.kickstarter.com/about?ref=global-footer>
- Kickstarter. (s.f.-b). Create something to share with others. Intro to Kickstarter for designers, makers and technologists (Diapositivas de PowerPoint).
- Kickstarter. (s.f.-c). Empieza tu proyecto. <https://www.kickstarter.com/learn?lang=es>
- Kickstarter. (s.f.-d). Financiamiento: Kickstarter. <https://www.kickstarter.com/help/handbook/funding?lang=es>
- Kickstarter. (s.f.-e). Nuestras normas. https://www.kickstarter.com/rules?ref=learn_faq
- Kickstarter. (s.f.-f). Prensa: Kickstarter. <https://www.kickstarter.com/press?ref=hello>
- Kickstarter. (2021). Revopoint POP: Precise 3D Scanner for 3D printing (Revopoint 3D, Ed.). <https://www.kickstarter.com/projects/2125914059/revopoint-pop-high-precision-3d-scanner-for-3d-printing>
- Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746-1751. <https://doi.org/10.3115/v1/D14-1181>
- Kohavi, R. & Provost, F. (1998). Glossary of Terms Journal of Machine Learning. <http://ai.stanford.edu/~ronnyk/glossary.html>

- Kostadinov, S. (2019). *Understanding Encoder-Decoder Sequence to Sequence Model*. <https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346>
- Kraus, S., Richter, C., Brem, A., Cheng, C.-F. & Chang, M.-L. (2016). Strategies for reward-based crowdfunding campaigns. *Journal of Innovation & Knowledge*, 1, 13-23. <https://doi.org/10.1016/j.jik.2016.01.010>
- Lee, H.-D. (2019). Factors affecting successful crowdfunding. *ACM International Conference Proceeding Series*, 379-382. <https://doi.org/10.1145/3306500.3306524>
- Lee, I. & Shin, Y. (2018). Fintech: Ecosystem, business models, investment decisions, and challenges. *Business Horizons*, 61(1), 35-46.
- Lee, S., Lee, K. & Kim, H.-c. (2018). Content-based Success Prediction of Crowdfunding Campaigns: A Deep Learning Approach. *Companion of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing*, 193-196. <https://doi.org/10.1145/3272973.3274053>
- Li, F.-F., Johnson, J. & Yeung, S. (2019). *Convolutional Neural Networks* (Reporte técnico). Universidad Standford. http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture05.pdf
- Li, Y., Rakesh, V. & Reddy, C. K. (2016). Project Success Prediction in Crowdfunding Environments. *Ninth ACM International Conference on web search and data mining*, 247-256. <https://doi.org/10.1145/2835776.2835791>
- Lichtig, B. (2015). Crowdfunding Success: The Short Story - Analyzing the Mix of Crowd-funded Ventures. *Wharton Research Scholars*, 121. https://repository.upenn.edu/wharton_research_scholars/121/
- López Briega, R. E. (2016). Redes neuronales convolucionales con TensorFlow. <https://relopezbrriegar.github.io/blog/2016/08/02/redes-neuronales-convolucionales-con-tensorflow/>
- López-Golán, M., Vaca Tapia, A. C., Benavides García, N. & Coronado Otavallo, X. M. (2017). Crowdfunding campaigns. Its effectiveness in audiovisual projects in the Latin American context — Las campañas de crowdfunding. Su eficacia en proyectos audiovisuales en el contexto latinoamericano. *Iberian Conference on Information Systems and Technologies, CISTI*, 1-6. <https://doi.org/10.23919/CISTI.2017.7976016>
- Machine Learning for Artists. (2019). Redes Neuronales. https://ml4a.github.io/ml4a/es/neural_networks/
- Maimon, O. & Rokach, L. (2010). *Data Mining and Knowledge Discovery Handbook* (Primera edición). Springer.
- Malik, U. (2019). *Python for NLP: Movie Sentiment Analysis using Deep Learning in Keras*. <https://stackabuse.com/python-for-nlp-movie-sentiment-analysis-using-deep-learning-in-keras/>

- MathWorks. (s.f.). Máquina de vectores de soporte (SVM). <https://la.mathworks.com/discovery/support-vector-machine.html>
- Merino, M. (2019). Conceptos de inteligencia artificial: qué es el aprendizaje por refuerzo. <https://www.xataka.com/inteligencia-artificial/conceptos-inteligencia-artificial-que-aprendizaje-refuerzo>
- Microsoft. (2018). Algoritmos de minería de datos (Analysis Services: Minería de datos). <https://docs.microsoft.com/es-mx/sql/analysis-services/data-mining/data-mining-algorithms-analysis-services-data-mining?view=sql-server-2017>
- Microsoft. (2019). Conceptos de minería de datos. <https://docs.microsoft.com/es-es/sql/analysis-services/data-mining/data-mining-concepts?view=sql-server-2017>
- MissingLink.ai. (s.f.). *Keras Conv1D: Working with 1D Convolutional Neural Networks in Keras*. <https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/>
- Mitra, T. & Gilbert, E. (2014). The Language that Gets People to Give: Phrases that Predict Success on Kickstarter, 49-61. <https://doi.org/10.1145/2531602.2531656>
- Molina Arias, M. & Ochoa Sangrador, C. (2017). Pruebas diagnósticas con resultados continuos o politómicos. Curvas ROC. *Evidencias en Pediatría*, 13, 12. http://archivos.evidenciasenpediatria.es/files/41-13133-RUTA/Fundamentos_MBE_12.pdf
- Mukaka, M. M. (2012). Statistics Corner: A guide to appropriate use of Correlation coefficient in medical research. *Malawi Medical Journal*, 24(3), 69-71. https://www.researchgate.net/publication/236604665_Statistics_Corner_A_guide_to_appropriate_use_of_Correlation_coefficient_in_medical_research
- Ng, A. (2018). *RNN W1L11 : Bidirectional RNN* [Archivo de video]. <https://www.youtube.com/watch?v=bTXGpATdKRY>
- Nishida, N. & Nakayama, H. (2015). *Multimodal gesture recognition using multi-stream recurrent neural network* (Reporte técnico). Universidad de Tokio. <http://www.nlab.ci.i.u-tokyo.ac.jp/projects/deeplearn-e.html>
- Pardo, C. J. (s.f.). MACHINE LEARNING – Regresión Logística. <https://carlosjuliopardoblog.wordpress.com/2017/12/31/regresion-logistica/>
- Pennington, J., Socher, R. & Manning, C. D. (2014a). *GloVe: Global Vectors for Word Representation* (Reporte técnico). Universidad Standford. California, Estados Unidos de América. <https://nlp.stanford.edu/pubs/glove.pdf>
- Pennington, J., Socher, R. & Manning, C. D. (2014b). *GloVe: Global Vectors for Word Representation*. <https://nlp.stanford.edu/projects/glove/>
- Phi, M. (2018). *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

- Poole, D., Mackworth, A. & Goebel, R. (1998). Computational Intelligence: A Logical Approach. *Oxford University Press*.
- Prabhu, R. (2018). *Understanding of Convolutional Neural Network (CNN) — Deep Learning*. <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>
- Project Management Institute. (2017). *A guide to the project management body of knowledge (PMBOK guide)* (Sexta edición). PMBOK.
- Puente, A. (2019). *Predicción del estado de financiamiento de proyectos de tecnología en web de crowdfunding Kickstarter mediante modelo(s) de Aprendizaje Automático* (Tesis de pregrado). Universidad ESAN. Lima, Perú.
- Ranjan, C. (2019). *Rules-of-thumb for building a Neural Network*. <https://towardsdatascience.com/17-rules-of-thumb-for-building-a-neural-network-93356f9930af>
- Rao, D. & McMahan, B. (2019). *Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning*. O'Reilly.
- Real Academia Española. (s.f.). Hidra. <https://dle.rae.es/hidra>
- Real Academia Española. (2020). Diccionario de la lengua española (Vigesimotercera edición). <https://dle.rae.es/>
- Redacción Gestión. (2015). Emprendimiento en el Perú se origina más por oportunidades de negocio que por desempleo. *Diario Gestión*. <https://gestion.pe/economia/emprendimiento-peru-origina-oportunidad-negocio-desempleo-80578>
- Redacción Gestión. (2018). Perú es el tercer país con mayor cantidad de emprendimientos en fase temprana a nivel mundial. *Diario Gestión*. <https://gestion.pe/economia/peru-tercer-pais-mayor-cantidad-emprendimientos-fase-temprana-nivel-mundial-240264>
- Russell, S. & Norvig, P. (2004). *Inteligencia Artificial: Un Enfoque Moderno* (J. M. Corchado Rodríguez, F. Martín Rubio, J. M. Cadenas Figueredo, L. D. Hernández Molinero, E. Paniagua Arís, R. Fuentetaja Pinzán, M. Robledo de los Santos & R. Rizo Aldeguer, Trad.; Segunda edición). Pearson Educación, S.A. <https://luismejias21.files.wordpress.com/2017/09/inteligencia-artificial-un-enfoque-moderno-stuart-j-russell.pdf>
- Russell, S. & Norvig, P. (2009). *Inteligencia Artificial: Un Enfoque Moderno* (Tercera edición). Prentice Hall.
- Sancho Caparrini, F. (2017). *Entrenamiento de Redes Neuronales: mejorando el Gradiente Descendiente* (Reporte técnico). Universidad de Sevilla. Sevilla, España. <http://www.cs.us.es/~fsancho/?e=165>
- Sancho Caparrini, F. (2018). *Clasificación Supervisada y No Supervisada* (Reporte técnico). Universidad de Sevilla. Sevilla, España. <http://www.cs.us.es/~fsancho/?e=77>
- Sandoval, L. (s.f.). Barreras del Emprendedor ¿Por qué cuesta tanto hacerlo? <https://www.emprender-facil.com/es/barreras-del-emprendedor/>

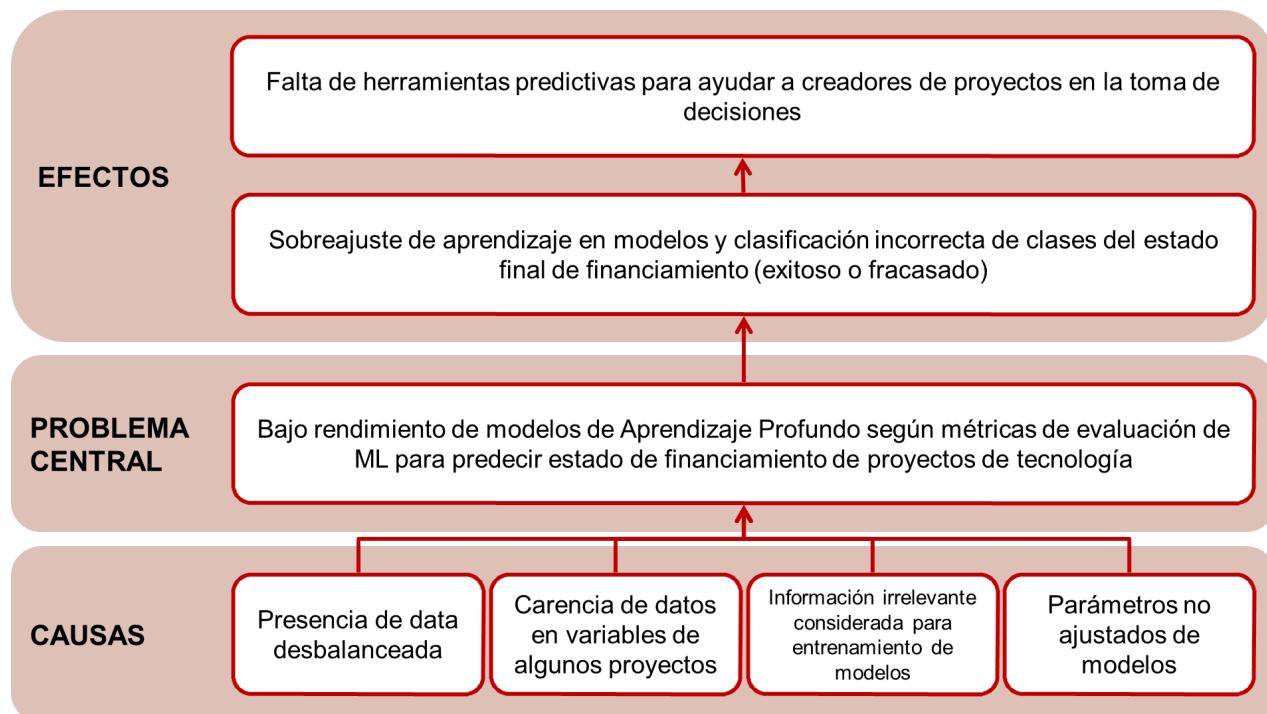
- SAS Institute. (s.f.). ¿Qué es Deep Learning? https://www.sas.com/es_pe/insights/analytics/deep-learning.html
- Sawhney, K., Tran, C. & Tuason, R. (2016). *Using Language to Predict Kickstarter Success* (Reporte técnico). Universidad Standford. California, Estados Unidos de América. <https://stanford.edu/~kartiks2/kickstarter.pdf>
- Shafique, U. & Qaiser, H. (2014). A Comparative Study of Data Mining Process Models (KDD, CRISP-DM and SEMMA). *International Journal of Innovation and Scientific Research*, 12(1), 217-222. <http://www.ijisr.issr-journals.org/abstract.php?article=IJISR-14-281-04>
- Shafqat, W. & Byun, Y.-C. (2019). Topic Predictions and Optimized Recommendation Mechanism Based on Integrated Topic Modeling and Deep Neural Networks in Crowdfunding Platforms. *Applied Sciences*, 9(24), 5496. <https://doi.org/10.3390/app9245496>
- SitioBigData.com. (2019a). Machine Learning: Selección Métricas de clasificación. <http://sitiobigdata.com/2019/01/19/machine-learning-metrica-clasificacion-parte-3/#>
- SitioBigData.com. (2019b). ReLU: Funciones de activación. <http://sitiobigdata.com/2019/06/22/relu-funciones-activacion/>
- Smola, A. J. & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3), 199-222. <https://link.springer.com/article/10.1023/B:STCO.0000035301.49549.88>
- Solidaridad Latina. (s.f.). ¿Cómo funciona el crowdfunding en Latinoamérica? <https://solidaridadlatina.com/actualizacion/como-funciona-crowdfunding-latinoamerica/>
- Stokes, K., Clarence, E., Anderson, L. & Rinne, A. (2014). *Making sense of the UK collaborative economy*. http://collaboriamo.org/media/2014/10/making_sense_of_the_uk_collaborative_economy_14.pdf
- Sutton, R. S. & Barto, A. G. (2018). Finite Markov Decision Processes. *Reinforcement Learning: An Introduction* (Segunda edición, p. 48). MIT Press. <http://incompleteideas.net/book/RLbook2018.pdf>
- The Hustle. (2019). What are your chances of successfully raising money on Kickstarter? <https://thehustle.co/crowdfunding-success-rate>
- Tung, F.-W. & Liu, X.-Y. (2018). Understanding backers' motivations and perceptions of information on product-based crowdfunding platforms, 84-88. <https://doi.org/10.1109/ISCBI.2018.00026>
- Ullah, S. & Zhou, Y. (2020). Gender, Anonymity and Team: What Determines Crowdfunding Success on Kickstarter. *Journal of Risk and Financial Management*, 13(4), 80. <https://doi.org/10.3390/jrfm13040080>
- Universo Crowdfunding. (s.f.). ¿Qué es el crowdfunding? <https://www.universocrowdfunding.com/que-es-el-crowdfunding/>

- Viera Balanta, V. (2013). *Backpropagation explicación*. <https://www.youtube.com/watch?v=0odQ286nsIY>
- Web Robots. (2019). Kickstarter Datasets. <https://webrobots.io/kickstarter-datasets/>
- Xuefeng, L. & Zhao, W. (2018). Using Crowdfunding in an Innovative Way: A Case Study from a Chinese Crowdfunding Platform, 1-9. <https://doi.org/10.23919/PICMET.2018.8481838>
- Yu, A. (2017). *The Complete Crowdfunding Course for Kickstarter & Indiegogo* (Diapositivas de PowerPoint). Udemy.
- Yu, P.-F., Huang, F.-M., Yang, C., Liu, Y.-H., Li, Z.-Y. & Tsai, C.-H. (2018). Prediction of Crowdfunding Project Success with Deep Learning, 1-8. <https://doi.org/10.1109/ICEBE.2018.00012>
- Yuan, H., Lau, R. Y. & Xu, W. (2016). The Determinants of Crowdfunding Success: A Semantic Text Analytics Approach. *Decision Support Systems*, 91, 67-76. <https://doi.org/10.1016/j.dss.2016.08.001>
- Yuan, X., Li, L. & Wang, Y. (2019). Nonlinear Dynamic Soft Sensor Modeling With Supervised Long Short-Term Memory Network. *IEEE Transactions on Industrial Informatics*, 1-1. <https://doi.org/10.1109/TII.2019.2902129>
- Zambrano, J. (2018). *¿Aprendizaje supervisado o no supervisado? Conoce sus diferencias dentro del machine learning y la automatización inteligente*. <https://medium.com/@juanzambrano/aprendizaje-supervisado-o-no-supervisado-39ccf1fd6e7b>
- Zhang, Y. & Wallace, B. C. (2017). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *Proceedings of the The 8th International Joint Conference on Natural Language Processing*, 253-263. <https://www.aclweb.org/anthology/I17-1026.pdf>
- Zhou, M., Zhang, X., Wang, A. G., Du, Q., Qiao, Z. & Fan, W. (2015). Money Talks: A Predictive Model on Crowdfunding Success Using Project Description. *21st Americas Conference on Information Systems, AMCIS 2015, Puerto Rico, August 13-15, 2015*. <http://aisel.aisnet.org/amcis2015/BizAnalytics/GeneralPresentations/37>
- Zhou, V. (2019). *A Simple Explanation of the Bag-of-Words Model*. <https://towardsdatascience.com/a-simple-explanation-of-the-bag-of-words-model-b88fc4f4971>
- Zimovnov, A. (2018). *NLP - Text Preprocessing and Text Classification (using Python)* [Archivo de video]. <https://youtu.be/nxhCyeRR75Q>

Anexos

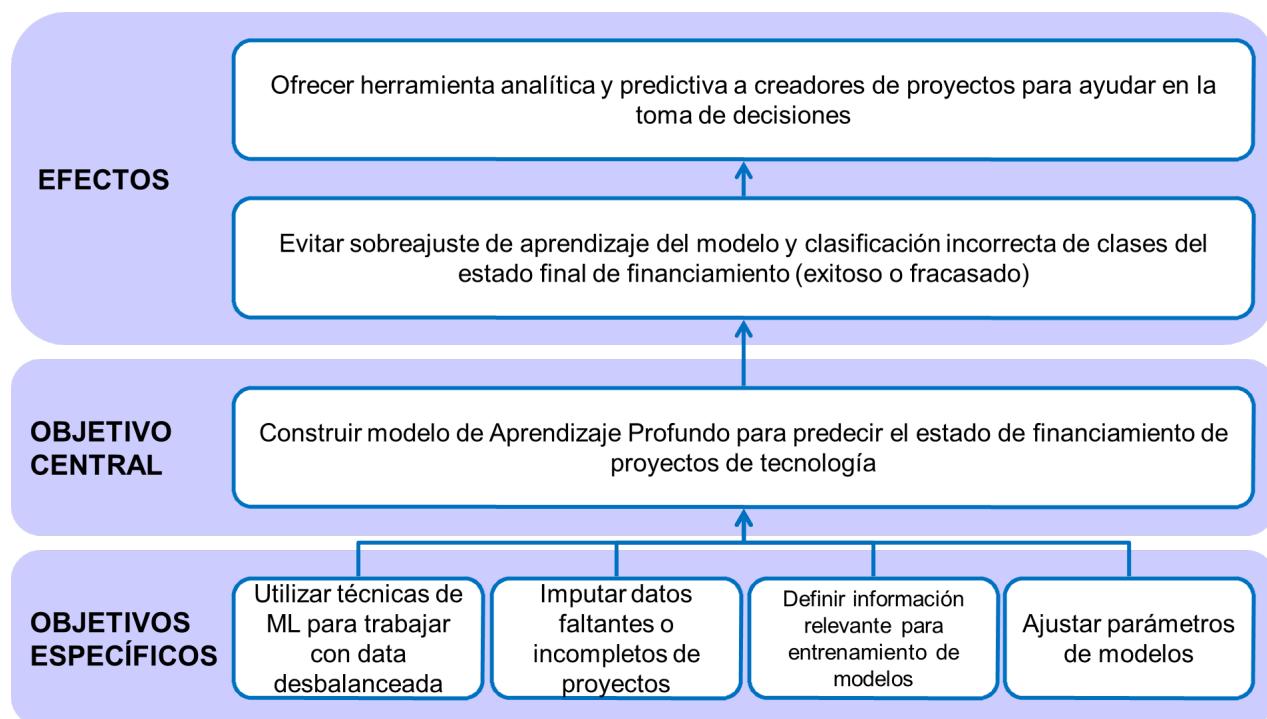
Anexo A

Árbol de Problemas



Anexo B

Árbol de Objetivos



Anexo C

Matriz de Consistencia

PROBLEMAS	OBJETIVOS	HIPÓTESIS
Problema General	Objetivo General	Hipótesis General
Problemas Específicos	Objetivos Específicos	Hipótesis Específicas
Bajo rendimiento de modelos de Aprendizaje Profundo según métricas de evaluación de Machine Learning para predecir estado de financiamiento de proyectos de tecnología.	Construir modelo de Aprendizaje Profundo para predecir éxito de financiamiento de proyectos de tecnología.	El modelo propuesto de Aprendizaje Profundo predice el estado de financiamiento de proyectos de tecnología.
Presencia de data desbalanceada.	Utilizar técnicas de Machine Learning para trabajar con data desbalanceada.	Las técnicas de Machine Learning ayudan a los modelos a trabajar con data desbalanceada.
Carencia de datos en variables de algunos proyectos.	Imputar datos faltantes o incompletos de proyectos.	Los datos faltantes o incompletos de proyectos son imputados.
Información irrelevante considerada para entrenamiento de modelos.	Definir información relevante para entrenamiento de modelos.	Se considera solo información relevante para entrenamiento de modelos.
Parámetros no ajustados de modelos.	Ajustar parámetros de modelos.	Los parámetros de los modelos están ajustados.
Sobreajuste de aprendizaje de modelos y clasificación incorrecta de clases del estado final de financiamiento (exitoso o fracasado).	Evitar sobreajuste de aprendizaje del modelo y clasificación incorrecta de clases del estado final de financiamiento (exitoso o fracasado).	Se evita el sobreajuste de aprendizaje del modelo y éste clasifica correctamente las clases del estado final de financiamiento (exitoso o fracasado).
Falta de herramientas predictivas para ayudar a creadores de proyectos en la toma de decisiones.	Ofrecer herramienta analítica y predictiva a creadores de proyectos para ayudar en la toma de decisiones.	Se ofrece herramienta analítica y predictiva para ayudar a los creadores de proyectos en la toma de decisiones.