Universidad Nacional de San Juan

FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

DEPARTAMENTO DE INFORMÁTICA

TRABAJO FINAL

Ciencia de Datos en la Predicción del Fenómeno Climático de la Helada, centrado en su intensidad, duración y alternativas de mitigación.

Licenciatura en Ciencias de la Computación

Autor: Joaquín José Cortez Robles – DNI: 41774731

Asesor: Lic. María Isabel Masanet

Co-Asesor: Mg. Ing. Raúl Klenzi

Co-Asesor: Lic. Manuel Ortega

San Juan, Argentina

2023

Índice de contenidos

1. Introducción

1.1. Resumen

Las heladas son una de las problemáticas que afectan a los agroproductores y ocasionan grandes pérdidas económicas. En este trabajo se plantea resolver el problema de la predicción de ocurrencia de heladas, y sus características de intensidad y duración, utilizando métodos de Ciencia de Datos. En torno a esto, se exploró y profundizó sobre los conceptos de Ciencia de Datos y su proceso, Inteligencia Artificial (IA), Machine Learning (ML), Deep Learning (DL), computación paralela y, por su puesto, el fenómeno climático de la helada.

Para el desarrollo de este trabajo, se utilizaron datos captados por sensores agro meteorológicos dispuestos en dos estaciones meteorológicas ubicadas en la provincia de San Juan entre los años 2013 y 2019. Se utilizó el lenguaje de programación Python, con diferentes bibliotecas, así como también Knime Analytics (KA) -Konstanz Information Miner- una plataforma de minería de datos que permite el desarrollo de modelos en un entorno visual, para la limpieza, procesamiento, modelado y visualización de los datos.

La predicción de ocurrencia de heladas, se realizó con la estrategia de ventanas de tiempo, la cual fue abordada por algoritmos de *Machine Learning* y *Deep Learning*. Por otro lado, se trabajó en las características de la helada, así como su ocurrencia, mediante agrupación por días de los datos, y su predicción mediante algoritmos de *Machine Learning*.

Los algoritmos de *Machine Learning* utilizados fueron Random Forest, clasificador y regresor, y regresión logística y lineal con diferentes hiperparámetros. Por otro lado, en cuanto a *Deep Learning* se utilizaron redes neuronales *FeedForward* y LSTM (Long Short Term Memory, del inglés Memoria de Corto-Largo Plazo), las cuales fueron comparadas tanto con métricas de predicciones como en tiempos de ejecución, en CPU (Central Processing Unit) y GPU (Graphic Processing Unit).

Se lograron buenos resultados prediciendo la ocurrencia de helada, en especial con Random Forest y también mediante la estrategia de ventanas de tiempo con redes neuronales LSTM. También se logró estimar la intensidad y duración de heladas antes de que estas ocurran, en especial en sus categorías más severas. Se compararon los resultados en los tiempos de ejecución trabajando redes neuronales sobre CPU y GPU y se encontró que existe una gran reducción en los tiempos en caso de entrenar las redes neuronales sobre GPU.

1.2. Motivación

Una de las problemáticas que actualmente conciernen a los agroproductores de la provincia de San Juan, es la ocurrencia de heladas y particularmente las heladas tardías en meses de octubre y noviembre, que afecta a los cultivos en plena floración llegando a ocasionar pérdidas totales. Poder predecir esta condición climática y con ello alertar al productor a efectos de que se tomen las medidas de mitigación y así atenuar o evitar pérdidas, es un aporte relevante que se realiza en este trabajo final.

En el caso de la predicción de heladas es importante destacar el momento en que se producirá y la duración e intensidad de la misma a efectos de ajustar adecuadamente las diferentes instancias de mitigación.

Otra consideración importante a la hora de tener en cuenta la temática del presente trabajo, es que el comportamiento de las heladas varía según la localidad en la que se produzcan, de forma que los modelos de predicción realizados con datos de otras zonas del mundo no resultarán útiles para predecir la ocurrencia de heladas en la provincia de San Juan.

1.3. Problema a resolver

El problema que se plantea resolver en el desarrollo de este trabajo es el de la predicción de la ocurrencia de heladas, centrado en sus características de duración e intensidad.

1.4. Propuesta de solución

La propuesta de solución que se presenta al problema mencionado es utilizar técnicas de Ciencia de Datos sobre datos provenientes de zonas agroproductivas de la provincia de San Juan, para lograr un mayor entendimiento de la problemática desde un punto de vista analítico. De esta forma, se podrán aplicar algoritmos de aprendizaje de máquina a fin de llegar a predecir la ocurrencia de la helada junto con sus características.

1.5. Objetivos

1.5.1. Objetivo general

- Determinación de intensidad y duración, en instancia de predicción, del fenómeno climático de la helada.

1.5.2. Objetivos específicos

- Utilizar estrategias de Deep Learning que permitan modelar las series temporales tratadas.
- Visualizar y comparar los mejores parámetros estadísticos que permitan determinar la calidad de los modelos de Machine learning (ML) utilizados.
- Habilitar la configuración de Knime Analytics que permita la utilización de la placa de video del tipo Nvidia que admita la paralelización de los modelos de ML utilizados.

2. Antecedentes

2.1. Trabajos previos

La presente propuesta surge en el marco de trabajo compuesto por los proyectos de investigación llevados adelante en el ámbito del Departamento e Instituto de Informática de la Facultad de Ciencias Exactas Físicas y Naturales DI-IdeI-FCEFN y en el Instituto de Automática de la Facultad de Ingeniería INAUT-FI de la Universidad Nacional de San Juan UNSJ.

Uno de los proyectos involucrados "Evaluación de visualizaciones eficientes en Ciencia de Datos" comenzó su tercer año de ejecución, al momento de comenzar el presente trabajo, en el ámbito del Laboratorio de Sistemas Inteligentes para Extracción de Conocimiento en Datos Masivos DI-IdeI-FCEFyN vinculando sinérgicamente el accionar de docentes investigadores y alumnos pertenecientes a las carreras de la Licenciatura en Sistemas de Información LSI y Licenciatura en Ciencias de la Computación LCC y cuenta con el aval y subsidio del Consejo de Investigaciones Científicas y Técnicas y de Creación Artística CICITCA-UNSJ. La propuesta cuenta con antecedentes logrados en el tema conforme a sucesivos proyectos aprobados y subsidiados por el ente mencionado en los que el grupo, conformado por docentes responsables de diferentes asignaturas que dan soporte al área de la Ciencia de Datos (Data Science -DS-), viene trabajando desde el año 2010. Así mismo el proyecto llevado adelante en el ámbito del Instituto de Automética de la Facultad de Ingeniería (INAUT-FI) y con el cual se han desarrollado tareas conjuntas, es el PIO84 "Telemetría Agrícola" y desde su interacción con el medio agroproductivo sanjuanino se han obtenido y se procesaron datos de estaciones meteorológicas de la provincia correspondientes a la estación INTA Pocito y a la del establecimiento privado San Francisco en el departamento Sarmiento distantes ambos 37Km. Estos datos son representados como series temporales y sobre este tipo de datos se trabajó en la presente propuesta.

Algunos de los objetivos ya plasmados en el marco de los proyectos antes mencionados, han sido desarrollar plataformas del tipo "responsive" a fin de que se adapte a cualquier dispositivo que utilice navegador web y desde donde bajo la presentación de información descriptiva y predictiva del estado de los cultivos y diferentes variables climáticas, se faciliten las tomas de decisiones por parte de los agricultores (trabajos finales de grado en LCC defendidos durante el 2022).

Los autores de (1) desarrollaron un índice llamado IG (del portugués Índice de Geada) que permite la predicción de la ocurrencia de heladas en Brasil y países adyacentes (Argentina, Paraguay, Uruguay y Bolivia) con una anticipación de hasta 120 horas. Utilizaron los datos obtenidos de distintas estaciones meteorológicas, desde 2012 a 2018, tomando sólo los meses más fríos (de mayo a septiembre). Las variables que se consideraron en el proceso de desarrollo del modelo fueron la temperatura, la humedad relativa, la velocidad del viento, la presión atmosférica y la nubosidad. El índice toma un enfoque multivariado y se basó principalmente en el análisis de la media y desviación estándar de las variables. Para el análisis de los resultados obtenidos usaron el error cuadrático medio y el R^2. Con este trabajo los autores concluyeron que la temperatura es la variable que mostró tener la mayor influencia en los resultados obtenidos por el IG. Por otro lado, el índice requería un ajuste particular para los estados de Minas Gerais, Río de Janeiro y São Paulo.

En el trabajo (2) se describieron modelos precisos y computacionalmente eficientes implementados utilizando sensores IoT para uso práctico en la predicción de eventos de heladas para aplicaciones agrícolas. Se desarrollaron algoritmos de aprendizaje automático (ML) para la predicción de tales eventos de heladas. Los algoritmos de ML investigados incluyen redes neuronales profundas, redes neuronales de convolución (CNN) y modelos de Random Forest con plazos de 6 a 48 horas. Los resultados mostraron una precisión prometedora para uso en heladas y aplicaciones de predicción de temperatura mínima. También se probó la transferibilidad del modelo mediante aprendizaje continuo y pruebas utilizando datos de Internet. Así mismo se calculó la importancia de los parámetros de los modelos de Random Forest y se logró determinar qué parámetros proporcionaron a los modelos la información más útil para las predicciones.

Según (3), trabajo realizado en la provincia de Mendoza, con características climáticas similares a la provincia de San Juan, se propone un nuevo enfoque, que en lugar de utilizar el sensor información pasada para la predicción, como lo hacen los métodos más avanzados, suponga que la predicción de heladas en un lugar podría mejorarse utilizando la información de los sensores vecinos más relevantes. Dado que se presenta una cantidad relativamente pequeña en el tamaño de datos de la ocurrencia de helada y por lo tanto, se presenta un desequilibrio en los

datos, se propone utilizar algoritmos de ML como redes bayesianas y Random Forest donde el conjunto de datos de entrenamiento incluye nuevas muestras utilizando SMOTE (técnica de sobremuestreo de minorías sintéticas). Los resultados muestran que la selección de los vecinos más relevantes y entrenar los modelos con SMOTE aumenta significativamente la tasa de detección de escarcha del predictor, convirtiendo los resultados en un recurso útil para los tomadores de decisiones.

Los autores de (4) evalúan distintas alternativas para tratar el desequilibrio en los datos, una de ellas lleva a cabo en la fase de preprocesamiento técnicas de remuestreo, que consiste en modificar el conjunto de datos agregando nuevos registros, eliminando alguno de ellos o ambas. En este trabajo se analizaron los resultados de aplicar distintos métodos de remuestreo a los datos registrados por dos estaciones meteorológicas para el estudio del fenómeno de la helada con un clasificador de tipo Random Forest. Se aplicaron cuatro métodos de remuestreo: SMOTE, ADASYN, SMOTETomek y SMOTEENN. Los resultados obtenidos muestran que, independientemente de las variables consideradas, para todos los casos la exactitud balanceada del modelo que opera sobre el conjunto de datos original es inferior al valor de dicha métrica para el modelo que usa datos sobremuestreados. De los cuatro enfoques de remuestreo, SMOTEENN genera el mejor valor en la exactitud balanceada para tres de los cuatro modelos ejecutados. Siendo el modelo que toma las variables temperatura y humedad relativa el de mejor exactitud balanceada. Se concluyó que los mejores resultados se obtuvieron con el algoritmo Random Forest tomando como entrada un conjunto de datos balanceado por el método SMOTEENN considerando las variables temperatura y humedad.

El trabajo (5) aborda el procesamiento de datos climáticos y análisis de algoritmos que contribuyen a determinar el pronóstico del fenómeno meteorológico de la ocurrencia de helada. En este se analizaron distintas variables asociadas al clima, registradas en estaciones meteorológicas automáticas ubicadas en la provincia de San Juan. El objetivo de este estudio fue desarrollar una herramienta moderna para que el productor agropecuario pueda tomar decisiones acertadas ante la ocurrencia de este tipo de fenómeno para evitar o mitigar el daño que ocasiona en los cultivos. Se analizaron la calidad de las respuestas del algoritmo de regresión lineal considerando como entrada los valores de la temperatura durante las últimas tres horas, operando para el rango horario comprendido entre la 0hs hasta las 8hs, período donde usualmente ocurre la helada. También se revisó la literatura respecto del uso de las redes

neuronales para el pronóstico de temperatura. Existiendo antecedentes de la utilización distintos tipos de redes, multicapa perceptrón (MLP), convolucionales (CNN) y de memoria a largo-corto plazo (LSTM).

En el trabajo (6) se utilizan datos de 9 años – desde abril de 2003 hasta abril de 2019 - de pronósticos meteorológicos históricos de distintas regiones de Estados Unidos para entrenar distintos modelos de Random Forest con la finalidad de predecir tornados, granizos y vientos severos. En este trabajo se separa el análisis por regiones: centro, este y oeste de dicho país, proponiendo 5 módelos distintos de Random Forest, logrando así un total de 15 modelos. Estos modelos fueron probados posteriormente con datos desde 2012 a 2016. Las relaciones estadísticas identificadas por los modelos tienen una correspondencia considerable con las relaciones entre variables atmosféricas y climas severos, dando crédito a la veracidad de las soluciones del modelo.

Durante el desarrollo de este trabajo final, y que por lo tanto forma parte del mismo, se desarrolló (7) donde se profundizó en el abordaje metodológico para el procesamiento de datos meteorológicos provenientes del mismo conjunto de datos utilizados para el presente trabajo. Se analizó el impacto de diferentes variables meteorológicas en la ocurrencia, intensidad y duración de las heladas. El objetivo perseguido fue predecir si en un día determinado ocurrirá una helada, que tan intensa será y cuál será su duración para poder alertar al productor agropecuario acerca de este fenómeno, a fin de activar los mecanismos de mitigación correspondientes. También se aplicaron distintos modelos de aprendizaje de máquina, trabajando las predicciones mediante clasificadores y regresores para poder llegar a anticipar la ocurrencia y las características de la helada, los cuales se profundizaron posteriormente en este trabajo. Particularmente los modelos de Random Forest regresor tuvieron resultados aceptables.

3. Marco teórico

3.1. Ciencia de Datos

La ciencia de datos se ocupa del procesamiento de grandes volúmenes de datos utilizando herramientas y técnicas para encontrar patrones, obtener información significativa y apoyar la toma decisiones.

Es un enfoque multidisciplinario que combina principios y prácticas de los campos de las matemáticas, la estadística, la inteligencia artificial y la ingeniería informática para analizar grandes cantidades de datos. Este análisis ayuda a los científicos de datos a hacer y responder preguntas como qué sucedió, por qué sucedió, qué sucederá y qué se puede hacer con los resultados. (8)

3.1.1. Tipos de análisis

Análisis descriptivo

En el análisis descriptivo se examinan los datos para obtener información sobre lo que sucedió o lo que está sucediendo en el entorno de datos. Se caracteriza por visualizaciones de datos como gráficos de torta, gráficos de barras, gráficos de líneas, tabla, entre otros. Por ejemplo, para datos de una estación meteorológica este tipo de análisis describirá la cantidad de días en los que ocurrió helada, cual fue la temperatura promedio en un periodo de tiempo, como evolucionó la humedad a lo largo de una determinada cantidad de meses, entre otras.

Análisis de diagnóstico

El análisis de diagnóstico es una inmersión profunda en los datos para comprender por qué sucedió algo. Se caracteriza por técnicas como el desglose, el descubrimiento de datos, la extracción de datos y las correlaciones. Se pueden realizar múltiples operaciones y transformaciones, en un conjunto de datos determinado, para descubrir patrones únicos en cada una de estas técnicas. Siguiendo el ejemplo anterior, en este caso se evaluarían cuáles son los factores climáticos que propician la ocurrencia de helada.

Análisis predictivo

El análisis predictivo utiliza datos históricos para realizar pronósticos precisos sobre los patrones de datos que pueden ocurrir en el futuro. Se caracteriza por técnicas como el aprendizaje automático, la predicción, la coincidencia de patrones y el modelado predictivo. En cada una de estas técnicas, las computadoras están capacitadas para aplicar ingeniería inversa a

las conexiones de causalidad en los datos. Por ejemplo, el equipo de servicio de vuelos podría usar la ciencia de datos para predecir si en un día determinado ocurrirá o no helada.

Análisis prescriptivo

El análisis prescriptivo lleva los datos predictivos al siguiente nivel. No solo predice lo que es probable que suceda, sino que también sugiere una respuesta óptima a ese resultado. Puede analizar las implicaciones potenciales de diferentes opciones y recomendar el mejor curso de acción. Utiliza análisis de gráficos, simulación, procesamiento de eventos complejos, redes neuronales y motores de recomendación de aprendizaje automático.

3.1.2. El proceso de ciencia de datos

El proceso de ciencia de datos, también a veces denominado como ciclo de vida de ciencia de datos, es un enfoque sistemático para resolver un problema. Provee un marco para articular el problema como una pregunta, decidir cómo resolverlo y luego presentar la solución a las partes interesadas.

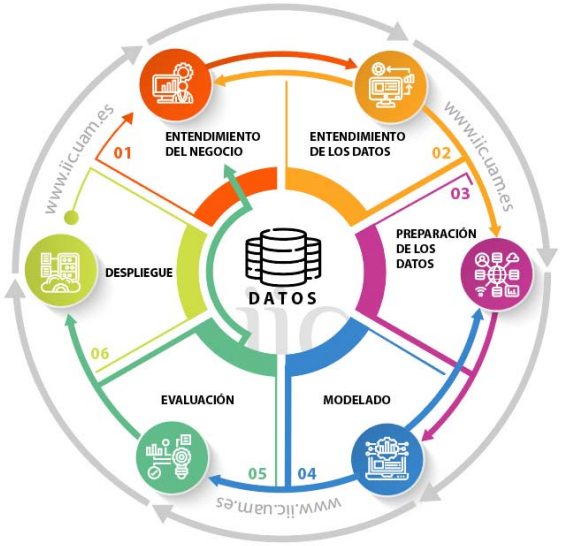

Ilustración 1 Etapas del ciclo de vida de Ciencia de Datos (9)

Entendimiento y definición del problema

En esta primera fase, las partes interesadas realizan regularmente las siguientes tareas: examinar las tendencias, realizar estudios de casos de análisis de datos similares y estudiar el área de

dominio en el que se va profundizar. En esta etapa se realiza una evaluación de los recursos disponibles, el tiempo total involucrado y los requisitos tecnológicos.

Recolección de datos

Los datos pueden ser preexistentes, recién adquiridos o descargados de Internet. Los científicos de datos pueden extraer datos de bases de datos internas o externas, herramientas de software, registros de servidores web, redes sociales o comprarlos de fuentes de terceros confiables.

Preparación de los datos

En esta etapa se lleva adelante un proceso de estandarizar los datos de acuerdo con un formato predeterminado. Incluye el manejo de datos faltantes, la corrección de errores de datos y la eliminación de datos atípicos. Algunas tareas realizadas en esta etapa, incluyen:

- Selección de datos relevantes.
- Integración de datos provenientes de distintas fuentes, fusionando los conjuntos de datos.
- Tratar los valores faltantes.
- Tratar los datos erróneos eliminándolos,
- Comprobación de valores atípicos mediante técnicas de visualización como diagramas de caja y manejo de los mismos.

Esta etapa es quizás la que más tiempo demanda de todo el proceso y es posible que se retorne a estas múltiples veces.

Análisis exploratorio de datos

En esta etapa, se deben comprender en profundidad, los datos donde que ya están disponibles en el formato requerido. Esta comprensión proviene del análisis de datos utilizando varias herramientas estadísticas disponibles. Aquí también se realiza una análisis inferencial de los datos donde se examinan mediante la formulación de diferentes funciones estadísticas y se identifican variables dependientes e independientes. Un análisis cuidadoso revela qué datos son relevantes o importantes y cuál es la distribución de los mismos. Se utilizan los más variados gráficos para visualizar los datos y lograr así, una mejor comprensión de los mismos.

Modelado de datos

En esta etapa se decidirá que técnica de aprendizaje de máquina se utilizará para los modelos (regresión, clasificación o clustering), de acuerdo a los objetivos que se persigan. Una vez seleccionada la técnica, se deberá optar por el modelo específico, para esto se prueban distintos algoritmos y parámetros con la finalidad de determinar el modelo más apropiado.

Evaluación del modelo

En esta etapa se analizan las diferentes métricas para determinar cuál es el rendimiento del modelo y si es aceptable para los objetivos que se persiguen. Dependiendo del modelo utilizado se emplearán distintas métricas. El modelo puede probarse con datos de prueba predeterminados para evaluar la precisión de los resultados. El modelo de datos se puede ajustar muchas veces para mejorar los resultados.

Visualización y comunicación de los resultados

En esta etapa final se convierte el conocimiento extraído de los datos en acción. Se realizan diagramas, gráficos y cuadros para representar tendencias y predicciones. El resumen de datos ayuda a las partes interesadas a comprender e implementar los resultados de manera efectiva.

3.2. Inteligencia Artificial

3.2.1. Definición

Es la ciencia y la ingeniería de fabricar máquinas inteligentes, especialmente programas informáticos inteligentes. Está relacionado con la tarea similar de usar computadoras para comprender la inteligencia humana, pero la IA no tiene que limitarse a métodos que son biológicamente observables. (10)

3.2.2. Deep Learning y Machine Learning

Dado que los términos *Machine Learning* (aprendizaje automático) y *Deep Learning* (aprendizaje profundo) tienden a usarse indistintamente, vale la pena señalar los matices entre los dos. Ambos son sub campos de la inteligencia artificial, y el aprendizaje profundo es en realidad un sub campo del aprendizaje automático, como se muestra en la Ilustración 2.

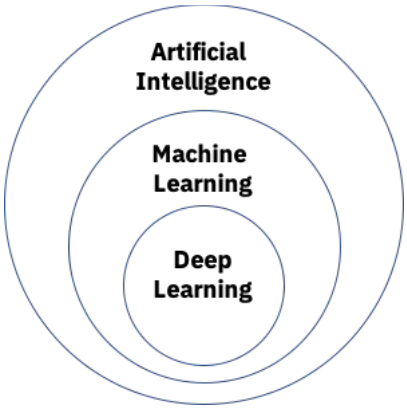

Ilustración 2 Inteligencia artificial, machine learning y deep learning (11)

La forma en que el aprendizaje profundo y el aprendizaje automático difieren es cómo aprende cada algoritmo. El aprendizaje automático "profundo" puede usar conjuntos de datos etiquetados, también conocidos como aprendizaje supervisado, para entrenar su algoritmo, pero no necesariamente requiere un conjunto de datos etiquetados. El aprendizaje profundo puede ingerir datos no estructurados en su forma sin procesar (por ejemplo, texto, imágenes) y puede determinar automáticamente el conjunto de características que distinguen las diferentes categorías de datos entre sí. Esto elimina parte de la intervención humana requerida y permite el uso de conjuntos de datos más grandes. (12)

El aprendizaje automático clásico, o "no profundo", depende más de la intervención humana para aprender. Los expertos humanos determinan el conjunto de características para comprender las diferencias entre las entradas de datos, lo que generalmente requiere datos más estructurados para aprender. (13)

El aprendizaje profundo utiliza redes neuronales, donde el término profundo refiere a una red neuronal con más de tres capas, incluidas las capas de entrada y salida.

3.3. Series temporales

3.3.1. Definición

El análisis de series temporales es una forma de analizar una secuencia de puntos de datos recopilados durante un intervalo de tiempo. En el análisis de series temporales, se registran puntos de datos a intervalos constantes durante un período de tiempo determinado en lugar de simplemente registrar los puntos de datos de forma intermitente o aleatoria. Sin embargo, este tipo de análisis no es simplemente el acto de recopilar datos a lo largo del tiempo. (14)

Lo que distingue a los datos de series temporales de otros datos es que el análisis puede mostrar cómo cambian las variables con el tiempo. Esto es, el tiempo es una variable crucial porque muestra cómo se ajustan los datos a lo largo de los puntos de datos, así como los resultados finales. Proporciona una fuente adicional de información y un orden establecido de dependencias entre los datos.

El análisis de series temporales generalmente requiere una gran cantidad de puntos de datos para garantizar la coherencia y la confiabilidad. Un amplio conjunto de datos garantiza que tenga un tamaño de muestra representativo y que el análisis pueda eliminar los datos ruidosos. También garantiza que cualquier tendencia o patrón descubierto no sea un valor atípico y pueda explicar la variación estacional. Además, los datos de series temporales se pueden usar para realizar pronósticos, es decir, predecir datos futuros en función de datos históricos.

3.3.2. Conceptos de series temporales

- Estacionalidad: Se refiere a la propiedad de que las estadísticas de la serie temporal, como la media y la varianza, son constantes a lo largo del tiempo. En otras palabras, se refiere a las fluctuaciones periódicas. (15)

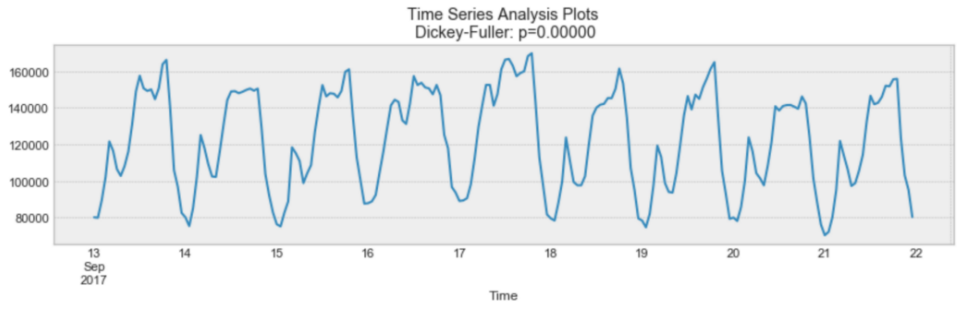

Ilustración 3 Serie temporal con estacionalidad (15)

- Autocorrelación: Se refiere a la relación entre los valores de la serie temporal en diferentes momentos del tiempo. Esto es, la similitud entre las observaciones en función del tiempo transcurrido entre ellas.

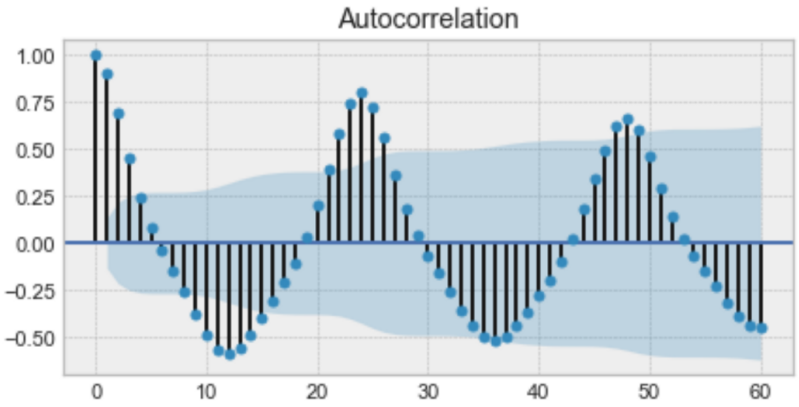

Ilustración 4 Serie temporal con autocorrelación (15)

- Estacionariedad: Se dice que una serie de tiempo es estacionaria si sus propiedades estadísticas no cambian con el tiempo. En otras palabras, tiene media y varianza constantes, y la covarianza es independiente del tiempo. (15)

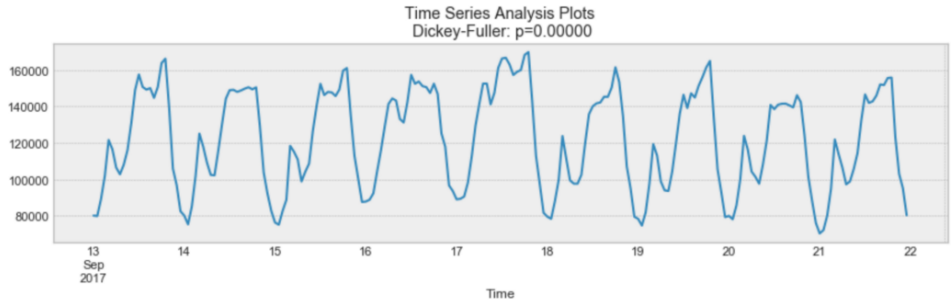

Ilustración 5 Serie temporal con estacionariedad (15)

Esto es, la estacionariedad se refiere a la constancia estadística de una serie temporal, mientras que la estacionalidad se refiere a los patrones regulares y recurrentes en una serie temporal que se repiten en períodos fijos.

3.3.3. Modelos de series temporales

Existen una variedad de métodos para estudiar datos. Entre los más comunes se encuentran (14):

- Modelos ARIMA de Box-Jenkins: estos modelos univariados se utilizan para comprender mejor una sola variable dependiente del tiempo, como la temperatura a lo largo del tiempo, y para predecir futuros puntos de datos de variables. Estos modelos

funcionan bajo el supuesto de que los datos son estacionarios (la media y la varianza no dependen del tiempo). Se debe tener en cuenta y eliminar tantas diferencias y estacionalidades en puntos de datos anteriores como sea posible. El modelo ARIMA incluye términos para tener en cuenta los promedios móviles, los operadores de diferencia estacional y los términos autorregresivos dentro del modelo.

- Modelos multivariados de Box-Jenkins: los modelos multivariados se utilizan para analizar más de una variable dependiente del tiempo, como la temperatura y la humedad, a lo largo del tiempo.
- Método Holt-Winters: El método Holt-Winters es una técnica de suavizado exponencial. Está diseñado para predecir resultados, siempre que los puntos de datos incluyan la estacionalidad.

3.4. Machine Learning

3.4.1. Definición

El aprendizaje automático es una rama de la inteligencia artificial (IA) y la informática que se centra en el uso de datos y algoritmos para imitar la forma en que los humanos aprenden, mejorando gradualmente su precisión. (13)

El algoritmo típico de aprendizaje automático supervisado consta de aproximadamente tres componentes:

- Un proceso de decisión: una receta de cálculos u otros pasos que toma los datos y estima qué tipo de patrón busca encontrar su algoritmo.
- Una función de error: un método para medir qué tan buena fue la "estimación" comparándola con ejemplos conocidos (en caso de estar disponibles). ¿El proceso de decisión lo hizo bien? Si no es así, ¿cómo cuantifica "qué tan grave" fue el fallo?
- Un proceso de actualización u optimización: un método en el que, el algoritmo analiza el fallo y luego actualiza cómo llega en su proceso a la decisión final, de modo que la próxima vez (próxima ejecución) el fallo no sea tan grande.

3.4.2. Métodos de aprendizaje automático

Aprendizaje automático supervisado

El aprendizaje supervisado se define por el uso de conjuntos de datos etiquetados para entrenar algoritmos para clasificar datos o predecir resultados con precisión. A medida que los datos de entrada se introducen en el modelo, el modelo ajusta sus pesos hasta lograr su correcto funcionamiento. Esto ocurre como parte del proceso de validación cruzada para garantizar que el modelo evite el sobreajuste o el ajuste insuficiente. El aprendizaje permite resolver una variedad de problemas del mundo real a escala, como clasificar el spam en una carpeta separada de su bandeja de entrada. Algunos métodos utilizados en el aprendizaje supervisado incluyen redes neuronales, naïve bayes, regresión lineal, regresión logística, Random Forest y máquina de vectores de soporte (SVM).

Aprendizaje automático no supervisado

El aprendizaje no supervisado utiliza algoritmos de aprendizaje automático para analizar y agrupar conjuntos de datos no etiquetados. Estos algoritmos descubren patrones ocultos o agrupaciones de datos sin necesidad de intervención humana. La capacidad de este método para descubrir similitudes y diferencias en la información lo hace ideal para el análisis exploratorio de datos, segmentación de grupos y reconocimiento de imágenes y patrones. También se utiliza para reducir la cantidad de funciones en un modelo a través del proceso de reducción de dimensionalidad. El análisis de componentes principales (PCA) y la descomposición de valores singulares (SVD) son dos enfoques comunes para esto. Otros algoritmos utilizados en el aprendizaje no supervisado incluyen redes neuronales, K-Means y métodos de agrupamiento probabilístico.

Aprendizaje semi supervisado

El aprendizaje semi supervisado ofrece un término medio entre el aprendizaje supervisado y el no supervisado. Durante el entrenamiento, utiliza un conjunto de datos etiquetados más pequeño para guiar la clasificación y la extracción de características de un conjunto de datos más grande sin etiquetar. El aprendizaje semi supervisado puede resolver el problema de no tener suficientes datos etiquetados para un algoritmo de aprendizaje supervisado. También ayuda si es demasiado costoso etiquetar suficientes datos (13).

3.4.3. Algoritmos de aprendizaje de máquina

Existen diversos algoritmos de aprendizaje de máquina, a continuación, se desarrollarán algunos de ellos.

3.4.3.1. Hiperparámetros

Los hiperparámetros son variables de configuración externa que se utilizan para administrar el entrenamiento de modelos de aprendizaje de máquina. A veces llamados *hiperparámetros de modelos*, los hiperparámetros se configuran de manera manual antes de entrenar un modelo. Son diferentes de los parámetros, que son elementos internos derivados de manera automática durante el proceso de aprendizaje y que no están configurados por científicos de datos.

Los hiperparámetros controlan de forma directa la estructura, funciones y rendimiento de los modelos. El ajuste de hiperparámetros permite modificar el rendimiento del modelo para lograr resultados óptimos, por lo cual este proceso es una parte fundamental del machine learning.

3.4.3.2. Regresión Lineal

El análisis de regresión lineal se utiliza para predecir el valor de una variable en función del valor de otra variable. La variable que desea predecir se llama variable dependiente. La variable que se está usando para predecir el valor de la otra variable se llama variable independiente.

Esta forma de análisis estima los coeficientes de la ecuación lineal, involucrando una o más variables independientes que predicen mejor el valor de la variable dependiente. La regresión lineal se ajusta a una línea recta o superficie que minimiza las discrepancias entre los valores de salida previstos y reales. Utilizan un método de "mínimos cuadrados" para descubrir la línea que mejor se ajusta a un conjunto de datos emparejados. Luego estima el valor de X (variable dependiente) a partir de Y (variable independiente). (16)

La regresión lineal es un algoritmo utilizado para analizar la relación entre las variables de entrada independientes y al menos una variable objetivo. Este tipo de regresión se usa para predecir resultados continuos, variables que pueden tomar cualquier resultado numérico. Por ejemplo, dados los datos sobre el vecindario y la propiedad, ¿puede un modelo predecir el valor de venta de una casa? Las relaciones lineales ocurren cuando la relación de datos que se observa tiende a seguir una línea recta en general, y como tal, este modelo se puede usar para observar si un punto de datos aumenta, disminuye o permanece igual en relación con alguna variable independiente, como el tiempo. transcurrido o posición.

Los modelos de aprendizaje automático se pueden emplear para analizar datos con el fin de observar y mapear regresiones lineales. Las variables independientes y las variables objetivo se pueden ingresar en un modelo de aprendizaje automático de regresión lineal, y el modelo luego asignará los coeficientes lineales de mejor ajuste a los datos. En otras palabras, los modelos de regresión lineal intentan trazar una línea recta, o una relación lineal, a través del conjunto de datos.

3.4.3.3. Regresión Logística

La regresión logística es un algoritmo de aprendizaje supervisado que se utiliza para problemas de clasificación. En lugar de una salida continua como en la regresión lineal, un modelo logístico predice la probabilidad de que ocurra un evento binario. Por ejemplo, dado un correo electrónico, ¿puede un modelo predecir si el contenido es spam o no?

Los algoritmos de aprendizaje automático pueden usar modelos de regresión logística para determinar resultados categóricos. Cuando se le proporciona un conjunto de datos, el modelo de regresión logística puede verificar cualquier peso y sesgo y luego usar las variables objetivo categóricas dependientes dadas para comprender cómo categorizar correctamente ese conjunto de datos. (16)

La regresión logística estima la probabilidad de que ocurra un evento, en función de un conjunto de datos determinado de variables independientes. Dado que el resultado es una probabilidad, la variable dependiente está limitada entre 0 y 1. En la regresión logística, se aplica una transformación logit a las probabilidades, es decir, la probabilidad de éxito dividida por la probabilidad de fracaso. Esto también se conoce comúnmente como probabilidades logarítmicas, o el logaritmo natural de las probabilidades, y esta función logística se representa mediante las siguientes fórmulas:

$$Logit(pi)=1/(1+exp(-pi)) \quad (1)$$

$$ln(pi/(1-pi))=B0+B1*X1+\ldots+Bk*Xk \quad (2)$$

En esta ecuación de regresión logística ec. (1) es la variable dependiente o de respuesta y x es la variable independiente. El parámetro beta, o coeficiente, en este modelo se estima comúnmente a través de la estimación de máxima verosimilitud (MLE). Este método prueba diferentes valores de beta a través de múltiples iteraciones para optimizar el mejor ajuste de probabilidades logarítmicas. Todas estas iteraciones producen la función de verosimilitud

logarítmica, y la regresión logística busca maximizar esta función para encontrar la mejor estimación de parámetros. Una vez que se encuentra el coeficiente óptimo (o los coeficientes si hay más de una variable independiente), las probabilidades condicionales para cada observación se pueden calcular, registrar y sumar para producir una probabilidad predicha. Para la clasificación binaria, una probabilidad inferior a 0,5 predecirá 0, mientras que una probabilidad superior predecirá 1. Después de calcular el modelo, es una buena práctica evaluar qué tan bien el modelo predice la variable dependiente, que se denomina bondad del ajuste.

3.4.3.4. Random Forest

3.4.3.4.1. Árbol de Decisión

Un árbol puede "aprender" dividiendo el conjunto de datos original en subconjuntos en función de una prueba de valor de atributo. Este proceso se repite en cada subconjunto derivado de una manera recursiva, denominada partición recursiva. La recursividad se completa cuando el subconjunto en un nodo tiene el mismo valor de la variable objetivo (todas las hojas del subárbol tienen el mismo valor de conclusión), o cuando la división ya no agrega valor a las predicciones. La construcción de un clasificador de árbol de decisión no requiere ningún conocimiento del dominio o configuración de parámetros y, por lo tanto, es apropiado para el descubrimiento de conocimiento exploratorio. Los árboles de decisión pueden manejar datos de alta dimensión. En general, el clasificador de árboles de decisión tiene buena precisión. (17)

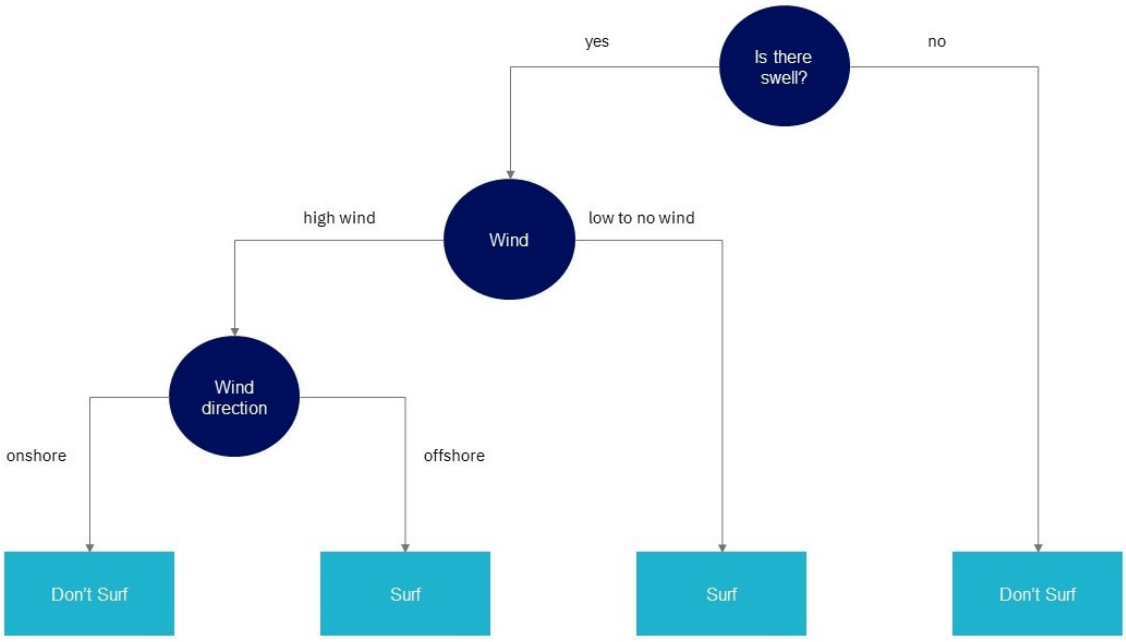

Ilustración 6 Ejemplo didáctico de un árbol de decisión (17)

En la Ilustración 6 se puede observar un ejemplo didáctico que consiste en el árbol de decisión necesario para hacer surf. Para esto hace una evaluación de las siguientes variables: olas, viento y la dirección del viento, para determinar si se puede practicar o no el deporte.
Si bien los árboles de decisión son algoritmos de aprendizaje supervisado comunes, pueden ser propensos a problemas, como sesgos y sobreajuste. Sin embargo, cuando varios árboles de decisión forman un conjunto en el algoritmo de Random Forest, prediciendo resultados más precisos, especialmente cuando los árboles individuales no están correlacionados entre sí.

3.4.3.4.2. Índice Gini

La impureza Gini es una medida que se utiliza para generar árboles de clasificación. Proporciona más información acerca de la distribución de los datos por nodo que la precisión de clasificación utilizada para informes de precisión del árbol.
La impureza de un nodo del árbol de clasificación se calcula utilizando el recuento de cada categoría de destino entre todos los registros correspondientes para un nodo dado. El total de impureza Gini se calcula como una suma de cuadrados del recuento de proporciones entre todas las categorías de destino por nodo al que se resta uno y los resultados se multiplican por el número de registros.
Por ejemplo, cuando se divide un nodo de árbol, el algoritmo busca un campo con la mejora más elevada del total de impureza calculada como el total de impureza entre todos los nodos hijo potenciales restados del total de impureza del nodo padre.

3.4.3.4.3. Algoritmo de Random Forest

Los algoritmos de Random Forest tienen tres hiperparámetros principales, que deben configurarse antes del entrenamiento. Estos incluyen el tamaño del nodo, la cantidad de árboles y la cantidad de características muestreadas. A partir de ahí, el clasificador de Random Forest se puede utilizar para resolver problemas de regresión o clasificación.
El algoritmo de bosque aleatorio (Random Forest) se compone de una colección de árboles de decisión, y cada árbol del conjunto se compone de una muestra de datos extraída de un conjunto de entrenamiento. De esa muestra de entrenamiento, un tercio se reserva como datos de prueba, que se suele denominar *out of the bag.* Luego, se inyecta otra instancia de aleatoriedad a través del agregado de nuevas características, lo que agrega más diversidad al conjunto de datos y

reduce la correlación entre los árboles de decisión. Dependiendo del tipo de problema, la determinación de la predicción variará. Para una tarea de regresión, se promediarán los árboles de decisión individuales, y para una tarea de clasificación, un voto mayoritario, es decir, la variable categórica más frecuente producirá la clase predicha. Finalmente, la muestra *out of the bag* se usa para la validación cruzada, finalizando esa predicción.

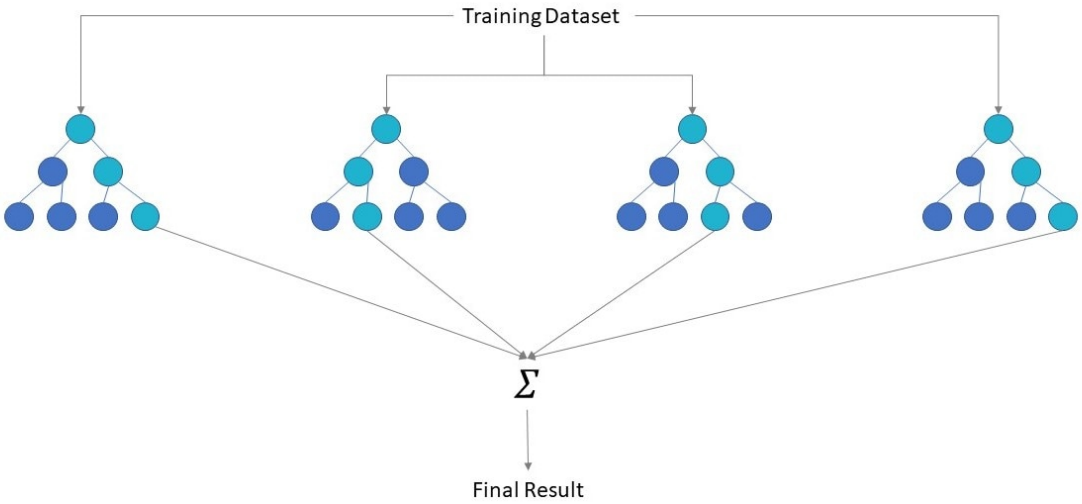

Ilustración 7 Diagrama de Random Forest (18)

3.4.3.4.4. Beneficios de Random Forest

- Riesgo reducido de sobreajuste: los árboles de decisión corren el riesgo de sobreajuste, ya que tienden a ajustarse estrechamente a todas las muestras dentro de los datos de entrenamiento. Sin embargo, cuando hay una gran cantidad de árboles de decisión en un Random Forest, el clasificador no sobreajustará el modelo, ya que el promedio de árboles no correlacionados reduce la varianza general y el error de predicción. (17)
- Brinda flexibilidad: dado que el Random Forest puede manejar tareas de regresión y clasificación con un alto grado de precisión, también convierte al clasificador de bosque aleatorio en una herramienta eficaz para estimar los valores faltantes, ya que mantiene la precisión cuando falta una parte de los datos. (17)
- Fácil de determinar la importancia de los atributos: el Random Forest facilita la evaluación de la contribución de las variables al modelo. Existen diversas formas de evaluar la importancia de los atributos. La importancia de Gini y la disminución media de la impureza (MDI) generalmente se usan para medir cuánto disminuye la precisión del modelo cuando se excluye una variable determinada. Sin embargo, la importancia

de la permutación, también conocida como precisión de disminución media (MDA), es otra medida de importancia. MDA identifica la disminución promedio en la precisión al permutar aleatoriamente los valores de características en muestras *out of the bag*. (17)

3.5. Aprendizaje Profundo

El aprendizaje profundo, o *Deep Learning,* es un subconjunto del aprendizaje automático, que es esencialmente una red neuronal. Estas redes neuronales intentan simular el comportamiento del cerebro humano, aunque lejos de igualar su capacidad, permite "aprender" de grandes cantidades de datos. Si bien una red neuronal con una sola capa aún puede hacer predicciones aproximadas, las capas ocultas adicionales pueden ayudar a optimizar y refinar la precisión. (10)

3.5.1. Redes Neuronales

Las redes neuronales, también conocidas como redes neuronales artificiales (ANN, del inglés Artificial Neural Network), son un subconjunto del aprendizaje automático y están en el corazón de los algoritmos de aprendizaje profundo. Su nombre y estructura están inspirados en el cerebro humano, imitando la forma en que las neuronas biológicas se envían señales entre sí. (19)

Las redes neuronales artificiales se componen de capas de nodos, que contienen una capa de entrada, una o más capas ocultas y una capa de salida. Cada nodo, o neurona artificial, se conecta con otro y tiene asociado un peso y un umbral. Si la salida de cualquier nodo individual está por encima del valor de umbral especificado, ese nodo se activa y envía datos a la siguiente capa de la red. De lo contrario, no se pasan datos a la siguiente capa de la red.

Las redes neuronales se basan en datos de entrenamiento para aprender y mejorar su precisión con el tiempo. Sin embargo, una vez que estos algoritmos de aprendizaje se ajustan con precisión permiten clasificar y agrupar datos a alta velocidad. Las tareas de reconocimiento de voz o reconocimiento de imágenes pueden llevar minutos en lugar de horas en comparación con la identificación manual por parte de expertos humanos.

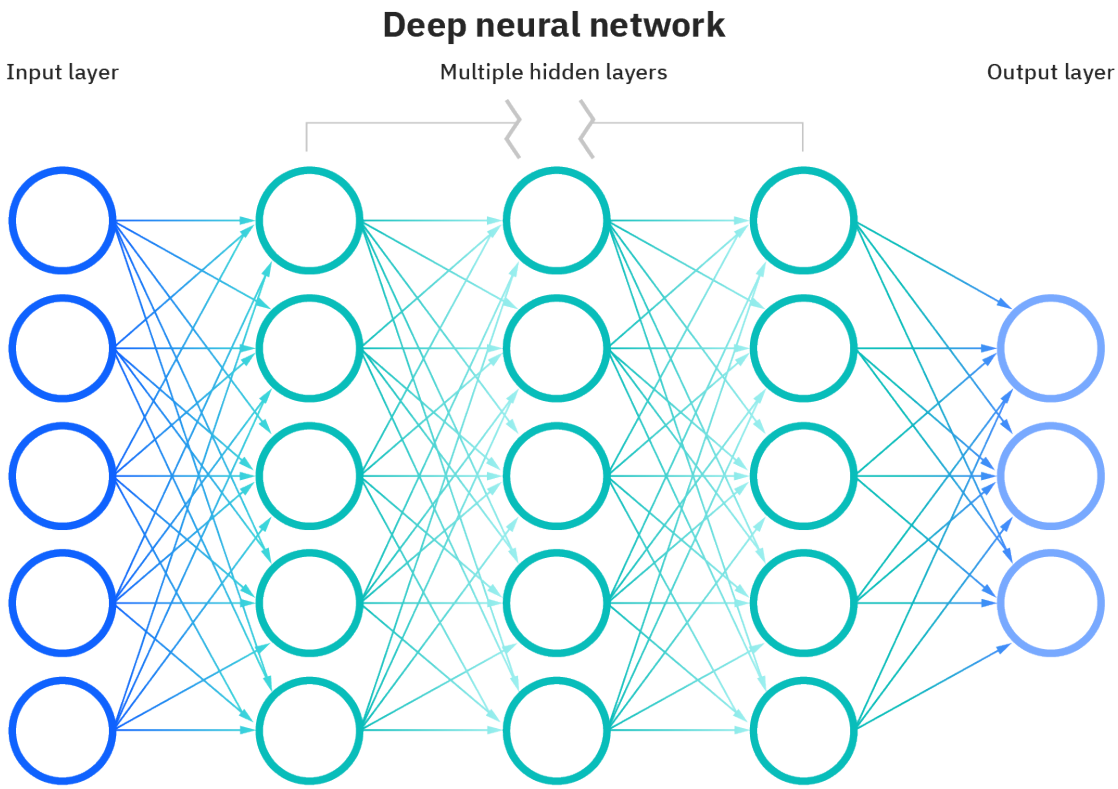

Ilustración 8 Diagrama de una red neuronal (19)

3.5.1.1. Aprendizaje en una Red Neuronal

El aprendizaje es la adaptación de la red para lograr mejores resultados de predicción dada una muestra. El aprendizaje implica ajustar los pesos y umbrales de la red para mejorar la precisión del resultado. Esto se hace minimizando los errores observados. Se considera que el aprendizaje está completo cuando el examen de observaciones adicionales no reduce de manera significativa la tasa de error. Incluso después del aprendizaje, la tasa de error normalmente no llega a 0. Si después del aprendizaje, la tasa de error es demasiado alta, normalmente se debe rediseñar la red. Prácticamente esto se hace definiendo una función de costo que se evalúa periódicamente durante el aprendizaje. Mientras la tasa de error continúe disminuyendo, el aprendizaje continúa.

El costo se define frecuentemente como una estadística cuyo valor solo puede ser aproximado. Las salidas son en realidad números, por lo que cuando el error es bajo, la diferencia entre la salida y la respuesta correcta es pequeña. El aprendizaje intenta reducir el total de las diferencias entre las observaciones. La mayoría de los modelos de aprendizaje pueden verse como una aplicación directa de la teoría de la optimización y la estimación estadística. (12)

3.5.1.2. Elementos de una red neuronal

3.5.1.2.1. Capas

- Capa de entrada: La capa de entrada toma la entrada sin procesar del dominio. No se realiza ningún cálculo en esta capa. Los nodos aquí solo pasan la información (características) a la capa oculta.
- Capa oculta: Como sugiere el nombre, los nodos de esta capa no están expuestos. Proporcionan abstracción a la red neuronal. La capa oculta realiza todo tipo de cálculos sobre las observaciones ingresadas a través de la capa de entrada y transfiere el resultado a la capa de salida.
- Capa de salida: Es la capa final de la red, la que trae la información aprendida a través de la capa oculta y entrega el valor final como resultado. (19)

3.5.1.2.2. Función de costo

La función de costo de una red neuronal es la suma de errores en cada capa. Esto se logra encontrando primero el error en cada capa y luego sumando el error individual para obtener el error total.

3.5.1.2.3. Backpropagation

Los pesos de las conexiones de red se ajustan repetidamente para minimizar la diferencia entre el vector de salida real de la red y el vector de salida deseado. *Backpropagation* tiene como objetivo minimizar la función de costo ajustando los pesos y sesgos de la red. Los gradientes de la función de costo determinan el nivel de ajuste con respecto a parámetros como función de activación, pesos, sesgo. (20)

3.5.1.2.4. Función de activación

La función de activación decide si una neurona debe activarse o no. Esto significa que decidirá si la entrada de la neurona a la red es importante o no en el proceso de predicción utilizando operaciones matemáticas más simples. La función principal de la función de activación es transformar la entrada ponderada sumada del nodo en un valor de salida para alimentar la siguiente capa oculta o como salida.

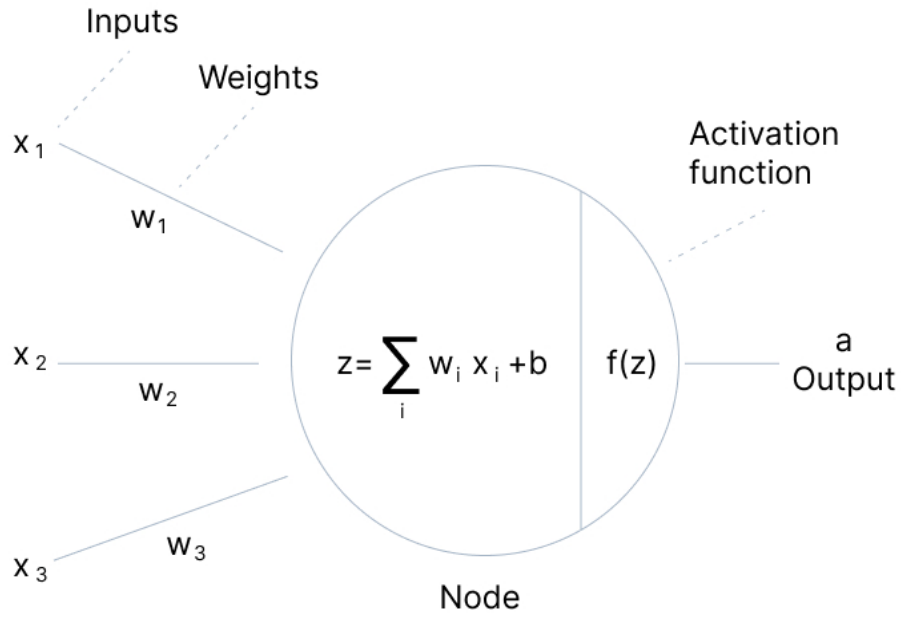

Ilustración 9 Función de activación de una red neuronal (20)

Las funciones de activación introducen un paso adicional en cada capa durante la *forward propagation*, pero es necesario. Supongamos que se tiene una red neuronal funcionando sin las funciones de activación. En ese caso, cada neurona solo realizará una transformación lineal en las entradas utilizando los pesos y sesgos. Es porque no importa cuántas capas ocultas adjuntemos en la red neuronal; todas las capas se comportarán de la misma manera porque la composición de dos funciones lineales es una función lineal en sí misma. Aunque la red neuronal se vuelve más simple, aprender cualquier tarea compleja es imposible, y el modelo sería solo un modelo de regresión lineal.

3.5.1.2.5. Tipos de funciones de activación

Función de activación binaria

La función de paso binario depende de un valor de umbral que decide si una neurona debe activarse o no. La entrada alimentada a la función de activación se compara con un cierto umbral; si la entrada es mayor que ella, entonces la neurona se activa; de lo contrario, se desactiva, lo que significa que su salida no pasa a la siguiente capa oculta.

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

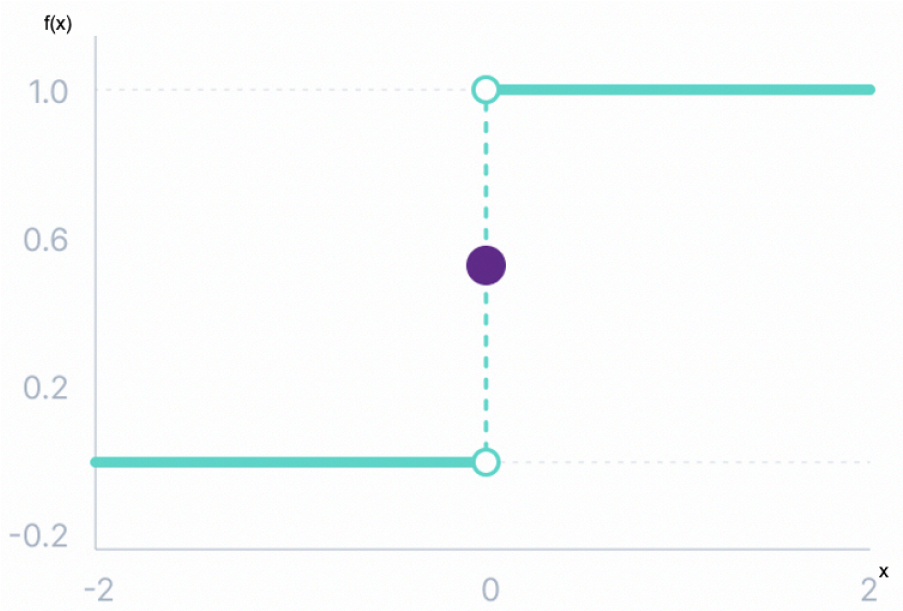

Ilustración 10 Función de activación binaria (20)

Función de activación lineal

La función de activación lineal, también conocida como "sin activación" o "función de identidad", es donde la activación es proporcional a la entrada. La función no hace nada con la suma ponderada de la entrada, simplemente retorna el valor que se le dio.

$$f(x) = x$$

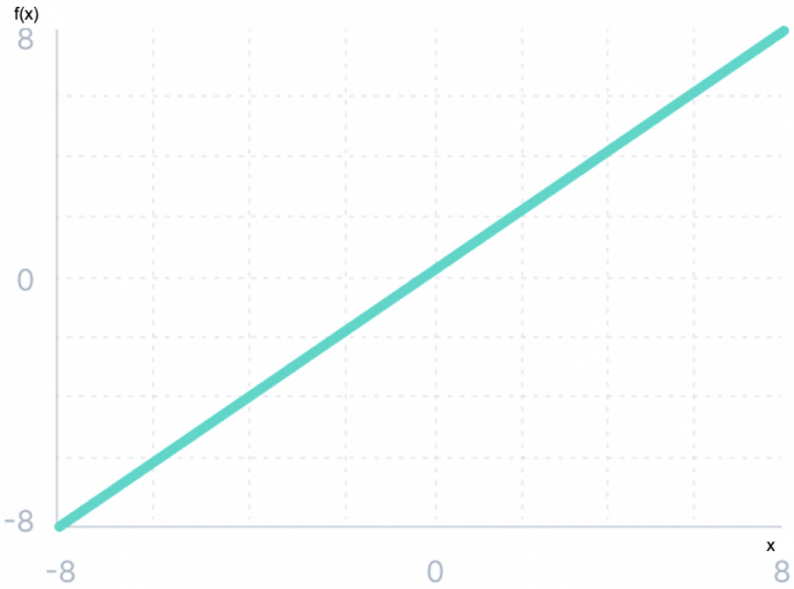

Ilustración 11 Función de activación lineal (20)

Funciones de activación no lineales

La función de activación lineal que se muestra anteriormente es simplemente un modelo de regresión lineal. Debido a su poder limitado, esto no permite que el modelo cree mapeos complejos entre las entradas y salidas de la red. Las funciones de activación no lineales resuelven las siguientes limitaciones de las funciones de activación lineales: Permiten la *backpropagation* porque ahora la función derivada estaría relacionada con la entrada, y es posible retroceder y comprender qué pesos en las neuronas de entrada pueden proporcionar una mejor predicción. Permiten el apilamiento de múltiples capas de neuronas, ya que la salida ahora sería una combinación no lineal de entrada que pasa a través de múltiples capas. Cualquier salida se puede representar como un cálculo funcional en una red neuronal. (20)

Función de activación Sigmoid o logística

Esta función toma cualquier valor real como entrada y genera valores en el rango de 0 a 1.

Cuanto mayor sea la entrada (más positiva), más cerca estará el valor de salida de 1, mientras que cuanto más pequeña sea la entrada (más negativa), más cerca estará la salida de 0.

Esta es una función ampliamente usada para modelos en los cuales el objetivo es predecir la probabilidad como salida. Dado que la probabilidad de cualquier cosa existe solo entre el rango de 0 y 1, sigmoid es la elección correcta debido a su rango. (20)

La función es diferenciable y proporciona un gradiente suave, es decir, evita saltos en los valores de salida. Esto está representado por una forma de S de la función de activación sigmoidea.

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

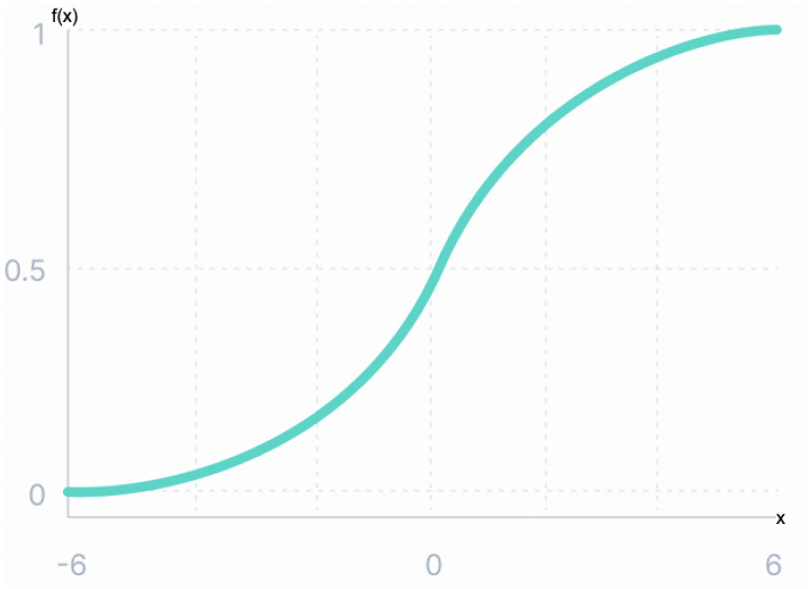

Ilustración 12 Función de activación Sigmoid / Logística (20)

Función de activación tanh

La función Tanh es similar a la función de activación sigmoid/logística, e incluso tiene la misma forma de S con una diferencia en el rango de salida de -1 a 1. En Tanh, cuanto mayor sea la entrada (más positiva), más cercano será el valor de salida de 1,0, mientras que cuanto menor sea la entrada (más negativa), más cerca estará la salida de -1,0. (20)

Los usos para esta función son:

- La salida de la función de activación de tanh está centrada en cero; por lo tanto, es posible mapear con facilidad los valores de salida como fuertemente negativos, neutrales o fuertemente positivos.
- Suele utilizarse en capas ocultas de una red neuronal ya que sus valores se encuentran entre -1 y 1; por lo tanto, la media de la capa oculta resulta ser 0 o muy cercana. Ayuda a centrar los datos y facilita el aprendizaje para la siguiente capa.

$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

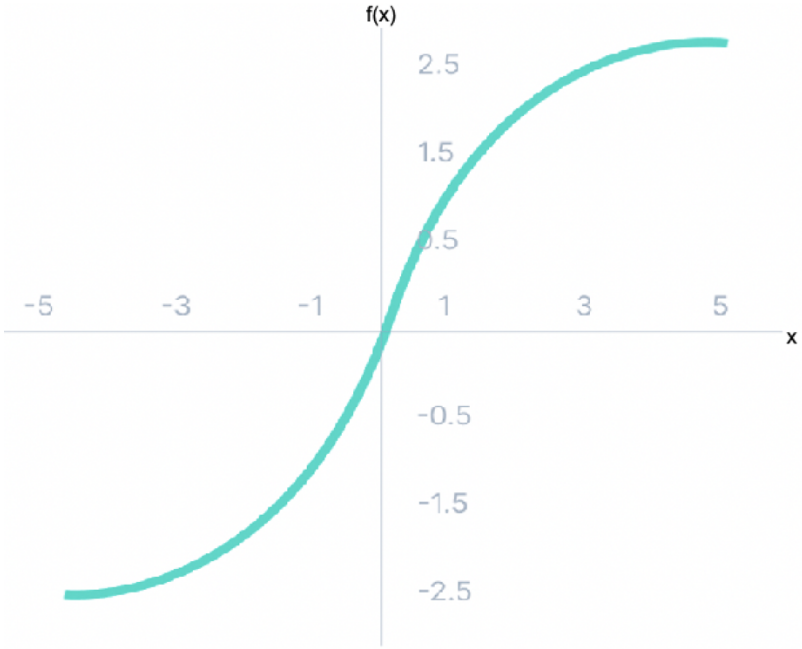

Ilustración 13 Función de activación tanh (20)

Función de activación ReLU

Aunque da la impresión de una función lineal, ReLU tiene una función derivada y permite la *backpropagation* al mismo tiempo que lo hace computacionalmente eficiente. El problema principal es que la función ReLU no activa todas las neuronas al mismo tiempo. Las neuronas solo se desactivarán si la salida de la transformación lineal es menor que 0. (20)

Dado que solo se activa un cierto número de neuronas, la función ReLU es mucho más eficiente desde el punto de vista computacional en comparación con las funciones sigmoid y tanh. ReLU acelera la convergencia del descenso del gradiente hacia el mínimo global de la función de pérdida debido a su propiedad lineal no saturada.

$$f(x) = max\ (0, x)$$

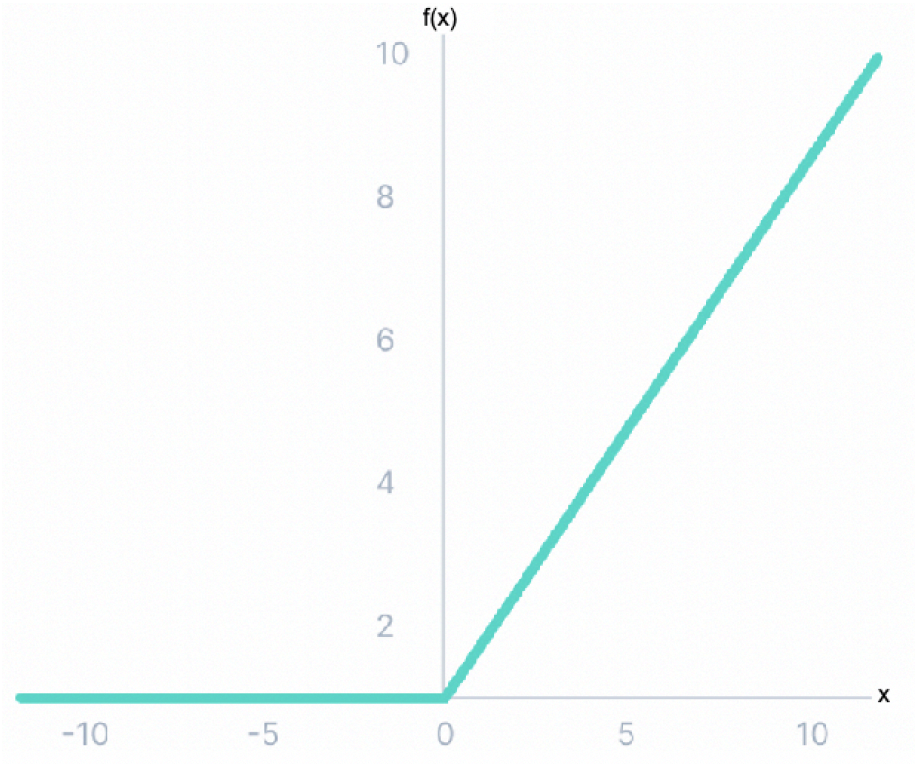

Ilustración 14 Función de activación ReLU (20)

3.5.1.3. Tipos de Redes Neuronales

3.5.1.3.1. Red Neuronal Perceptrón

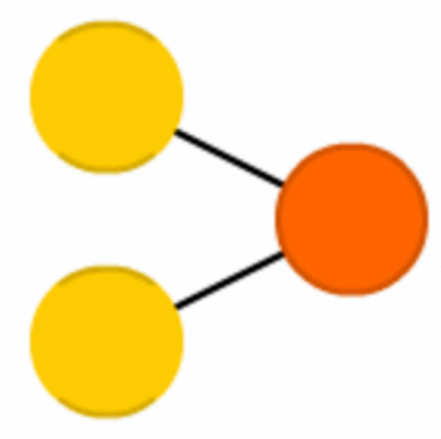

Ilustración 15 Representación gráfica de Perceptrón (21)

Es el modelo más simple y antiguo de neurona. Toma algunas entradas, las resume, aplica la función de activación y las pasa a la capa de salida.

3.5.1.3.2. Red Neuronal Feedforward (FF)

Una Red Neuronal FeedForward es una red neuronal artificial en la que las conexiones entre nodos no forman un ciclo. El modelo *feedforward* es la forma más simple de red neuronal ya que la información solo se procesa en una dirección. Si bien los datos pueden pasar a través de múltiples nodos ocultos, siempre se mueven en una dirección y nunca hacia atrás.

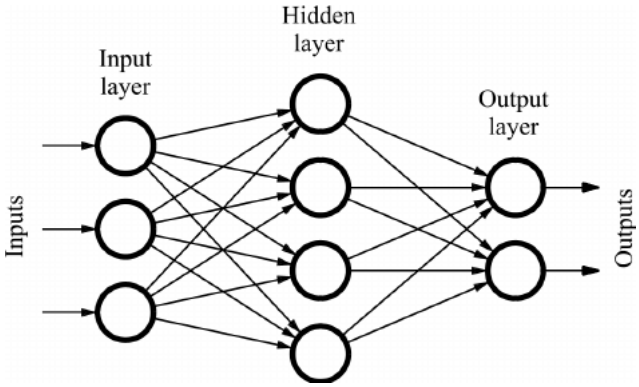

Ilustración 16 Esquema de una red neuronal FeedForward (21)

Una red neuronal *feedforward* se ve comúnmente en su forma más simple como un perceptrón de una sola capa. En este modelo, una serie de entradas ingresan a la capa y se multiplican por los pesos. Luego, cada valor se suma para obtener una suma de los valores de entrada ponderados. El perceptrón de una sola capa es un modelo importante de redes neuronales *feedforward* y se usa a menudo en tareas de clasificación. Además, los perceptrones de una sola capa pueden incorporar aspectos del aprendizaje automático. Usando una propiedad conocida como la regla delta, la red neuronal puede comparar las salidas de sus nodos con los valores previstos, lo que permite que la red ajuste sus pesos a través del entrenamiento para producir valores de salida más precisos. Este proceso de formación y aprendizaje produce una forma de descenso gradual. En perceptrones multicapa, el proceso de actualización de pesos es casi análogo, sin embargo, el proceso se define más específicamente como *backpropagation*. En tales casos, cada capa oculta dentro de la red se ajusta de acuerdo con los valores de salida producidos por la capa final.

3.5.1.3.3. Red Neuronal Feedforward profunda (DFF)

Las redes neuronales DFF son redes FF, pero con más de una capa oculta. Al entrenar una FF tradicional, solo se pasa una pequeña cantidad de error a la capa anterior. Debido a eso, apilar más capas conduce a un crecimiento exponencial de los tiempos de entrenamiento, lo que hizo que los DFF fueran bastante poco prácticos. Solo a principios de la década de 2000 se

desarrollaron una serie de enfoques que permitieron entrenar DFF de manera efectiva; ahora forman un núcleo de los sistemas modernos de Machine Learning, cubriendo los mismos propósitos que los FF, pero con mejores resultados. (19)

3.5.1.3.4. Red neuronal recurrente

Una red neuronal recurrente (RNN) es un tipo de red neuronal artificial que utiliza datos secuenciales o datos de series temporales. Estos algoritmos de aprendizaje profundo se usan comúnmente para problemas ordinales o temporales, como traducción de idiomas, procesamiento de lenguaje natural, reconocimiento de voz y subtítulos de imágenes; entre otros. Las redes neuronales recurrentes utilizan datos de entrenamiento para aprender. Se distinguen por su "memoria", ya que toman información de entradas anteriores para influir en la entrada y salida actual. (19)

Mientras que las redes neuronales profundas tradicionales asumen que las entradas y salidas son independientes entre sí, la salida de las redes neuronales recurrentes depende de los elementos previos dentro de la secuencia. Si bien los eventos futuros también serían útiles para determinar el resultado de una secuencia determinada, las redes neuronales recurrentes unidireccionales no pueden tener en cuenta estos eventos en sus predicciones.

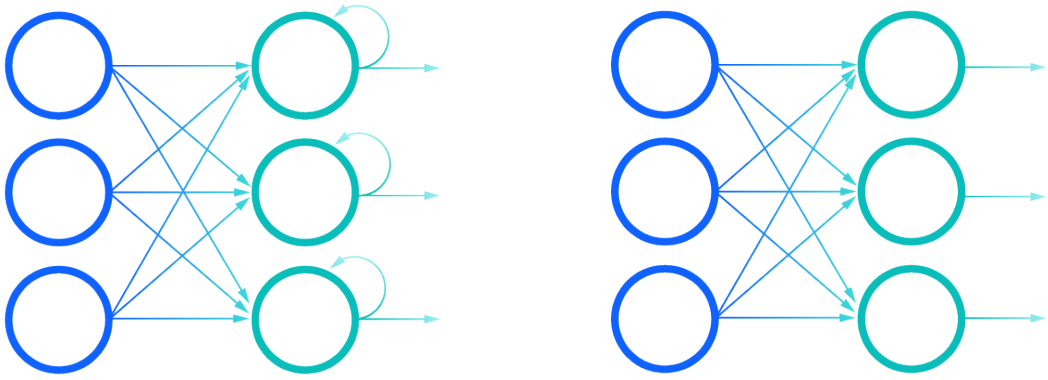

Ilustración 17 Del lado izquierdo, una red RNN, del lado derecho una red FF (19)

Otra característica distintiva de las redes recurrentes es que comparten parámetros en cada capa de la red. Mientras que las redes feedforward tienen diferentes pesos en cada nodo, las redes neuronales recurrentes comparten el mismo parámetro de peso dentro de cada capa de la red. Dicho esto, estos pesos aún se ajustan a través de los procesos de *backpropagation* y descenso de gradiente para facilitar el aprendizaje por refuerzo.

Las redes neuronales recurrentes aprovechan el algoritmo de retropropagación a través del tiempo (BPTT) para determinar los gradientes, que es ligeramente diferente de la retropropagación tradicional, ya que es específico de los datos de secuencia. Los principios de BPTT son los mismos que los de la retropropagación tradicional, donde el modelo se entrena a sí mismo calculando errores desde su capa de salida hasta su capa de entrada. Estos cálculos permiten ajustar los parámetros del modelo adecuadamente. BPTT difiere del enfoque tradicional en que BPTT suma los errores en cada paso de tiempo, mientras que las redes feedforward no necesitan sumar errores ya que no comparten parámetros en cada capa.

A través de este proceso, las RNN tienden a encontrarse con dos problemas, conocidos como gradientes de explosión y gradientes de desaparición. Estos problemas se definen por el tamaño del gradiente, que es la pendiente de la función de costo a lo largo de la curva de error. Cuando el gradiente es demasiado pequeño, continúa haciéndose más pequeño, actualizando los parámetros de peso hasta que se vuelven insignificantes, es decir, 0. Cuando eso ocurre, el algoritmo ya no está aprendiendo. Los gradientes explosivos se producen cuando el gradiente es demasiado grande, lo que crea un modelo inestable. En este caso, los pesos del modelo crecerán demasiado y finalmente se representarán como un valor indefinido. Una solución a estos problemas es reducir la cantidad de capas ocultas dentro de la red neuronal, eliminando parte de la complejidad en el modelo RNN.

3.5.1.3.5. Red neuronal Long Short Term Memory (LSTM)

Las redes Long Short Term Memory, generalmente llamadas simplemente "LSTM", son un tipo especial de RNN, capaces de aprender dependencias a largo plazo. Funcionan óptimamente en una gran variedad de problemas y ahora son ampliamente utilizadas.

Las redes LSTM están diseñadas explícitamente para evitar el problema de la dependencia a largo plazo. Todas las redes neuronales recurrentes tienen la forma de una cadena de módulos repetidos de red neuronal. En RNN estándar, este módulo repetitivo tendrá una estructura muy simple, como una sola capa de tanh. (22)

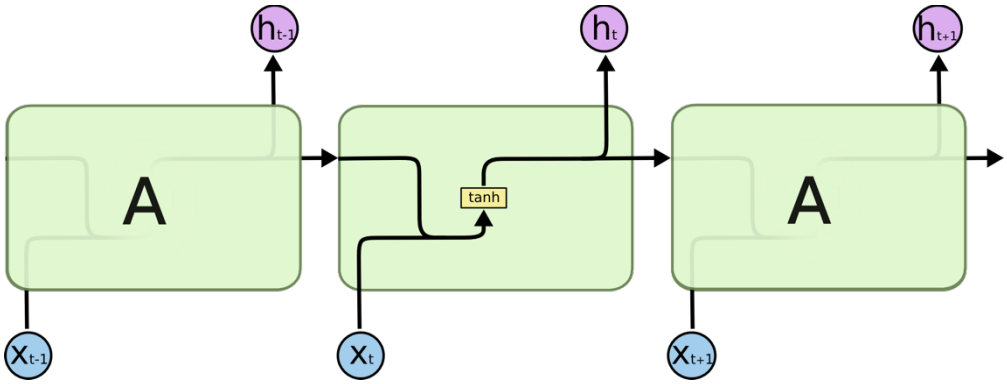

Ilustración 18 Modulo repetitivo de una red RNN estándar de una sola capa (22)

Las redes LSTM también tienen esta estructura similar a una cadena, pero el módulo repetitivo tiene una estructura diferente. En lugar de tener una sola capa de red neuronal, hay cuatro que interactúan de manera especial.

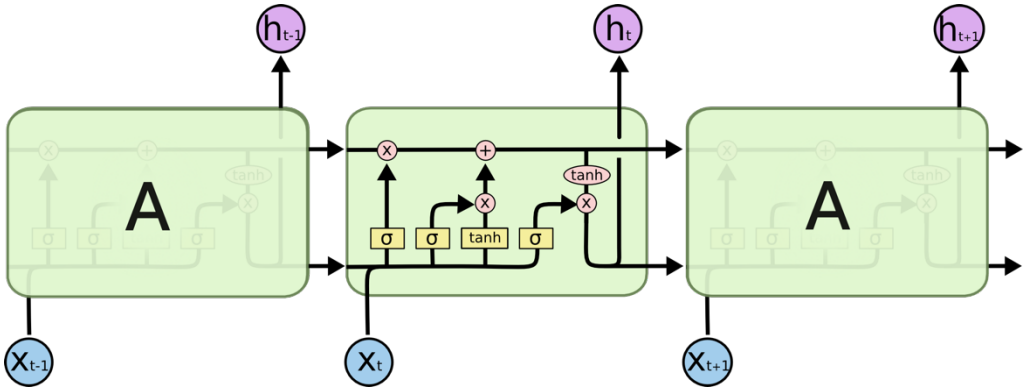

Ilustración 19 Módulo repetitivo de una red LSTM (22)

El LSTM tiene la capacidad de eliminar o agregar información al estado de la celda, cuidadosamente regulado por estructuras llamadas puertas.

Las puertas son una forma de dejar pasar información opcionalmente. Están compuestos por una capa de red neuronal sigmoidea y una operación de multiplicación puntual.

La capa sigmoidea genera números entre cero y uno, que describen la cantidad de cada componente que se debe dejar pasar. Un valor de cero significa no dejar pasar nada, mientras que un valor de uno significa dejar pasar todo. Un LSTM tiene tres de estas puertas para proteger y controlar el estado de la celda.

3.6. Computación paralela

La computación paralela se refiere al proceso de dividir problemas más grandes en partes más pequeñas, independientes y a menudo similares que pueden ejecutarse simultáneamente mediante múltiples procesadores que se comunican a través de la memoria compartida, cuyos resultados se combinan al finalizar como parte de un algoritmo general. El objetivo principal de la computación paralela es aumentar la potencia de computación disponible para acelerar el procesamiento de aplicaciones y la resolución de problemas. (23)

La infraestructura de cómputo paralelo generalmente se aloja dentro de un solo centro de datos donde se instalan varios procesadores en un rack de servidor; El servidor de aplicaciones distribuye las solicitudes de cálculo en pequeños fragmentos que luego se ejecutan simultáneamente en cada servidor.

En general, hay cuatro tipos de computación paralela, disponibles tanto de proveedores de computación paralela propietarios como de código abierto: paralelismo a nivel de bit, paralelismo a nivel de instrucción, paralelismo de tarea o paralelismo a nivel de superpalabra:

- Paralelismo a nivel de bits: aumenta el tamaño de la palabra del procesador, lo que reduce la cantidad de instrucciones que el procesador debe ejecutar para realizar una operación en variables mayores que la longitud de la palabra.
- Paralelismo a nivel de instrucción: el enfoque de hardware funciona sobre el paralelismo dinámico, en el que el procesador decide en tiempo de ejecución qué instrucciones ejecutar en paralelo; el enfoque de software funciona sobre el paralelismo estático, en el que el compilador decide qué instrucciones ejecutar en paralelo.
- Paralelismo de tareas: una forma de paralelización del código de computadora a través de múltiples procesadores que ejecuta varias tareas diferentes al mismo tiempo en los mismos datos.
- Paralelismo a nivel de superpalabra: una técnica de vectorización que puede explotar el paralelismo del código en línea.

Las aplicaciones paralelas generalmente se clasifican como paralelismo de grano fino, en el que las subtareas se comunicarán varias veces por segundo; paralelismo de grano grueso, en el que las subtareas no se comunican varias veces por segundo; o paralelismo tímido, en el que las subtareas rara vez o nunca se comunican. El mapeo en computación paralela se usa para resolver problemas paralelos al aplicar una operación simple a todos los elementos de una secuencia sin requerir comunicación entre las subtareas.

La popularización y evolución de la computación paralela en el siglo XXI se produjo en respuesta a la escalada de frecuencia del procesador que golpeó la pared de poder. Los aumentos en la frecuencia aumentan la cantidad de energía utilizada en un procesador, y escalar la frecuencia del procesador ya no es factible después de cierto punto; por lo tanto, los programadores y fabricantes comenzaron a diseñar software de sistemas paralelos y a producir procesadores de bajo consumo con múltiples núcleos para abordar el problema del consumo de energía y el sobrecalentamiento de las unidades centrales de procesamiento. (23)

La importancia de la computación paralela continúa creciendo con el uso cada vez mayor de procesadores multinúcleo y de la Unidad de Procesamiento Gráfico (GPU, del inglés Graphic Processing Unit). Las GPU funcionan junto con las CPU para aumentar el rendimiento de los datos y la cantidad de cálculos simultáneos dentro de una aplicación. Usando el poder del paralelismo, una GPU puede completar más trabajo que una CPU en un período de tiempo determinado.

3.6.1. Arquitecturas paralelas

La arquitectura de computadoras paralelas existe en una amplia variedad de computadoras paralelas, clasificadas según el nivel en el que el hardware admite el paralelismo. Las técnicas de programación y la arquitectura informática en paralelo trabajan juntas para utilizar eficazmente estas máquinas. Las clases de arquitecturas de computadoras paralelas incluyen:

- Computación multinúcleo: un procesador multinúcleo es un circuito integrado de procesador de computadora con dos o más núcleos de procesamiento separados, cada uno de los cuales ejecuta instrucciones de programa en paralelo. Los núcleos se integran en múltiples matrices en un solo paquete de chip o en una única matriz de circuito integrado, y pueden implementar arquitecturas como multiproceso, superescalar, vectorial o VLIW. Las arquitecturas multinúcleo se clasifican como homogéneas, que incluye solo núcleos idénticos, o heterogéneas, que incluye núcleos que no son idénticos.
- Multiprocesamiento simétrico: arquitectura de hardware y software de computadora multiprocesador en la que dos o más procesadores homogéneos independientes están controlados por una sola instancia de sistema operativo que trata a todos los

procesadores por igual y está conectado a una sola memoria principal compartida con acceso completo a todos los recursos comunes y dispositivos. Cada procesador tiene una memoria caché privada, se puede conectar mediante redes de malla en el chip y puede trabajar en cualquier tarea sin importar dónde se encuentren los datos para esa tarea en la memoria.

- Computación distribuida: los componentes del sistema distribuido están ubicados en diferentes computadoras en red que coordinan sus acciones comunicándose a través del protocolo HTTP puro, conectores tipo RPC y colas de mensajes. Las características significativas de los sistemas distribuidos incluyen fallas independientes de componentes y concurrencia de componentes. La programación distribuida generalmente se clasifica como arquitecturas cliente-servidor, de tres niveles, de n niveles o de igual a igual. Hay mucha superposición en computación distribuida y paralela y los términos a veces se usan indistintamente.
- Computación paralela masiva: se refiere al uso de numerosas computadoras o procesadores de computadora para ejecutar simultáneamente un conjunto de cálculos en paralelo. Un enfoque implica la agrupación de varios procesadores en un grupo de computadoras centralizado y fuertemente estructurado. Otro enfoque es la computación grid, en la que muchas computadoras ampliamente distribuidas trabajan juntas y se comunican a través de Internet para resolver un problema en particular.

Otras arquitecturas de computadoras paralelas incluyen computadoras paralelas especializadas, computación en clúster, computación en cuadrícula, procesadores vectoriales, circuitos integrados de aplicaciones específicas, computación de propósito general en unidades de procesamiento de gráficos (GPGPU) y computación reconfigurable con matrices de puertas programables en campo. La memoria principal en cualquier estructura de computadora paralela es memoria distribuida o memoria compartida.

3.6.2. GPU

La unidad de gráfica de procesamiento, o GPU, se ha convertido en uno de los tipos de tecnología informática más importantes, tanto para la informática personal como para la empresarial. Diseñado para el procesamiento en paralelo, la GPU se utiliza en una amplia gama de aplicaciones, incluida la representación de gráficos y videos (24). Aunque son más conocidas

por sus capacidades en los juegos, las GPU se están volviendo más populares para su uso en la producción creativa y la inteligencia artificial (IA). (24)

Las GPU se diseñaron originalmente para acelerar la representación de gráficos 3D. Con el tiempo, se volvieron más flexibles y programables, mejorando sus capacidades. Esto permitió a los programadores de gráficos crear efectos visuales más interesantes y escenas realistas con técnicas avanzadas de iluminación y sombreado. Otros desarrolladores también comenzaron a aprovechar el poder de las GPU para acelerar drásticamente las cargas de trabajo adicionales en computación de alto rendimiento (HPC), aprendizaje profundo y más.

3.6.2.1. GPU y CPU

La GPU evolucionó como un complemento de su prima cercana, la CPU (unidad central de procesamiento). Si bien las CPU han seguido brindando aumentos de rendimiento a través de innovaciones arquitectónicas, velocidades de reloj más rápidas y la adición de núcleos, las GPU están diseñadas específicamente para acelerar las cargas de trabajo de gráficos de computadora. Al comprar un sistema, puede ser útil conocer el papel de la CPU frente a la GPU para que pueda aprovechar al máximo ambos. (24)

3.6.3. Computación paralela y machine learning

Para cualquier red neuronal, la fase de entrenamiento del modelo de aprendizaje profundo es la tarea que requiere más recursos computacionales. Durante el entrenamiento, una red neuronal recibe entradas, que luego se procesan en capas ocultas utilizando pesos que se ajustan durante el entrenamiento y luego el modelo arroja una predicción. Los pesos se ajustan para encontrar patrones con el fin de hacer mejores predicciones. Ambas operaciones son esencialmente multiplicaciones de matrices. Una simple multiplicación de matrices es representada en la Ilustración 20.

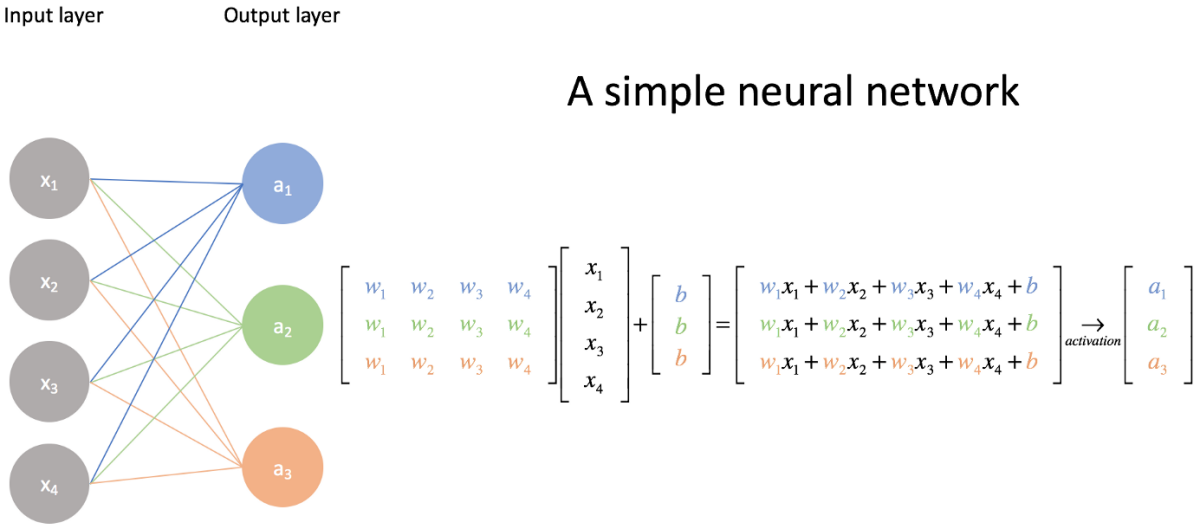

Ilustración 20 Red neuronal simple como multiplicación de matrices (25)

Sí la red neuronal tiene alrededor de 10, 100 o incluso 100.000 parámetros. Una computadora aún podría manejar esto en cuestión de minutos, o incluso horas como máximo.

Pero, cuando la red neuronal empieza a contener parámetros en el orden de los miles de millones, llevaría años entrenar este tipo de sistemas empleando el enfoque tradicional. Por lo tanto, es inviable llevar a cabo este proceso en una computadora tradicional con tan solo un CPU.

Los modelos de aprendizaje profundo se pueden entrenar más rápido ejecutando todas las operaciones al mismo tiempo en lugar de una tras otra. Esto se puede lograr utilizando una GPU para entrenar el modelo. Las GPU están optimizadas para entrenar inteligencia artificial y modelos de aprendizaje profundo, ya que pueden procesar múltiples cálculos simultáneamente. Tienen una gran cantidad de núcleos, lo que permite un mejor cálculo de múltiples procesos paralelos. Además, los cálculos en el aprendizaje profundo deben manejar grandes cantidades de datos, lo que hace que el ancho de banda de la memoria de una GPU sea el más adecuado.

Hay algunos parámetros decisivos para determinar si usar una CPU o una GPU para entrenar un modelo de aprendizaje profundo:

- Ancho de banda de memoria: El ancho de banda es una de las razones principales por las que las GPU son más rápidas para la computación que las CPU. Con grandes conjuntos de datos, la CPU ocupa mucha memoria mientras entrena el modelo.

- Tamaño del conjunto de datos: Entrenar un modelo en aprendizaje profundo requiere un gran conjunto de datos, de ahí las grandes operaciones computacionales en términos de memoria. Para calcular los datos de manera eficiente, una GPU es una opción óptima. Cuanto mayor sean los cálculos, mayor será la ventaja de una GPU sobre una CPU.
- Optimización: Las tareas de optimización poseen menor complejidad en la CPU. Los núcleos de CPU, aunque menos, son más potentes que miles de núcleos de GPU. Cada núcleo de CPU puede funcionar con diferentes instrucciones (arquitectura MIMD), mientras que los núcleos de GPU, que generalmente se organizan en bloques de 32 núcleos, ejecutan la misma instrucción en un momento dado en paralelo (arquitectura SIMD). La paralelización en redes neuronales densas es difícil dado el esfuerzo que requiere. Por lo tanto, las técnicas de optimización complejas son más difíciles de implementar en una GPU que en una CPU.

3.7. Fenómeno meteorológico de la helada

3.7.1. Definición de helada

De acuerdo al criterio agro meteorológico, la helada ocurre cuando la temperatura del aire que se siente sobre la temperatura terrestre alcanza una temperatura por debajo de los 0°C provocando la muerte de los tejidos vegetales. La observación de la temperatura a la que ocurre el fenómeno se realiza en el termómetro de mínima, instalado a 1,5 metros de altura del suelo y, claramente, sin registrar la intensidad, duración y origen de la helada.Las heladas se presentan de acuerdo al proceso físico (radiación) que consiste en la pérdida de calor que sufren las plantas y el suelo durante la noche. Para que la helada se produzca, se deben dar determinadas condiciones ambientales locales, como la ausencia de vientos, cielo despejado, sequedad atmosférica e inversión de la temperatura en el aire próximo al suelo. Esta última condición ocasiona la presencia de temperaturas más bajas en los niveles inferiores, donde se producen los mayores daños, mientras que en los niveles superiores se producen daños mínimos o nulos. Hay que recordar que en condiciones normales la temperatura del aire va disminuyendo conforme se incrementa la altura. El estudio del fenómeno de la inversión de temperatura tiene su importancia, sobre todo para una aplicación racional de los métodos de combate del efecto de heladas.La denominación de heladas blancas y negras obedece a que, en las primeras, los cultivos y el suelo presentan un aspecto blanquecino debido al hielo depositado sobre ellos; y

en la segunda, debido a que la necrosis (muerte de los tejidos) da una apariencia negruzca a los cultivos. (26)

3.7.2. Elementos meteorológicos que afectan la formación de heladas

Las condiciones atmosféricas están representadas por elementos meteorológicos como temperatura, precipitación, humedad, dirección y velocidad del viento, presión, nubosidad, radiación solar, humedad y visibilidad. Estos cambian de un lugar a otro y con el tiempo. Los principales factores meteorológicos que afectan la formación de escarcha son el viento, la nubosidad, la humedad y la radiación solar.

La temperatura del aire disminuye al aumentar la distancia desde la superficie de la Tierra. Sin embargo, a medida que aumenta la altitud la temperatura aumenta, provocando una inversión térmica. Diversas condiciones climáticas provocan inversiones de temperatura. Cuando ocurre una inversión de temperatura, la capa de aire es barrida por otras capas descendentes y capas más frías. Este fenómeno aparece principalmente en el valle, en invierno y está asociado con los cielos despejados y las bajas temperaturas cerca de la superficie de la Tierra. (26)

3.7.2.1. Viento

El viento es fundamental para que se desarrolle una helada, pues cuando hay corrientes de aire se mezcla el aire frío, que se encuentra cercano al suelo, con el más caliente que está en niveles superiores, lo que hace más difícil el desarrollo de una helada. Por tanto, una de las condiciones que favorece la ocurrencia de heladas es la ausencia de viento.

3.7.2.2. Nubosidad

Las nubes son extensos conjuntos de pequeñas gotas de agua y cristales de hielo suspendidos en el aire. Se forman cuando el vapor de agua presente en el aire llega a los niveles altos de la atmósfera y se condensa porque la temperatura es más baja. Cuando el cielo está cubierto por nubes, éstas disminuyen la pérdida de calor del suelo por radiación hacia la atmósfera y devuelven parte de ese calor a la Tierra. Para que ello ocurra, la temperatura del aire en movimiento debe ser mayor a la del punto de rocío (la temperatura a la cual el aire no admite más humedad). Cuando sigue descendiendo la temperatura puede llegar a los 0°C y el vapor de agua que contiene produce una capa delgada de hielo en la superficie de la Tierra, que se conoce como escarcha blanca.

3.7.2.3. Humedad atmosférica

Cuando disminuye la temperatura a los 0° C o menos, y el viento es escaso, el vapor de agua contenido en el aire, se condensa; si la humedad es abundante, ésta produce niebla y cuando tiene poco contenido de humedad, se forma la helada. Por ello una gran humedad atmosférica reduce la probabilidad de ocurrencia de heladas. Cuando se presenta una helada, en los cuerpos de agua de una zona y en objetos sobre el terreno se pueden formar capas de hielo.

3.7.2.4. Radiación Solar

Una cantidad de radiación solar es absorbida por la superficie de la Tierra y otra es devuelta desde su superficie a la atmósfera (radiación reflejada). Durante el día, el suelo retiene el calor y durante la noche lo pierde; estos procesos dependen de la nubosidad y del viento que existan sobre ciertas regiones del planeta. Cuando los días son más cortos y las noches más largas, aumenta la ocurrencia de heladas; aunque exista una menor acumulación de calor en el suelo, habrá un mayor tiempo para que se transmita hacia el aire.

3.7.3. Daños de las heladas sobre las plantas

Los efectos dañinos de las heladas sobre los cultivos no siempre son los mismos, varían en su intensidad de acuerdo a los siguientes factores: especie y variedad considerada, tipo de órgano expuesto, etapa fenológica, contenido hídrico de la planta, intensidad de la helada, duración de la helada, temperatura de la planta y el órgano. (27)

Se establecen 4 grados de daños por heladas, que involucran la acción de los factores anteriores.

- El frío daña o mata órganos vegetativos, tales como hojas y tallos, perturbando las funciones de los órganos restantes.
- La helada destruye un gran porcentaje de flores, impidiendo así que muchas de ellas se transforman en frutos.
- La baja temperatura destruye los frutos en formación, y los que sobreviven resultan mal formados.
- El frío es lo suficientemente intenso y prolongado como para provocar la muerte de la planta completa.

3.7.4. Métodos de control o mitigación recomendados

3.7.4.1. Métodos pasivos.

Son de carácter preventivo, de cómo evitar totalmente el período de las heladas. Es lo más eficiente y menos costoso.

- Ubicación en la zona de cultivo
    - Evitar sembrar en lugares bajos, pues constituyen cauces naturales de las masas de aire frío.
    - Sembrar en sitios donde el aire frío tiende a dispersarse, como las zonas más altas.
    - En pendientes y terrenos ligeramente inclinados, cualquier obstáculo, natural o artificial, que se opongan a la libre circulación del aire frío, aumenta el riesgo de heladas en los sitios por encima de las barreras.
- Fecha de siembra y temporada de cultivo: Sembrar en épocas fuera del período de las heladas.
- Manejo de suelo
    - El suelo debe de mantenerse húmedo, libre de malezas, lisos y compactos.
    - En suelos claros y arenosos, las heladas son menos fuertes.
- Manejo de plantas
    - Buenas condiciones fitosanitarias, pues así resisten más el frío. Una buena poda ayuda a la planta a librarse del efecto de las heladas.
    - Buen aporque. Utilizando productos químicos (ácidos orgánicos) se puede retrasar la floración de las plantas, aumentando su reposo invernal para escapar de las heladas.

3.7.4.2. Métodos directos

Con estos métodos se pretende, por un lado, disminuir la pérdida de calor y, por el otro, aportar el calor necesario para evitar la ocurrencia de una helada en un área dada. Se fundamenta este control en lograr aumentar 2 grados centígrados sobre la temperatura que afecta a la planta (temperatura letal congelante) (26).

- Coberturas: Pueden utilizarse plásticos. Se pretende con esto evitar la pérdida de calor del suelo. Otros materiales pueden ser: paja, papel, tela, ramas, pasto, vidrio. No se presta para cultivos de porte alto, además es costoso.
- Riegos: Si se tiene sistema de riego, se facilita este control, pues es uno de los métodos más efectivos que se conoce actualmente para combatir las heladas. Este método requiere una inversión inicialmente alta. Debe cuidarse en no aplicar riego en caso de vientos fuertes y frescos, porque el efecto sería contrario al fin que se persigue.
- Calefacción: Es el más antiguo que se conoce. Bien manejado es un sistema práctico, eficaz y económico. Se lo utiliza con mucha frecuencia en frutales caducifolios y cítricos. Todo radica en poder generar humo sobre el cultivo, evitando la pérdida libre de calor del suelo hacia la atmósfera. El material a quemar puede ir desde paja, ramas, llantas viejas, carbón mineral, entre otros. También existen calefactores preparados para el efecto, de lo contrario se hace un hoyo en el suelo y se distribuye el material a ser quemado. Hacer humo -y no fuego- es esencial para una buena protección.
- Ventilación: Se basa en el principio de poder mezclar el aire en una noche de inversión térmica con molinos de viento. Actualmente, en cultivos muy rentables se utilizan helicópteros para obtener el efecto de la mezcla.

3.7.5. Importancia de la predicción de heladas

La predicción precisa de heladas puede potencialmente reducir el daño por heladas, ya que proporciona una oportunidad a los agricultores de prepararse contra ellas.

Es importante poder predecir cuándo cae la temperatura hasta un valor crítico para poner en marcha los métodos activos de protección contra las heladas. Activar y detener la protección a la temperatura apropiada es importante ya que evita las pérdidas que resultan de poner en marcha demasiado tarde, además ahorra energía al reducir el tiempo de funcionamiento de los distintos métodos.

4. Metodología

4.1. Introducción

Como se explicita en la Ilustración 1 la metodología a utilizar será CRISPDM donde los primeros cuatro módulos (de la representación Data Science Life Cycle) hacen referencia al preprocesamiento de datos tarea que lleva la mayor parte del esfuerzo de facilitar la toma de decisiones desde el conocimiento extraído y subyacente en los datos, para los restantes módulos ser parte esencial del desarrollo de la aplicación, presentándose las herramientas que permitieron llevar adelante el presente trabajo de tesis.

4.2. Herramientas de software utilizadas

4.2.1. Python.

El lenguaje de programación Python, es un lenguaje interpretado, de alto nivel, multipropósito, multiparadigma, que ha tenido un gran desarrollo en el último tiempo, como muestran las estadísticas de uso elaboradas mensualmente por TIOBE (28)y GitHub (29), y que ubican al lenguaje entre los tres primeros lenguajes de programación de mayor uso. Python es el lenguaje elegido por desarrolladores para aplicaciones en Inteligencia Artificial y en aprendizaje de máquina, dado que provee numerosas bibliotecas para estas áreas de conocimiento, así como también para visualización de datos. Python soporta múltiples paradigmas de programación: estructurado, orientado a objetos y funcional.

4.2.1.1. Bibliotecas

4.2.1.1.1. Numpy

NumPy es una biblioteca fundamental para la computación científica en Python. Proporciona un objeto de matriz multidimensional, varios objetos derivados y una variedad de funciones para operaciones rápidas en matrices, que incluyen manipulación matemática, lógica, de formas, clasificación, selección, E / S, transformaciones discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria, entre otras.

En el núcleo del paquete NumPy, está el objeto ndarray. Este encapsula matrices n-dimensionales de tipos de datos homogéneos, con muchas operaciones que se realizan en código compilado para el rendimiento. Hay varias diferencias importantes entre las matrices NumPy y el estándar de Python:

Las matrices NumPy tienen un tamaño fijo en el momento de la creación, a diferencia de las listas de Python (que pueden crecer dinámicamente). Cambiar el tamaño de un ndarray creará una nueva matriz y eliminará la original.
Se requiere que todos los elementos de una matriz NumPy sean del mismo tipo de datos y, por lo tanto, tendrán el mismo tamaño en la memoria. La excepción: uno puede tener matrices de objetos (Python, incluido NumPy), lo que permite matrices de elementos de diferentes tamaños.
Las matrices NumPy facilitan operaciones matemáticas avanzadas y de otro tipo en grandes cantidades de datos. Por lo general, estas operaciones se ejecutan de manera más eficiente y con menos código de lo que es posible usando las operaciones integradas de Python.
Muchas otras bibliotecas empleadas para Ciencia de Datos, como puede ser Pandas o Matplotlib, hacen uso de Numpy. Es por esto que muchas veces no se utilizan directamente las funciones de Numpy, pero se utilizan indirectamente a través de estas otras bibliotecas, dada su gran rapidez y eficiencia para realizar los cálculos.

4.2.1.1.2. Pandas

Es una biblioteca de Python que provee el robusto objeto Data Frame, que permitirá hacer el análisis de datos más rápido y sencillo.
Entre las características de Pandas, se destacan:

- Herramientas para leer y escribir datos entre estructuras de datos en memoria y diferentes formatos: CSV y archivos de texto, Microsoft Excel, bases de datos SQL y el rápido formato HDF5;
- Manejo integrado de datos faltantes, facilitando el trabajo con conjuntos de datos incompletos.
- *Reshaping* y pivoteo flexible de conjuntos de datos;
- Facilita el indexado en grandes conjuntos de datos.
- Las columnas se pueden insertar y eliminar de dataframes para mutabilidad de tamaño.
- Agregar o transformar datos con un potente grupo por motor que permite operaciones de división, aplicación y combinación en conjuntos de datos.
- Fusión y unión de conjuntos de datos de alto rendimiento.
- Facilita el trabajo con series de tiempo. Provee operaciones para trabajar con tiempos y fechas.
- Rendimiento optimizado.

- Python con pandas se usa en una amplia variedad de dominios académicos y comerciales, que incluyen finanzas, neurociencia, economía, estadísticas, publicidad, análisis web, entre otros.

4.2.1.1.3. Matplotlib

Matplotlib es una biblioteca para hacer gráficos 2D de matrices en Python. Aunque tiene su origen en la emulación de los comandos gráficos de MATLAB, es independiente de MATLAB y se puede utilizar de forma declarativa y orientada a objetos. Aunque Matplotlib está escrito principalmente en Python puro, hace un uso intensivo de NumPy para proporcionar un buen rendimiento incluso para matrices grandes.

4.2.1.1.4. Seaborn

Seaborn es una biblioteca para hacer gráficos estadísticos en Python. Se basa en Matplotlib y se integra estrechamente con las estructuras de datos de pandas.

Seaborn ayuda a explorar y comprender datos. Sus funciones para graficar funcionan sobre dataframes y matrices que contienen conjuntos de datos completos y realizan internamente el mapeo semántico y la agregación estadística necesarios para producir gráficos informativos. Su API declarativa orientada a conjuntos de datos le permite concentrarse en lo que significan los diferentes elementos de sus gráficos, en lugar de en los detalles de cómo dibujarlos.

4.2.1.1.5. Pyplot & Cufflinks

Plotly es una biblioteca de visualización de datos interactiva en Python, que funciona a través de un framework Dash. Está desarrollada sobre Flask, un framework de desarrollo web backend en Python muy relevante, y el frontend está desarrollado en ReactJS, el framework de desarrollo web frontend más utilizado en la actualidad. Esto es importante dado que siendo una aplicación web puede presentar incompatibilidades con ciertas versiones de determinados navegadores, al utilizar estas tecnologías mencionadas reducimos las posibilidades de incompatibilidad al mínimo.

Plotly se abstrae de todas las tecnologías y protocolos necesarios para construir una aplicación web interactiva, permitiendo un rápido desarrollo. Las aplicaciones creadas con Dash son renderizadas en el navegador. Se despliegan las apps en los servidores y luego se las comparte a través de las URLs. Dado que son vistas en el navegador, Dash es multiplataforma y listo para ver desde dispositivos móviles.

Plotly es adecuado para desarrollar aplicaciones de visualización de datos, con interfaces altamente personalizables, escribiendo el código únicamente en Python. Esto lo hace particularmente adecuado para el trabajo que implique procesamiento de datos en este lenguaje. Cufflinks, por su lado, es otra biblioteca de pandas que vincula Plotly con Pandas, para facilitar la realización de gráficos visuales interactivos.

4.2.1.1.6. Scikit Learn

Scikit Learn es una biblioteca que contiene funciones para aprendizaje de máquina en Python. Proporciona una variedad de herramientas altamente eficientes para el aprendizaje automático y el modelado estadístico, incluyendo clasificación, regresión, agrupamiento y reducción de la dimensionalidad a través de una interfaz en Python. Esta biblioteca se basa en NumPy, SciPy y Matplotlib.

4.2.1.1.7. Tensorflow

TensorFlow es una plataforma integral de código abierto para crear aplicaciones de aprendizaje automático. Es una biblioteca utilizada para realizar diversas tareas enfocadas en el entrenamiento e inferencia de redes neuronales profundas. Permite a los desarrolladores crear aplicaciones de aprendizaje automático utilizando diferentes herramientas, bibliotecas y recursos de la comunidad. Tensorflow provee bibliotecas para diferentes lenguajes de programación, entre ellos, Python.

4.2.1.1.8. Keras

Keras es una biblioteca de software de código abierto que proporciona una interfaz Python para redes neuronales artificiales. Keras actúa como una interfaz para la biblioteca TensorFlow, abstrayéndose de ésta y reduciendo su nivel de complejidad en lo que respecta a código y facilidad para crear y manipular redes neuronales.

Está diseñada para permitir la rápida manipulación de redes neuronales profundas, se enfoca en ser fácil de usar, modular y extensible. Fue desarrollado como parte del esfuerzo de investigación del proyecto ONEIROS (Sistema Operativo de Robot Inteligente Neuroelectrónico de Extremo Abierto), y su principal autor y mantenedor es François Chollet, un ingeniero de Google.

4.2.2. Visual Studio Code

Es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, permitiendo a los usuarios cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto, aunque la descarga oficial está bajo software privativo e incluye características personalizadas por Microsoft.

Es compatible con varios lenguajes de programación, entre ellos, Python. Visual Studio Code, provee, además diferentes funcionalidades para facilitar el desarrollo en este lenguaje, además de poseer complementos libres para descargar especializados en el procesamiento de datos, como lo puede ser el complemento de Jupyter Notebooks. También incluye la funcionalidad de cambio de entorno de desarrollo, lo cual posibilita seleccionar el conjunto de bibliotecas con el cual se desea trabajar.

4.2.3. Anaconda

Anaconda es una Suite de código abierto que abarca una serie de aplicaciones, librerías y conceptos diseñados para el desarrollo de la Ciencia de datos con Python. En líneas generales Anaconda Distribution es una distribución de Python que funciona como un gestor de entorno, un gestor de paquetes y que posee una colección de más de 720 paquetes de código abierto.

Anaconda Navigator es una interfaz gráfica de usuario de escritorio incluida en la distribución de Anaconda que permite a los usuarios iniciar aplicaciones y administrar paquetes, entornos y canales de Conda sin usar comandos de línea de comandos. Navigator puede buscar paquetes en Anaconda Cloud o en un repositorio local de Anaconda, instalarlos en un entorno, ejecutar los paquetes y actualizarlos. Está disponible para Windows, macOS y Linux. Es a través de Anaconda que se instalan las diferentes bibliotecas mencionadas anteriormente.

4.2.4. Jupyter Notebooks

La aplicación Jupyter Notebook es una aplicación cliente - servidor que permite editar y ejecutar documentos de notebook a través de un navegador web. La aplicación Jupyter Notebook se puede ejecutar en un escritorio local que no requiera acceso a Internet o se puede instalar en un servidor remoto y acceder a través de Internet.

Los documentos de cuadernos (o "notebooks") son documentos producidos por la aplicación Jupyter Notebook, que contienen código (por ejemplo, Python) y elementos de texto enriquecido (párrafo, ecuaciones, figuras, enlaces, etc.). Los documentos de notebook son tanto documentos legibles por humanos que contienen la descripción del análisis y los resultados (cifras, tablas, etc.) como documentos ejecutables que se pueden ejecutar para realizar análisis de datos.

4.2.5. Knime

Knime Analytics es una conocida plataforma de aprendizaje automático en línea, gratuita, de código abierto que proporciona análisis, integración e informes de datos de extremo a extremo. Con la Plataforma KNIME Analytics, los científicos de datos pueden habilitar fácilmente la creación de flujos de trabajos visuales a través de una interfaz gráfica, arrastrando y soltando los diferentes nodos que permiten la construcción del pipeline de DS. No requiere conocimientos de programación. Para crear flujos de trabajo, un usuario puede elegir entre más de 2000 nodos. KNIME Analytics permite a los desarrolladores realizar varias acciones; desde E/S básica hasta manipulaciones de datos, transformaciones y minería de datos. La mejor parte de KNIME Analytics es que consolida todo el proceso de funciones en un solo flujo de trabajo.

4.3. Datos utilizados en el desarrollo de este trabajo

Los datos se han obtenido de dos estaciones meteorológicas, ubicada una de ellas en el predio de la Estación Experimental Agropecuaria San Juan del Instituto de Tecnología Agropecuaria (INTA) en el departamento de Pocito y la otra situada en el Establecimiento San Francisco S.A. (explotación privada), ubicado en la localidad de Cañada Honda, departamento Sarmiento, provincia de San Juan; existiendo entre ellas distancia de 37km aproximadamente. Ambas estaciones registran cada 10 minutos los valores de distintas variables meteorológicas, como temperatura máxima, temperatura mínima, humedad, velocidad del viento, precipitación, presión atmosférica, radiación solar, punto de rocío, entre otras. Proveyendo datos que son la fuente de información para el análisis de fenómenos meteorológicos en la zona.

Ilustración 21 Ubicación de las estaciones INTA y San Francisco

4.3.1. Características de los datos

4.3.1.1. Conjunto de datos de la estación San Francisco

El conjunto de datos tiene un total de 299671 registros y 42 columnas, las cuales refieren a las distintas variables, como son fecha y hora, temperatura, humedad, punto de rocío, radiación solar, lluvia, dirección del viento, velocidad del viento, entre otros. Posteriormente se profundizará en las diferentes columnas del conjunto de datos.

Se observaron datos faltantes en las siguientes variables: dirección del viento y dirección de viento máxima, debiendo destacar que las mismas se obtienen de la dirección dominante en los últimos 30 minutos de los atributos velocidad de viento y velocidad máxima de viento respectivamente, de todas maneras, por ser variables de tipo categórico fueron filtradas atendiendo al procesamiento de los datos de tipo numérico a modelar. Para índice THSW (una forma de medir la sensación térmica) y el punto de rocío, aún siendo variables numéricas no se tuvieron en cuenta para el modelo.

Por otro lado, se pudo obtener que hay periodos donde no se captaron datos. Se encontraron siete “huecos”, de una duración promedio de 12 días y 16 horas. Posteriormente, se profundizará en cómo se abordó este problema.

4.3.1.2. Conjunto de datos de la estación INTA

4.3.1.2.1. Atributos

Las variables relevantes captadas por los sensores de esta estación son las mismas que para la estación de San Francisco.

4.3.1.2.2. Tamaño

El conjunto de datos tiene un total de 161301 registros y 35 columnas.

4.3.1.2.3. Datos faltantes

Se observaron datos faltantes en las siguientes variables: dirección del viento y velocidad del viento, aunque esta ausencia de datos se debe a que para esos registros no hubo presencia de vientos. En cuanto a la dirección dado que es una variable de tipo categórica, no se realizó ninguna acción sobre estos faltantes, y respecto a la velocidad, se colocó el valor cero.

4.4. Metodología aplicada

Durante el desarrollo del trabajo, para el tratamiento y análisis del conjunto de datos se aplicaron las etapas de la Ciencia de Datos mencionadas con anterioridad.

Inicialmente los conjuntos de datos correspondientes a los distintos años, para una misma estación fueron agrupados en un único conjunto, a fin de poder realizar distintos tipos de análisis sobre los mismos que implican comparaciones interanuales y series temporales. Se decidió dejar por separados los datos correspondientes a las dos estaciones mencionadas, INTA y San Francisco, dado que ambas contenían registros con la misma marca temporal, esto es, fueron tomados al mismo tiempo, ya que ambas captan los datos cada diez minutos, sobre los mismos años. Además de esto, ambas estaciones se encuentran distantes geográficamente, lo que implica que los valores para las mediciones no tienen por qué ser idénticos. Durante la mayor parte del trabajo, se emplearon los datos de la Estación San Francisco, dado que se verifica que tienen una cantidad notablemente inferior de registros faltantes y, por lo tanto, menor cantidad de baches temporales, aportando así, una mayor cantidad de información.

Luego de haber realizado la lectura de los datos, originalmente en formato CSV, a través de la biblioteca Pandas estos son transformados a una estructura de datos propia de dicha biblioteca, llamados *Dataframes*, esto facilitará el procesamiento de los datos.

Una vez realizado esto, se procedió a realizar un preprocesamiento sobre el conjunto de datos, para ello se ajustan los tipos correspondientes a cada registro, ya que, por ejemplo, muchas veces algunos datos numéricos, se encuentran representados como cadenas, imposibilitando la realización de determinadas operaciones. Durante esta etapa también se ajustaron las fechas al formato ISO 8601 y se ordenó el conjunto de datos cronológicamente. También se obtuvo información acerca de la cantidad de datos faltantes y se realizó un abordaje de los mismos a través de técnicas de interpolación. Asimismo, en algunas ocasiones, para el análisis posterior, se decidió recortar los datos no correspondientes a períodos de helada.

Posteriormente, pasando a la etapa de análisis exploratorio de datos, se obtuvieron distintas métricas de los conjuntos de datos, y se realizaron diferentes gráficas recurriendo a las diferentes bibliotecas de visualización mencionadas, a fin de lograr un mayor entendimiento de los datos.

Una vez realizado este análisis, se dividió el conjunto de datos con el que se esté trabajando en dos subconjuntos disjuntos, donde uno se utilizó para entrenar los modelos y el otro para probarlos, y así, determinar su rendimiento y aptitud a la hora de predecir las heladas junto con sus características.

Finalmente, se entrenaron y se evaluaron los distintos modelos, ajustando repetidamente los diferentes hiperparámetros. Los distintos modelos se compararon utilizando métricas preestablecidas facilitadas por las diferentes bibliotecas de aprendizaje de máquina así como también, matrices de confusión, haciendo una comparación de estas métricas se determina cuál es el modelo que resultará más apto para el aspecto de la predicción de helada que se quiera lograr.

5. Desarrollo

5.1. Abordaje del conjunto de datos

5.1.1. Lectura de datos

La lectura de los datos, que consisten en archivos CSV, con las características mencionadas anteriormente, se presentaron separados por años. Para realizar su lectura se utilizó la biblioteca Pandas, y en particular el método *read_csv* para realizar la lectura de cada archivo, cada archivo que se leyó, se convirtió en un Dataframe, cada uno de los dataframes leídos se insertó en un arreglo, finalmente estas estructuras de datos se consolidaron en un único dataframe mediante el método *concat*, que permite unir diferentes dataframes en uno.

5.1.2. Preprocesamiento de los datos

5.1.2.1. Fechas y orden del conjunto de datos

Para poder disponer de los datos en un formato útil y eficiente, es necesario realizar un procesamiento previo de los mismos a fin de poder ser utilizados por los modelos de aprendizaje de máquina utilizados en etapas posteriores del pipeline.

Lo primero que se realizó es transformar el formato de las fechas, ya que son captadas y almacenados por el sensor en formato DD/MM/AAAA, y para facilitar el procesamiento posterior de los datos, resulta conveniente adaptar el formato de las fechas a ISO 8601 (AAAA/MM/DD). Una vez realizado esto, se puede ordenar el dataframe utilizando el método de pandas *Dataframe.sort_values()*.

5.1.2.2. Valores no válidos y nulos

El conjunto de datos posee columnas con valores no válidos, los cuales son distintos a los valores nulos, y es necesario identificarlos ya que no pueden ser automáticamente detectados por métodos de la biblioteca Pandas.

Luego de analizar el conjunto de datos, se observó que cuando el sensor falla al captar un determinado valor para una variable meteorológica, coloca el valor *"---"* para este registro, por lo tanto, se llevó a cabo un algoritmo que encuentre estos valores y los reemplace por el valor *None*, un tipo de dato del lenguaje Python para representar los valores nulos.

Otra de las acciones necesarias que se deben llevar a cabo, es la correcta asignación de tipos de datos a todas las variables, para esto se analiza el dataframe utilizando métodos anteriormente mencionados para determinar -por ejemplo- cuando un atributo tiene un tipo especificado incorrecto, siendo el más común la asignación de tipo de dato cadena a atributos de tipo flotante o entero.

Habiendo realizado este reemplazo y considerando que ahora todos los valores que se tienen en el dataframe son válidos -más allá de que sean nulos- se puede proceder a ejecutar diferentes métodos de la biblioteca Pandas que permitirán dar un valor a estos registros nulos.

5.1.2.3. Brechas temporales en los datos

Por distintos motivos que varían desde fallas en los sensores, hasta daños en los mismos, existen periodos de tiempo en los que los sensores no pudieron captar información, produciéndose huecos en la información, los cuales claramente necesitan ser manejados de alguna forma, más aún al utilizar modelos que trabajen sobre series temporales.

Para detectar estos huecos temporales se elaboró un simple algoritmo que los detecta, informa la fecha de inicio y la fecha de fin, así como su duración.

Para el caso del conjunto de datos de la Estación San Francisco, se encontraron 7 huecos, con una duración promedio de 12 días, 16 horas y 48 minutos:

- Inicio: 2014-03-11 10:40:00 AM | Duración: 4 días, 7:40:00
- Inicio: 2017-11-01 11:50:00 AM | Duración: 0:20:00
- Inicio: 2016-01-06 01:30:00 PM | Duración: 0:30:00
- Inicio: 2016-12-01 03:10:00 PM | Duración: 0:30:00
- Inicio: 2014-01-01 12:00:00 AM | Duración: 60 días, 12:20:00
- Inicio: 2018-10-18 01:10:00 PM | Duración: 0:20:00

Se puede observar que claramente el valor que incrementa el promedio es aquel *gap* correspondiente a 60 días.

Existieron tanto registros faltantes para algunas variables, así como también períodos en los que no se capturaron datos producto de fallas en los equipos sensores de registración. Para estos casos, se empleó una interpolación lineal hacia adelante para obtener estos valores.

Para ejemplificar esta interpolación, se muestra en la Ilustración 22 Evolución temporal de temperaturas sin interpolación de datos el hueco en la serie temporal correspondiente a la temperatura más grande de todo el conjunto de datos. Se muestra la gráfica representando los datos posteriormente a realizar la interpolación. Se puede observar que claramente hay una

pérdida de información, sin embargo, esto ocurrió en un periodo de tiempo que no se considera relevante, dado que no se ubica en épocas en las que se pueda producir una helada.

Puede parecer innecesario realizar la interpolación de datos dentro de un periodo de tiempo que no será considerado, sin embargo, es necesario dado que la interpolación es uno de los primeros pasos del preprocesamiento que se realiza, luego de haber filtrado las columnas. Esto se hace para facilitar el procesamiento posterior de la información, ya que las librerías utilizadas para el abordaje de series temporales asumen que las muestras se encuentran en periodos igualmente espaciados de tiempo.

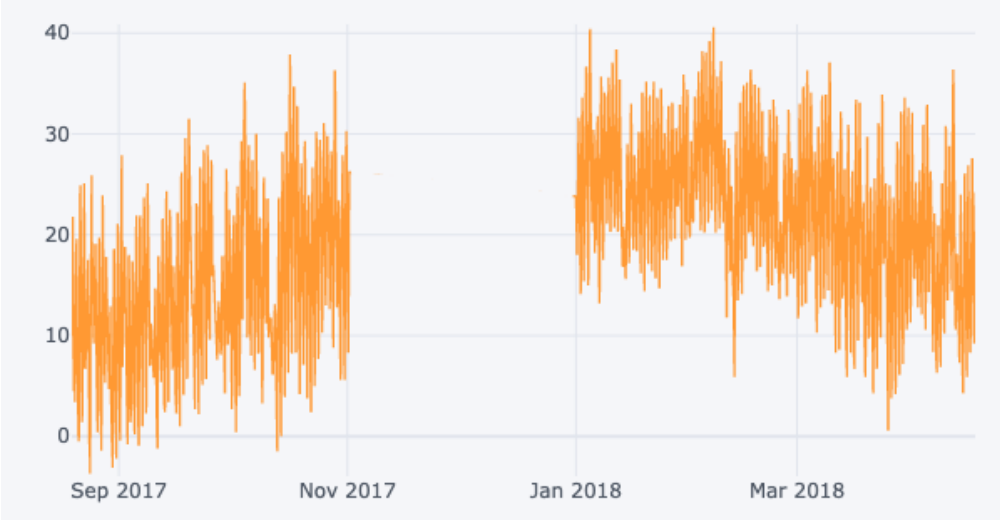

Ilustración 22 Evolución temporal de temperaturas sin interpolación de datos

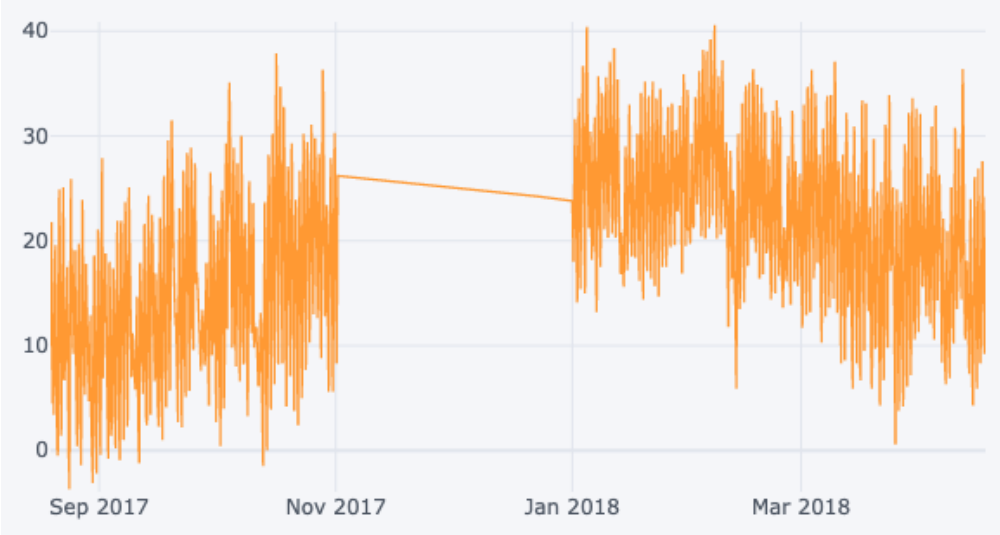

Ilustración 23 Evolución temporal de temperaturas sin interpolación de datos

5.1.3. Análisis exploratorio de datos (EDA)

Una vez realizada una limpieza de los datos en la etapa de preprocesamiento, se realizó un análisis más profundo para entender sus características y comportamiento.

Se empezó dividiendo el conjunto de datos en dos subconjuntos disjuntos. Uno que contenga todos aquellos registros que corresponden a helada, esto es, una temperatura menor a 0 grados, y, por otro lado, aquellos registros que no corresponden a helada.

Utilizando el método *describe()* de Pandas, podemos obtener resúmenes acerca de cada una de las diferentes variables.

	Temp. Ext.	Temp. Max.	Temp. Min.	Hum. Ext.	Pto. Rocio	Vel. Viento	Rec. Viento	Vel. Max.
count	13069.000000	13069.000000	13069.000000	13069.000000	13069.000000	13069.000000	13069.000000	13069.0000
mean	-2.271015	-2.143882	-2.375606	85.545336	-4.417354	0.038105	0.006418	0.443729
std	1.731287	1.746481	1.737111	7.698622	2.159726	0.332683	0.055781	1.282754
min	-9.400000	-9.300000	-9.600000	43.000000	-11.800000	0.000000	0.000000	0.000000
25%	-3.300000	-3.200000	-3.400000	82.000000	-5.800000	0.000000	0.000000	0.000000
50%	-1.900000	-1.800000	-2.000000	88.000000	-4.100000	0.000000	0.000000	0.000000
75%	-0.900000	-0.800000	-1.000000	91.000000	-2.700000	0.000000	0.000000	0.000000
max	-0.100000	1.000000	-0.100000	97.000000	-0.500000	14.500000	2.410000	22.500000

Ilustración 24 Métricas estadísticas obtenidas mediante el método describe() para el conjunto de datos de heladas

	Temp. Ext.	Temp. Max.	Temp. Min.	Hum. Ext.	Pto. Rocio	Vel. Viento	Rec. Viento
count	299397.000000	299397.000000	299397.000000	299397.000000	299397.000000	299397.000000	299397.000000
mean	17.714594	17.871213	17.551045	57.792655	7.743941	2.080659	0.348373
std	8.964899	8.962198	8.960467	22.640332	6.526865	2.933843	0.489395
min	0.000000	0.000000	-2.700000	0.000000	-35.000000	0.000000	0.000000
25%	10.900000	11.100000	10.700000	40.000000	2.800000	0.000000	0.000000
50%	17.700000	17.800000	17.500000	57.000000	7.900000	0.000000	0.000000
75%	24.400000	24.600000	24.300000	78.000000	12.653041	3.200000	0.540000
max	42.700000	42.700000	42.700000	98.000000	27.000000	33.800000	5.630000

Ilustración 25 Métricas estadísticas obtenidas mediante el método describe() para el conjunto de datos sin heladas

De Ilustración 24 e Ilustración 25 se puede hacer una comparación para lograr una mayor compresión del conjunto de datos y de las variables que pueden ser relevantes tomar como entradas de los modelos que se planteen. Por ejemplo, se puede apreciar que para la variable “Hum. Ext.”, que hace referencia a la humedad, se tiene un promedio de 57,79% en los datos sin registros de helada, mientras que se tiene un 85,54% para el *dataframe* que contiene únicamente registros de helada.

5.1.3.1. Desbalance en el conjunto de datos

Por otro lado, de estas figuras se puede apreciar que existe una cantidad de registros de helada muy inferior respecto a los de no helada. Del total de 299671 registros, tan sólo 13069 corresponden a periodos de helada, es decir, poco más del 4%. Para abordar este problema se consideraron únicamente los períodos en los que ocurren heladas.

Para esto fue necesario definir que es un periodo o época de helada, y a su vez, sustraer los datos correspondientes a este periodo, dejando de lado al resto.

Se definió como periodo de heladas el intervalo de tiempo que transcurre entre la primera y la última helada de cada año. De esta definición se desprende que la época de helada depende del año y por lo tanto no tiene una longitud fija.

En la Tabla 1 se muestra el primer y último día de helada para cada año

Tabla 1 Fechas de primera y última helada por año

Año	Primera helada	Última helada
2013	2013-05-15	2013-09-29
2014	2014-05-18	2014-09-12
2015	2015-05-03	2015-09-17
2016	2016-04-26	2016-10-21
2017	2017-05-20	2017-10-13
2018	2018-05-20	2018-10-02

De forma que, considerando únicamente los períodos de helada, la gráfica de las temperaturas resulta como se muestra en la Ilustración 26.

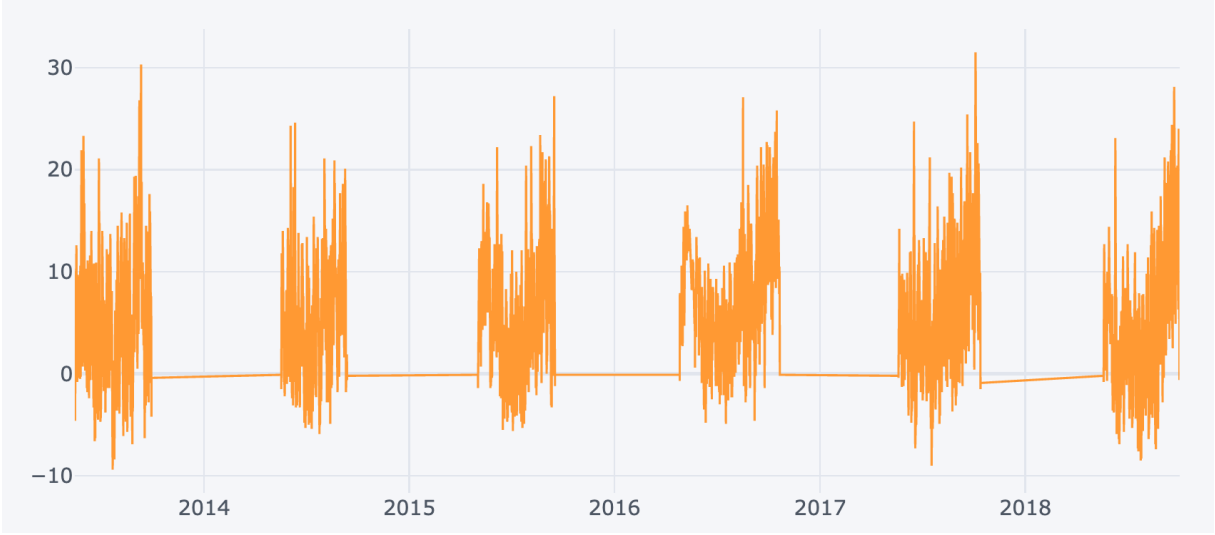

Ilustración 26 Gráfica de temperaturas recortada en periodos de helada

Si bien, como se mostró anteriormente, se realizó una interpolación de los valores para datos faltantes que posteriormente fueron eliminados, se conservó una copia del dataframe porque estos periodos de no helada serán relevantes para procesamientos abordados posteriormente en este trabajo.

También resultó interesante investigar en que horarios se concentraban las heladas, a fin de generar modelos que utilicen únicamente los datos de estos horarios y no del resto, así como también para clasificar los datos según la hora en horarios de helada y horarios de no helada.

Para esto se genera un gráfico de barras utilizando la biblioteca Pandas para obtener de forma visual en que horarios se concentran las heladas.

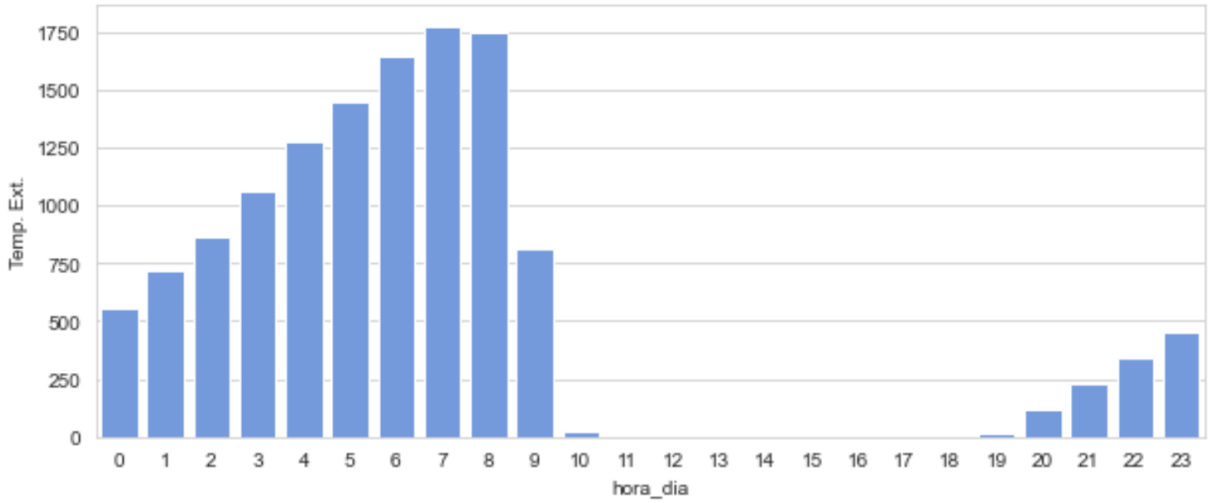

Ilustración 27 Gráfica de barras indicando cantidad de haladas por hora

Como se puede observar en la Ilustración 27, las heladas se concentran principalmente desde las 20 hasta las 9 horas, siendo la cantidad de registros de helada que se ubican fuera de este rango exigua.

Luego de haber realizado este recorte en la cantidad de registros, considerando épocas y horarios de helada, resultaron un total de 66342 registros, de los cuales 12475 corresponden a heladas, lo que significa ahora un 18% de los registros, disminuyendo considerablemente el desbalance entre los datos.

5.1.3.2. Análisis y reducción de las variables de entrada

Tal como se mencionó anteriormente, el conjunto de datos posee 32 atributos o columnas. No todos estos atributos son relevantes como variables de entrada de los modelos, ya que pueden presentar información redundante o repetida, a fin de hacer los modelos más rápidos y

eficientes, considerando el gran volumen de datos a procesar, fue necesario reducir esta cantidad de atributos, siempre manteniendo la misma cantidad de información.

Los primeros atributos a filtrar (atributos a utilizar en los siguientes pasos del procesamiento) son los que resultan de tipo no numérico, ya que los modelos propuestos trabajan únicamente datos de este tipo.

Posteriormente se procedió a filtrar los datos que, sin necesidad de previo análisis, por las características del conjunto de datos, se sabe que son redundantes. Para temperatura y humedad se hacen tres mediciones: "mínimo", "máximo" (en el periodo de diez minutos que corresponde a un registro) y "externo". Se decidió trabajar con la medición correspondiente a "externo". Habiendo filtrado estos datos y los correspondientes a valores no numéricos, quedan 19 atributos.

Con estos 19 atributos restantes se analizó la correlación, utilizando el método *Dataframe.corr()* para obtener la matriz de correlación, y a partir de ésta se utilizó el método *heatmap()* de la biblioteca Seaborn para realizar la matriz de calor, que se muestra en Ilustración 28.

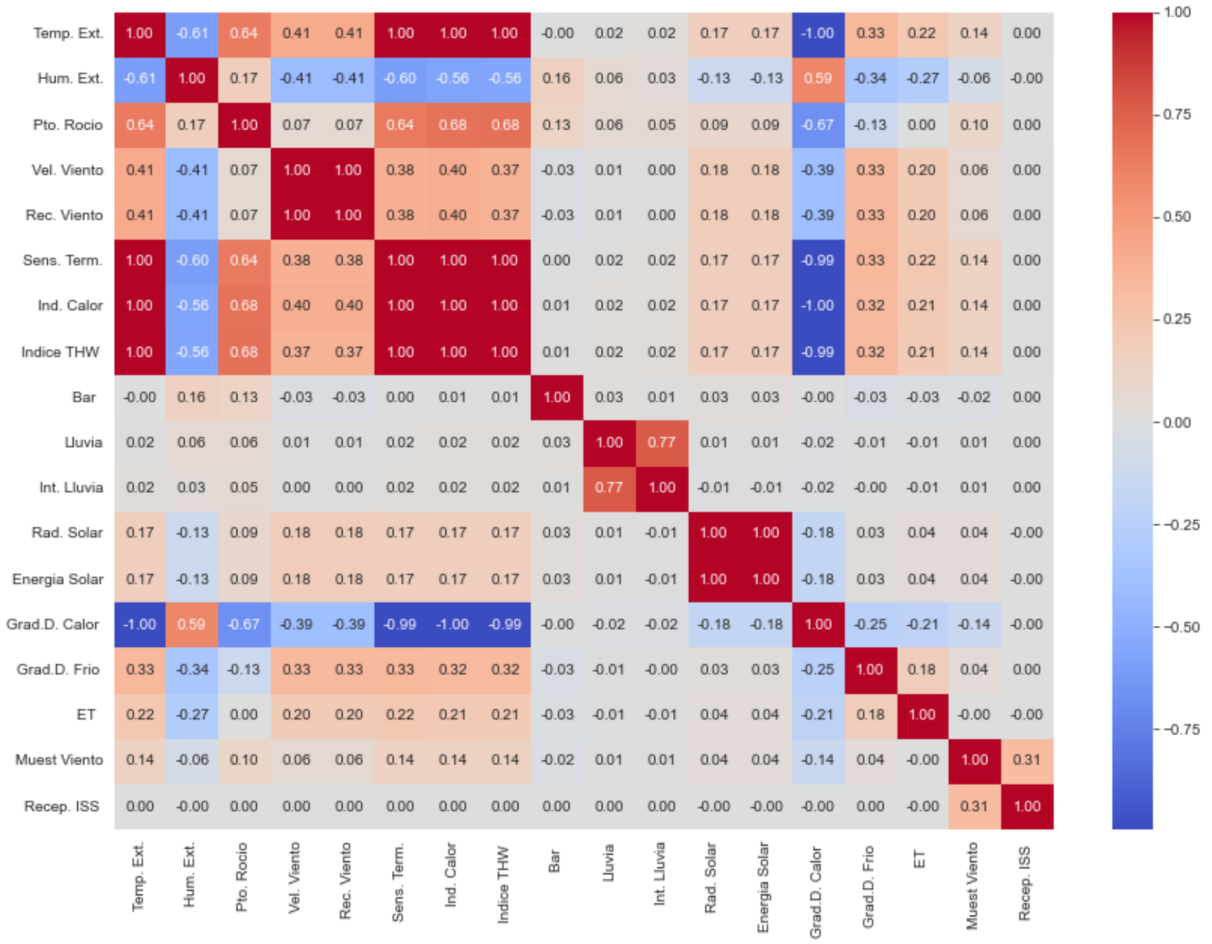

Ilustración 28 matriz de calor para las variables de entrada

Esta matriz de calor permite ver rápida y claramente cuáles atributos resultan redundantes, ya que tienen un índice de correlación cercano a 1(correlación directa) o -1 (correlación indirecta) y a su vez, son marcados en rojo o azul oscuro respectivamente. Se consideraron como redundantes aquellos atributos que tengan una correlación mayor o igual a 0.8, o menor a -0.8, con otro atributo.

Luego de que se eliminaron los atributos redundantes, resulta un total de 13 atributos, los cuales serán utilizados como entrada de los modelos que se mostrarán a continuación. En la Ilustración 29 Matriz de calor para las variables que serán entrada del modelo se puede observar la matriz de calor nuevamente, pero luego de haber filtrado los atributos.

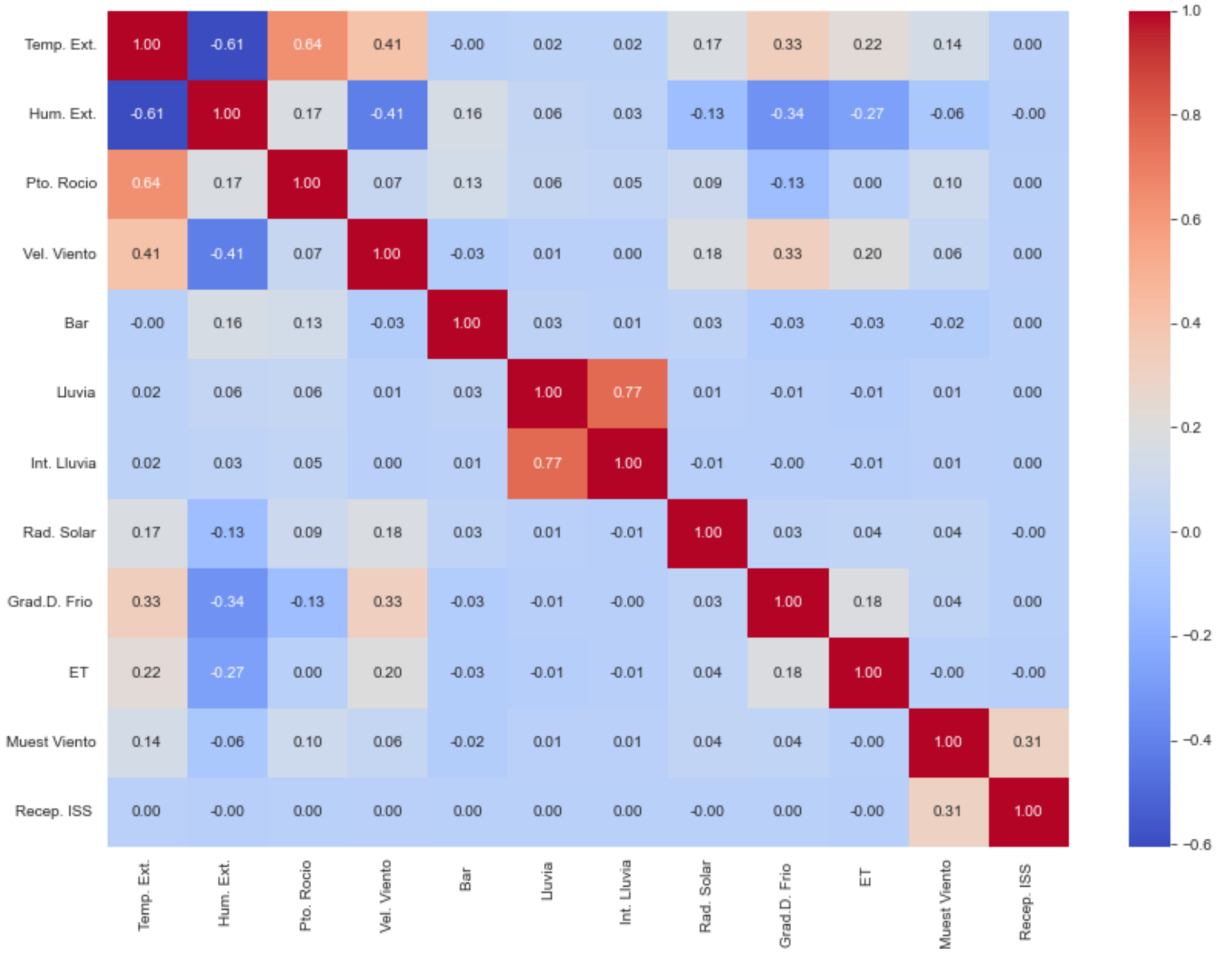

Ilustración 29 Matriz de calor para las variables que serán entrada del modelo

Una vez eliminados estos atributos, el conjunto de datos está listo para ser utilizado por parte de los modelos de aprendizaje de máquina.

5.2. Predicción de heladas

5.2.1. Predicción de heladas por ventanas de tiempo

5.2.1.1. Preprocesamiento de datos

Mediante esta estrategia se utilizaron ventanas temporales para poder predecir si ocurrirá una helada en K horas, particularmente, se tomaron los valores K = 2,3,4,5,6.

Para esta estrategia se agregaron al *dataframe* cinco nuevas columnas, denominadas *lag_temp_K* de tipo flotante, cada una con la temperatura en exactamente K horas, así como también otras cinco columnas de tipo booleana, llamadas *lag_K,* indicando si ocurrió o no heladas en dichas horas.

Como los datos están disponibles en registros tomados cada diez minutos, para obtener la temperatura en K se desplaza la lectura del registro para dicho atributo en $6K$ columnas, de forma tal que para obtener la temperatura en las próximas dos horas se hará un desplazamiento de 12 registros, para tres horas de 18 registros, para 4 horas de 24, para 5 horas de 30 registros y para 6 horas de 36 registros. Luego si la columna *lag_temp_K* es positiva o igual a cero, *lag_K* recibirá el valor falso, caso contrario, recibirá el valor verdadero.

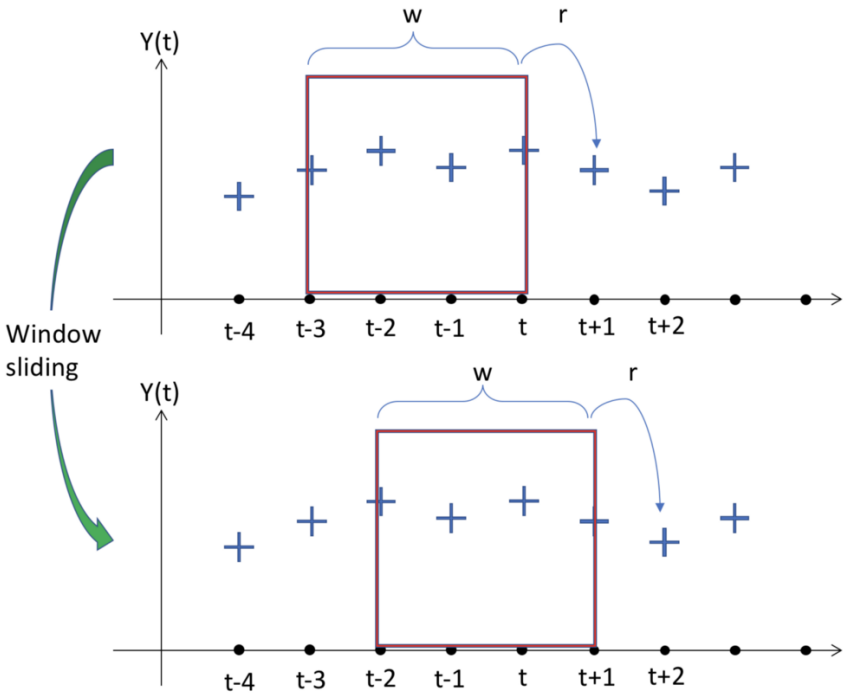

Ilustración 30Representación genérica de una ventana de tiempo (30)

En la Ilustración 30, w el tamaño de la ventana, en nuestro caso K. Por otro lado, *t* indica la la unidad de tiempo, en nuestro caso, una hora. Y *r* es la distancia de tiempo entre el fin de una ventana y el comienzo de la siguiente, en este trabajo eso serían diez minutos, o 0.167 horas.
Cabe aclarar que este proceso se realizó antes de recortar el dataset para prescindir de los horarios y épocas irrelevantes para el estudio de ocurrencia de la helada, y luego de haber ordenado el conjunto de datos por fecha y habiendo realizado una interpolación de los datos faltantes en las series temporales, porque es claro que el proceso mencionado depende del orden de los registros y de no haberse realizado este procesamiento, se estarían asumiendo datos erróneos.
Para esta estrategia se utilizaron modelos de predicción por clasificación, esto es, en base a las distintas variables meteorológicas de entrada de tipo numérica, se estableció si ocurrió o no helada, etiquetando un registro con *VERDADERO* o *FALSO respectivamente,* de acuerdo a la predicción del modelo.

5.2.2. Modelos

Luego del procesamiento adicional del conjunto de datos, necesario para adecuar los mismos a la presente estrategia, se creó un diccionario de modelos, el cual tiene como clave *lag_K*, indicando que contendrá el conjunto de datos de prueba y entrenamiento para la predicción de la helada en K horas, como así también contendrá una instancia del modelo y los resultados de la predicción obtenida por el mismo. Las instancias de los algoritmos que se utilizaron fueron Regresión Logística y Random Forest Clasificador de la biblioteca *ScikitLearn*.
Para evaluar los resultados de los modelos se analizaron las matrices de confusión, y se consideraron diferentes métricas como pueden ser la precisión, exactitud, F1 Score, y por, sobre todo, el *recall*, que se buscaba maximizar ya que es prioritario reducir la cantidad de falsos negativos.

5.2.2.1. Resultados del modelo con el algoritmo de Regresión Logística

5.2.2.1.1. Hiperparámetros

Los hiperparámetros utilizados fueron:

- Penalidad: regularización L2. Este parámetro agrega la "magnitud al cuadrado" del coeficiente como término de penalización a la función de pérdida.

- Dualidad: Falso. Formulación dual o primaria. Se prefiere dualidad=falso cuando la cantidad de muestras es mayor a la cantidad de características.
- Tolerancia: 0.0001. Este parámetro indica detener el algoritmo de buscar un máximo o un mínimo cuando la diferencia es inferior a determinado número. Esto es, tolerancia de error.
- C = 1.0. Un valor alto de C indica al modelo que otorgue un peso alto a los datos de entrenamiento y un peso menor a la penalización por complejidad. Esto es, un valor de C alto significa que los datos de entrenamiento son confiables, mientras que un valor bajo indica que los datos de entrenamiento pueden no ser del todo representativos.
- Ponderación de clases: ninguna. En este caso, indica que todas las clases tienen el mismo peso.
- Algoritmo de optimización: L-BFGS. De las siglas: Limited-memory Broyden–Fletcher–Goldfarb–Shanno. Aproxima las actualizaciones de la matriz de la segunda derivada con evaluaciones de gradiente. Almacena solo las últimas actualizaciones, por lo que ahorra memoria.

Se eligieron estos parámetros dado que son los parámetros por defecto en la biblioteca Scikit Learn. Durante el trabajo se cambiaron y modificaron estos parámetros, sin mayor impacto en los resultados, y cuando hubo diferencias significativas deterioraron la performance del modelo. Por esto tampoco se profundizó en mostrar los diferentes resultados para distintas configuraciones de hiperparámetros.

5.2.2.1.2. Matrices de confusión

A continuación, en la Tabla 2, así como en las tablas posteriores a esta y que presenten Matrices de Confusión, se indican en las columnas la cantidad de registros predichos para dicha clase, mientras que en las filas se muestran la cantidad de valores reales para cada clase.

Tabla 2 Matriz de confusión, para cada valor de K, para los resultados de Regresión Logística

K		Falso	Verdadero
2	Falso	41329	2361
	Verdadero	1983	7401
3	Falso	41320	2810
	Verdadero	2879	6065

4	Falso	42198	2414
	Verdadero	3903	4559
5	Falso	43180	2059
	Verdadero	4198	3637
6	Falso	44345	1633
	Verdadero	4357	2739

5.2.2.1.3. Métricas

Tabla 3 Métricas, para cada valor de K, para los resultados de Regresión Logística

K	Valor	Precisión	Recall	F1 Score	Exactitud
2	Falso	0.95	0.95	0.95	0.91
	Verdadero	0.78	0.71	0.74	
3	Falso	0.90	0.94	0.92	0.87
	Verdadero	0.64	0.50	0.56	
4	Falso	0.88	0.95	0.91	0.85
	Verdadero	0.54	0.31	0.39	
5	Falso	0.87	0.96	0.91	0.85
	Verdadero	0.46	0.19	0.27	
6	Falso	0.88	0.97	0.92	0.86
	Verdadero	0.41	0.11	0.18	

5.2.2.2. Resultados del modelo con el algoritmo Random Forest Clasificador

5.2.2.2.1. Hiperparámetros

Los hiperparámetros utilizados fueron:

- Número de árboles: 1.000.
- Criterio: Gini.
- Cantidad mínima de hojas: 2.

5.2.2.2.2. Matrices de confusión

Tabla 4 Matriz de confusión, para cada valor de K, para los resultados de Random Forest Clasificador

K		Casos predichos	
	Casos reales	Falso	Verdadero
2	Falso	41329	2361
	Verdadero	1983	7401
3	Falso	41320	2810
	Verdadero	2879	6065
4	Falso	42198	2414
	Verdadero	3903	4559
5	Falso	43180	2059
	Verdadero	4198	3637
6	Falso	44345	1633
	Verdadero	4357	2739

5.2.2.2.3. Métricas

Tabla 5 Métricas, para cada valor de K, para los resultados de Random Forest Clasificador

K	Valor	Precisión	Recall	F1 Score	Exactitud
2	Falso	0.95	0.95	0.95	0.92
	Verdadero	0.76	0.79	0.77	

3	Falso	0.93	0.94	0.94	0.89
	Verdadero	0.68	0.68	0.68	
4	Falso	0.92	0.95	0.93	0.88
	Verdadero	0.65	0.54	0.59	
5	Falso	0.91	0.95	0.93	0.88
	Verdadero	0.64	0.46	0.54	
6	Falso	0.91	0.96	0.94	0.89
	Verdadero	0.63	0.39	0.48	

5.2.2.3. Conclusiones para esta estrategia

Como se puede observar en Tabla 3 y Tabla 5, la performance de ambos modelos respecto a exactitud es similar, siendo ligeramente mejor Random Forest. Esta performance se mantiene para distintos valores de K, es decir un incremento en la cantidad de horas hacia futuro que se intenta predecir la helada, que se ve claramente en las gráficas de Ilustración 31 y Ilustración 32.

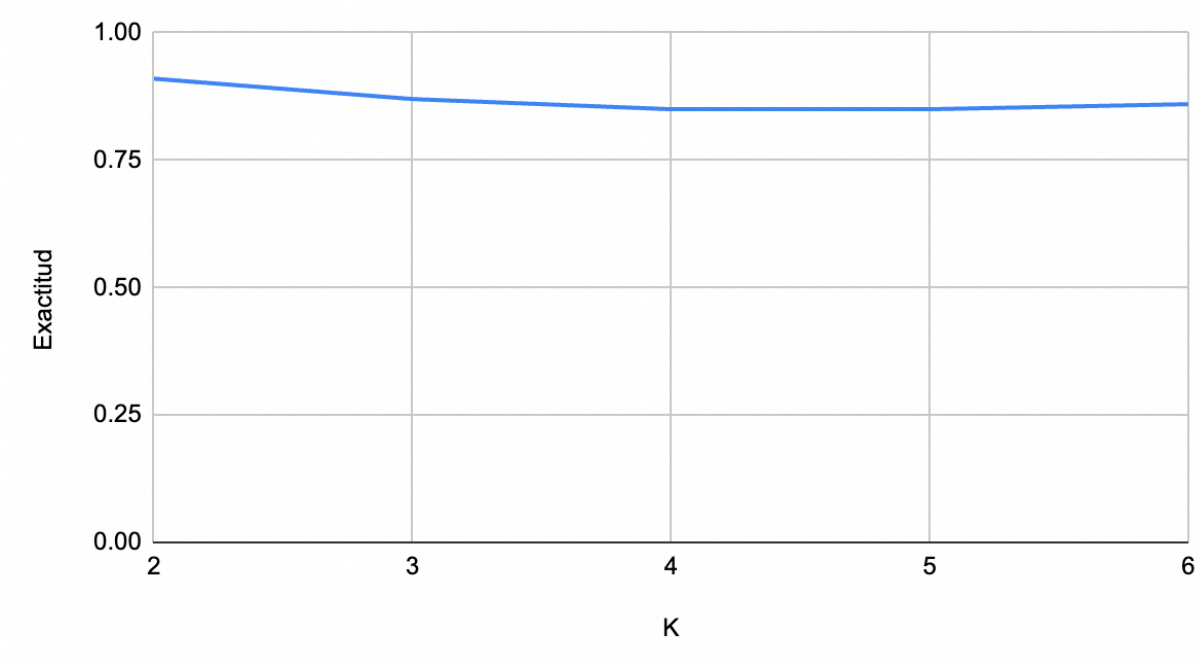

Ilustración 31 Exactitud de regresión logística, por valor de K.

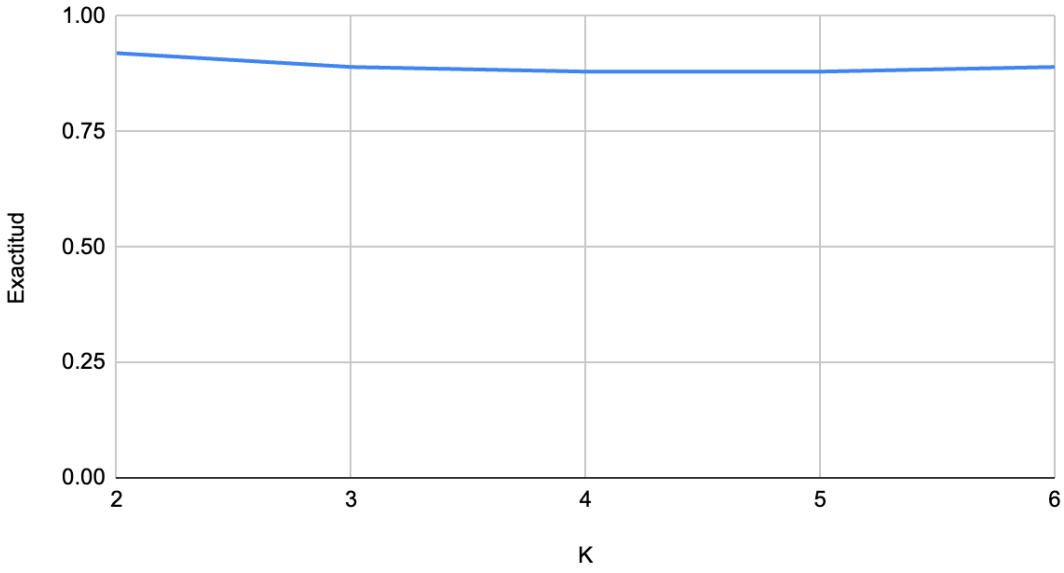

Ilustración 32 Exactitud de Random Forest Clasificador, por valor de K.

Sin embargo, considerando la métrica de *recall*, que es la que realmente interesa a fin de reducir la cantidad de falsos negativos para así poder alertar correctamente de la ocurrencia de una helada, se observa que el modelo de Random Forest hace un mucho mejor trabajo identificando las heladas y prediciendo su ocurrencia que el modelo de Regresión Logística. En ambos modelos también se observa que el *recall* decrementa a medida que se intenta predecir heladas en un futuro más lejano, esto es, que se incrementa el valor de K. En las gráficas de Ilustración 33 e Ilustración 34 se puede ver claramente este comportamiento, a su vez, también se observa

que Regresión Logística es mucho más sensible a los incrementos en K, de forma que su rendimiento empeora notablemente.

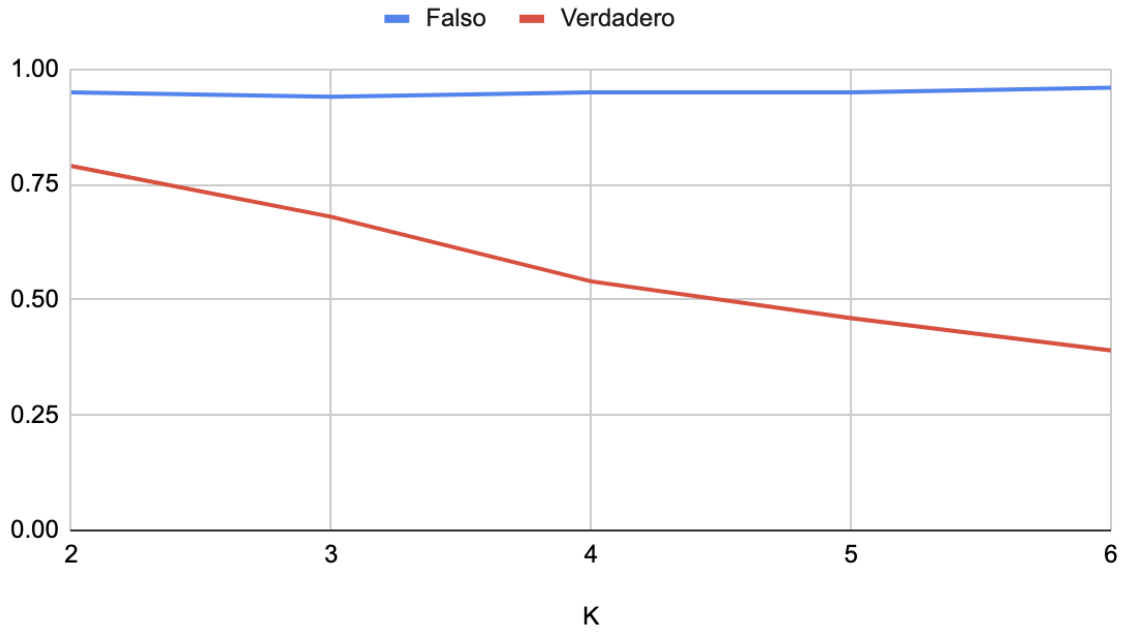

Ilustración 33 Variación del recall para Random Forest Clasificador, según valor de K.

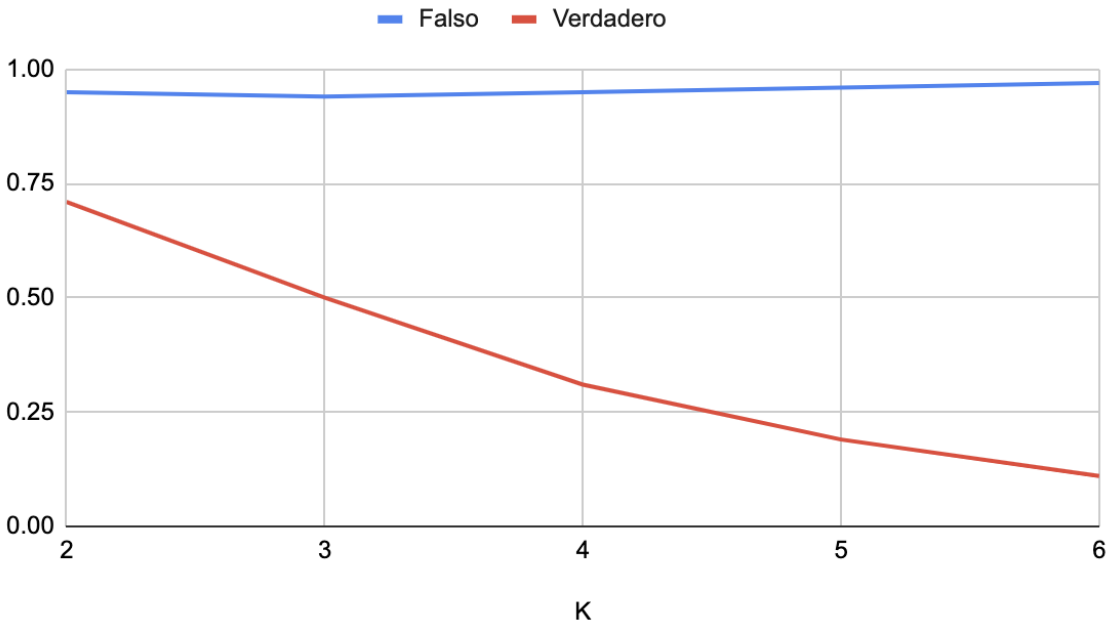

Ilustración 34 Variación del recall para Regresión Logística, según valor de K.

5.2.3. Predicción de heladas por promedios meteorológicos diarios

En la estrategia anterior, para predecir la ocurrencia de heladas se utilizaron ventanas temporales de K horas, lo que significa que el modelo toma como entrada los 3K registros

anteriores - K horas multiplicadas por un registro cada 10 minutos- y en base a eso predecir si ocurrirá o no helada en un horizonte de K horas hacia delante, tiempo en cual el productor puede tomar las medidas de mitigación correspondientes.

Para esta estrategia, donde además de la ocurrencia de la helada también se quiere determinar su duración y su intensidad, se utilizó un enfoque distinto en el abordaje previo del procesamiento de los datos.

Se agruparon los datos en "días lógicos". Un día lógico inicia a las 10 AM de un día real y termina a las 9:59 AM del día real siguiente. Luego, se separó este conjunto de datos agrupados, en dos conjuntos distintos.

Un primer conjunto con los datos en horarios en los que no ocurren heladas, que son de 10 a 19:59, el cual posee el promedio para las distintas variables meteorológicas durante esas horas. Estos son los datos que el modelo utilizará como entrada.

El segundo conjunto, con los datos en horarios en los que ocurren heladas, es decir, entre las 20 y las 9:59, que contendrá la temperatura mínima en ese rango de tiempo, la intensidad de la helada y la duración de la helada. Estos valores son los que el modelo intentará predecir.

Se utilizaron las temperaturas medias para los horarios de día con el objetivo de lograr un sistema que, basado en la media de las variables meteorológicas de un determinado día, llegada las 19:00, sea capaz de predecir si ocurrirá una helada, que tan intensa será y qué duración tendrá. Por otro lado, la razón por la que se tomó la temperatura mínima para los horarios de helada, es dado que el valor mínimo para dicha variable en ese periodo de tiempo determinará si ocurrirá helada y qué intensidad tendrá.

En cuanto a la intensidad de la helada, la misma está dada por la temperatura mínima que alcanza la helada. La duración, por otro lado, se obtuvo contando la cantidad de registros entre la primera y la última temperatura bajo cero en ese rango de tiempo y atento a que los datos se registran cada 10 minutos, se multiplicó por diez para obtener la duración.

Si bien existen casos en los que la temperatura cae por debajo de los cero grados y luego sube nuevamente, se consideró oportuno despreciar estos casos, debido a que en su mayoría lo que sucedía era que la temperatura subía unas pocas décimas por arriba de cero durante una escasa cantidad de registros y luego bajaba nuevamente. Desde el punto de vista del agricultor, carece de sentido práctico detener por completo todos los mecanismos de mitigación ante estos casos.

La Ilustración 27 presenta la cantidad de registros con heladas por hora a lo largo de los años relevados. Así a la 0hs durante esos años, algo más de 500 fueron los registros con heladas, en

tanto a las 7hs, superaron los 1.750. Se aprecia también que no se contabilizaron registros con heladas entre las 11hs y las 18hs.

Temp. Ext.	Hum. Ext.	Pto. Rocio	Vel. Viento	Bar	Lluvia	Int. Lluvia	Rad. Solar	Grad.D. Frio	ET	Muest Viento
12.456667	41.600000	-0.683333	1.866667	955.735000	0.000000	0.000	389.150000	0.000000	0.037000	233.600000
13.113333	45.400000	1.328333	2.000000	952.075000	0.000000	0.000	320.766667	0.000000	0.031333	233.516667
13.008333	66.566667	6.816667	3.440000	950.468333	0.008333	0.025	245.483333	0.000000	0.023833	233.616667
15.765000	58.300000	7.275000	1.146667	954.570000	0.000000	0.000	378.116667	0.000617	0.037500	233.750000
16.728333	51.400000	6.175000	1.760000	948.255000	0.000000	0.000	373.116667	0.005283	0.038667	233.783333
...	...	...	...	...	...	...	...	...	...	...
14.713333	74.583333	10.095000	0.986667	1014.120000	0.000000	0.000	200.583333	0.000000	0.018500	233.566667
24.348333	48.250000	11.958333	1.333333	1006.120000	0.000000	0.000	546.800000	0.045100	0.060500	233.816667
23.355000	40.150000	8.566667	3.280000	1005.301667	0.000000	0.000	557.750000	0.034933	0.063000	233.800000
24.960000	26.733333	2.863333	5.025000	1006.510000	0.000000	0.000	580.816667	0.046133	0.073167	233.750000

Ilustración 35 Tabla con los promedios para las distintas variables meteorológicas, donde cada fila representa un día

fecha_y	temp_min	intensidad	duracion
2013-05-16	-3.8	2.02	440
2013-05-17	7.4	0.00	0
2013-05-18	4.3	0.00	0
2013-05-19	-0.8	0.43	110
2013-05-20	1.6	0.00	0
2013-05-21	-0.4	0.21	50
2013-05-22	0.3	0.00	0
2013-05-23	0.3	0.00	0
2013-05-24	-0.5	0.27	40
2013-05-25	-0.3	0.16	90
2013-05-26	1.5	0.00	0
2013-05-27	7.3	0.00	0
2013-05-28	0.8	0.00	0
2013-05-29	0.7	0.00	0

Ilustración 36 Tabla con los valores para temperatura mínima, intensidad y duración de la helada, donde cada fila representa un día

5.2.3.1. Etiquetas y clasificación de las características de la helada

5.2.3.1.1. Temperaturas: ocurrencia de heladas.

Para la clasificación respecto a la ocurrencia de helada, es claro que existen dos clases posibles: ocurre helada o no ocurre helada. Para esto se utilizaron dos clases discretas (Verdadero, Falso) como se muestra en la Tabla 6 para la salida.

Se observó que, a partir de esta clasificación, ambas clases se encuentran balanceadas, donde los datos correspondientes a registros de heladas representan un 43% del total.

Tabla 6 Clasificación de ocurrencia de helada a partir de la temperatura

Clase	Valores de temperatura	Cantidad
Verdadero	< 0	383
Falso	≥ 0	480

5.2.3.1.2. Duración de las heladas.

Similarmente a lo que ocurre con las temperaturas, para las estrategias mediante clasificadores es necesaria la existencia de clases. Considerando las necesidades del productor agropecuario que le interesará saber que tan extensa será la helada, surge una clasificación en tres tipos: breve, media y extensa, y a su vez para aquellos registros que no corresponden a helada, se determina clase nula, que significa duración 0. El criterio que se utilizó para establecer la cantidad de etiquetas asociadas al clasificador de la duración, fue que cada uno de estos grupos mencionados contienen aproximadamente la misma cantidad de registros, con la finalidad de que el conjunto de datos se encuentre balanceado y así también poder establecer valores numéricos límites que impliquen qué duración corresponde a qué clase, especificado en la Tabla 7. Para lograr esto se analizó la distribución de las duraciones de las heladas, la cual se evidencia en la Ilustración 37.

Tabla 7 Clasificación de la duración de heladas, medidas en minutos

Clase	Desde	Hasta
BREVE	10’	209’
MEDIA	210’	449’
EXTENSA	450’	840’(máx.)

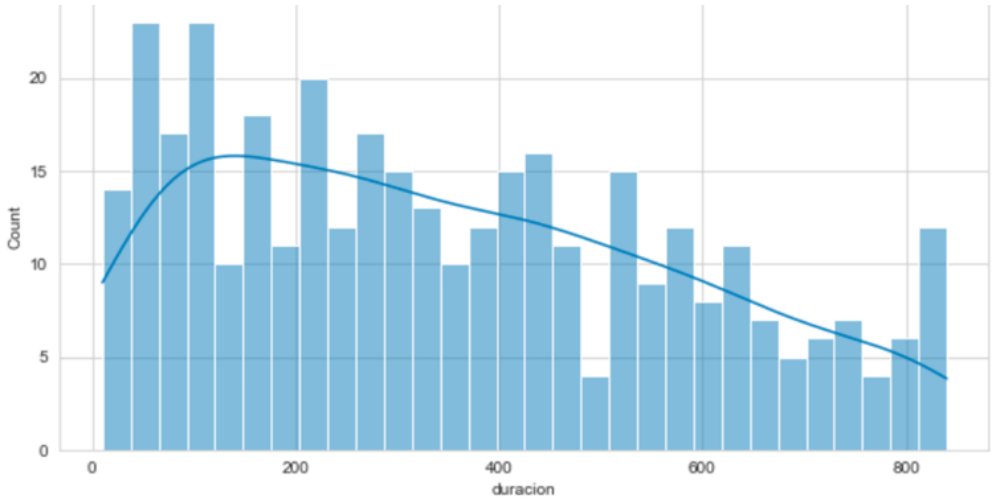

Ilustración 37 Gráfica de la distribución de la duración de las heladas

5.2.3.1.3. Intensidad de las heladas.

Continuando lo mencionado para la clasificación de la duración de las heladas, surge la clasificación para la intensidad de la helada, considerando esta vez las temperaturas mínimas alcanzadas. Atendiendo tanto a la necesidad de los modelos clasificadores de un conjunto finito de etiquetas dentro las cuales proyectar sus resultados, como a la necesidad del productor agropecuario de establecer distintos niveles de intensidad para la helada, se proponen las siguientes clases: baja, media y alta, evidenciadas en la Tabla 8, y como sucedía para la duración, aquellos registros en los que no hubiera helada, también se utilizará la clase nula, en este caso indicando que la temperatura fue mayor a cero.

Tabla 8 Clasificación de intensidad de heladas, a partir de la temperatura mínima

Clase	Desde	Hasta
BAJA	-0.1	-1.4
MEDIA	-1.5	-3.4
ALTA	-3.5	-9.4 (mín.)

Como sucedía anteriormente, para esto se analizó la distribución de las temperaturas para aquellos registros en los que se presentara helada, y a partir de esto se determinaron las cotas que enmarcan en qué clase de intensidad corresponde una helada, esta distribución se muestra en la Ilustración 38.

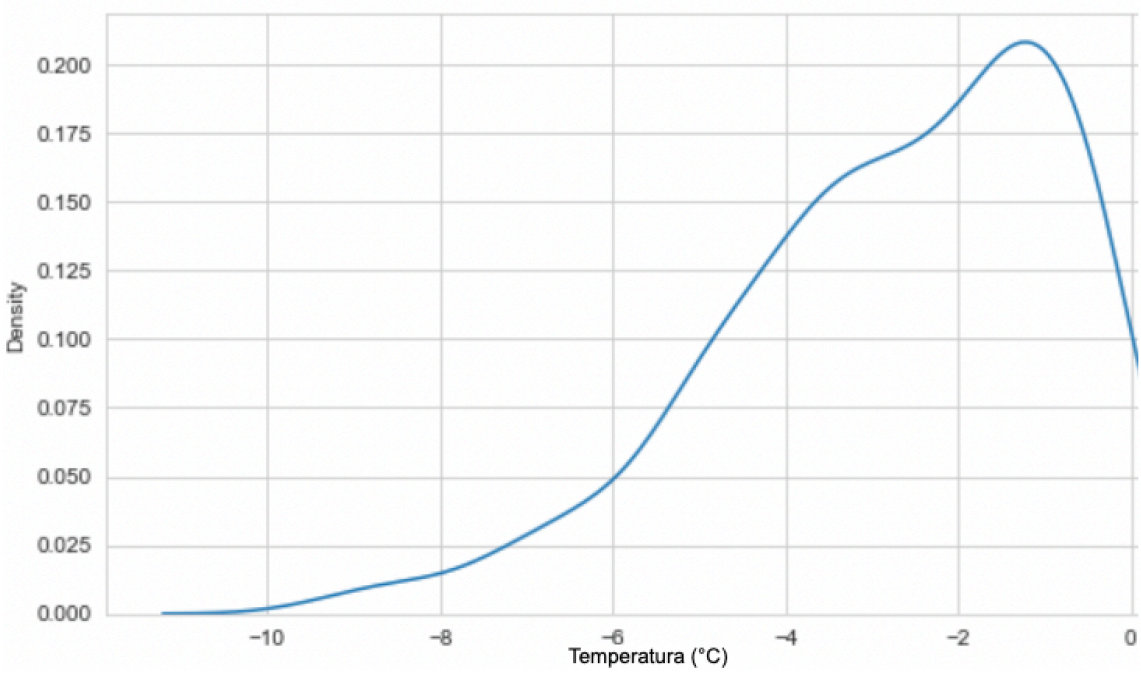

Ilustración 38 Distribución de la intensidad de heladas

5.2.3.2. Modelos

Para la predicción se utilizaron tanto modelos regresores como clasificadores. Dado que los modelos clasificadores requieren de la existencia de clases o etiquetas no continuas o discretas, en las cuales se enmarcan los resultados, nace la necesidad de crear distintas clases considerando los objetivos de predecir ocurrencia, intensidad y duración de la helada. Por otro lado, los modelos regresores arrojarán un resultado numérico acerca de cuánto durará la helada, que temperatura mínima alcanzará, determinando así su ocurrencia o no y su intensidad.

Tal como se mencionó anteriormente, se abordaron las predicciones como problemas de regresión y de clasificación. En ambos casos se utilizó como algoritmo de modelación Random Forest, dada su buena performance mostrada en estrategias y trabajos anteriores y también porque es un modelo relativamente simple si es comparado con, por ejemplo, redes neuronales recurrentes de tipo LSTM. Dada la baja cantidad de datos luego de haber realizado el preprocesamiento, se prefiere utilizar un algoritmo de las características de Random Forest para evitar un sobreajuste.

Los hiper parámetros utilizados fueron:

- Número de árboles: 1.000.
- Criterio: Gini para los problemas de clasificación y error cuadrático para los problemas de regresión.
- Cantidad mínima de hojas: 2.

5.2.3.3. Predicción de temperaturas y ocurrencia de heladas

El modelo arrojó un R^2 de 0.60. Este valor oscila entre 0 y 1, y mientras más cercano a uno mejor será el ajuste del modelo.

Por otro lado, se obtuvo un error cuadrático medio de 9.63. Este valor es básicamente el promedio de los errores al cuadrado, por lo tanto, mientras más cercano a cero mejor serán las predicciones del modelo.

Clasificando los resultados en “helada” y “no helada”, según si la temperatura predicha es menor o mayor a cero, respectivamente.

Como se puede apreciar en la Tabla 9, observando particularmente el valor obtenido para *recall*, que permite ver cuál es el comportamiento de los modelos respecto a los falsos negativos, que es de nuestro máximo interés reducir, es claro que el modelo se comporta adecuadamente.

Tabla 9 Métricas para el modelo de regresión a través de Random Forest, clasificando las temperaturas obtenidas por regresión.

Clase	Precisión	Exhaustividad (recall)	F1 Score	Exactitud
No helada	0.84	0.80	0.82	0.80
Helada	0.76	0.79	0.77	

Tabla 10 Métricas para el modelo de clasificación a través de Random Fores Clasificador.

Clase	Precisión	Exhaustividad (recall)	F1 Score	Exactitud
No helada	0.82	0.83	0.83	0.81
Helada	0.80	0.79	0.79	

Se puede observar de las Tabla 9 y

Tabla 10 que los resultados son similares, usando regresión y clasificación.

5.2.3.4. Predicción de intensidad de heladas

Haciendo una clasificación en base a la intensidad de las temperaturas obtenidas por el algoritmo de Random Forest Regresor, se obtuvieron las siguientes métricas observadas en Tabla 11

Tabla 11 Métricas para la predicción de temperaturas por regresor Random Forest, clasificadas en base a la intensidad.

Clase	Precisión	Exhaustividad (recall)	F1 Score	Exactitud
NULA	0.84	0.80	0.82	0.60
BAJA	0.34	0.28	0.31	
MEDIA	0.36	0.21	0.27	
ALTA	0.27	0.83	0.41	

Se puede ver que, la determinación en clases tiene margen de mejora atento a que la exactitud del modelo no es suficientemente buena. Sin embargo, se aprecia que el *recall* es elevado para las heladas de alta intensidad, lo cual es positivo a la hora de identificar heladas graves con anticipación.

Por otro lado, las métricas obtenidas por Random Forest clasificador se encuentran en la Tabla 12.

Tabla 12 Métricas para la predicción de temperaturas por el clasificador Random Forest.

Clase	Precisión	Exhaustividad (recall)	F1 Score	Exactitud
NULA	0.87	0.78	0.82	0.66
BAJA	0.08	0.18	0.11	
MEDIA	0.27	0.26	0.26	
ALTA	0.71	0.70	0.71	

En este caso, se observa que la exactitud del modelo sube, y existe una notable variabilidad en las distintas métricas para cada clase respecto al modelo regresor.

5.2.3.5. Predicción de duración de heladas

Utilizando un algoritmo de Random Forest regresor, este arrojó un R^2 de 0.16 y un error medio absoluto de 116 minutos. Los resultados de la clasificación de los valores obtenidos se encuentran en la Tabla 13.

Tabla 13 Métricas obtenidas para la predicción de la duración por regresor Random Forest.

Clase	Precisión	Exhaustividad (recall)	F1 Score	Exactitud
NULA	0.07	1.00	0.12	0.27
BREVE	0.75	0.16	0.26	
MEDIA	0.44	0.27	0.33	
EXTENSA	0.46	0.65	0.54	

Por otro lado, utilizando un Random Forest clasificador se obtuvieron las métricas presentes en la Tabla 14.

Tabla 14 Métricas obtenidas para la predicción de la duración por clasificador Random Forest.

Clase	Precisión	Exhaustividad (recall)	F1 Score	Exactitud
NULA	0.86	0.77	0.81	0.63
BREVE	0.08	0.23	0.12	
MEDIA	0.33	0.30	0.31	
EXTENSA	0.86	0.77	0.81	

Como se puede observar de Tabla 13 y Tabla 14, abordando la predicción de la duración como un problema de clasificación, la exactitud del modelo mejora notablemente. Y en ambos

modelos se mantuvo la tendencia de que mientras más extensa es la helada, mejor es la predicción.

Tanto a nivel intensidad como de duración se destaca el buen rendimiento del modelo. Esto se debe a que la dinámica de los sistemas térmicos ante una helada intensa y su posterior retorno a temperatura sobre 0 grados requiere de un mayor tiempo de duración.

5.2.3.6. Conclusiones para esta estrategia

De las distintas tablas de métricas obtenidas para el análisis de esta estrategia, se observa que para la predicción de la ocurrencia de helada se comporta de una forma superior a la estrategia anterior con ventanas de tiempo, con un recall de 0.83 para el modelo de Random Forest Clasificador, comparado con un recall de 0.79 obtenido para el mismo modelo bajo la estrategia de ventanas de tiempo con una ventana de dos horas Considerando que con la estrategia actual es posible estimar si se producirá una helada a las 20:00 horas para un día determinado, la estrategia presentada anteriormente sólo permite estimar si se producirá helada con sólo dos horas de anticipación.

Respecto a la predicción de la duración e intensidad de la helada, se puede observar que bajo esta estrategia tanto los modelos clasificadores como regresores de Random Forest arrojan valores deficientes para recall para la predicción de estas características, con la excepción de la categoría más severa de cada una, que son las heladas extensas y las de alta intensidad, con un recall de 0.77 y 0.83 respectivamente, lo mismo ocurre con las categorías nulas. De esto se interpreta que los modelos presentan dificultades discriminando heladas de intensidad baja e intensidad media, y con las heladas de duración media y breve.

5.2.4. Predicción de heladas con combinación de estrategias

En estrategias previas se utilizó un esquema de ventanas de tiempo, para poder predecir la hora en la que sucederá la helada. Por otra parte, el modelo de la estrategia por promedios meteorológicas diarios permite indicar si ocurrirá una helada y qué características tendrá, pero no determina la hora a la que ocurrirá, lo cual es algo necesario para el agroproductor a fin de establecer los mecanismos de mitigación en el horario correspondiente a la helada, ya que considerando únicamente el modelo que predice que puede ocurrir una helada en un rango de 14hs, estos mecanismos deberían estar activos durante todo este rango de tiempo, lo cual no es eficiente. Es por esto que se propone complementar ambos modelos presentados anteriormente. De esta forma se tienen las ventajas del modelo actual, donde la exhaustividad es más alta, esto

es, posee una menor cantidad de falsos negativos, a la vez que se puede obtener el horario específico en el que ocurrirá la helada. La Ilustración 39 Representación de la aplicación conjunta de ambos modelos muestra la aplicación conjunta de ambos modelos.

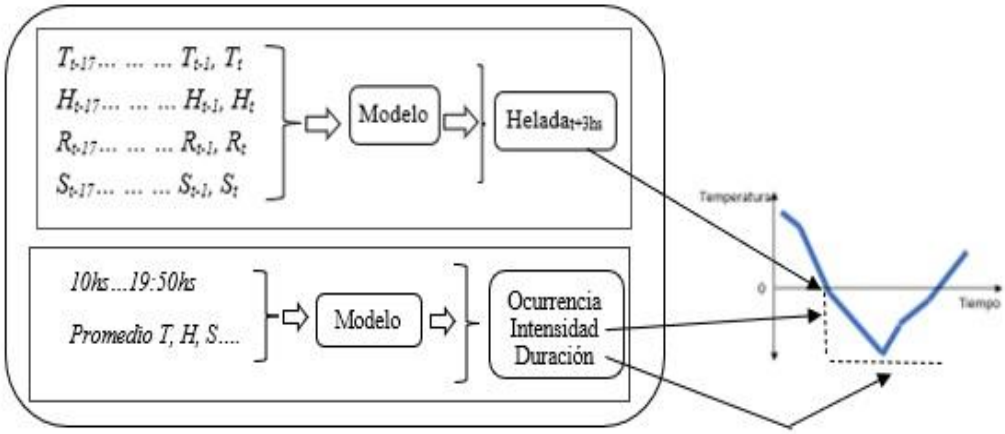

Ilustración 39 Representación de la aplicación conjunta de ambos modelos

5.2.5. Predicción de ocurrencia de heladas a través de redes neuronales.

En este trabajo también se exploró la posibilidad de predecir las heladas mediante el uso de redes neuronales, particularmente a través de la red de tipo Perceptrón Multicapa, que es de los modelos más simples dentro de este ámbito, así como también mediante el modelo de redes neuronales recurrentes LSTM (*Long Short Term Memory*) que son más complejos que el anterior mencionado, además de requerir mayor poder computacional. Para desarrollar esta sección se utilizó la herramienta de software libre Knime, especializada en Ciencia de Datos, que dispone de un módulo de integraciones para añadir las bibliotecas Tensorflow y Keras, necesarias para crear las redes neuronales que se desarrollaron y que además permiten la configuración para habilitar el procesamiento paralelo ejecutándose sobre GPU. En la sección posterior, donde se aborda el procesamiento de datos bajo esta herramienta, se mostrarán los resultados obtenidos para la predicción de ocurrencias de heladas bajo estos modelos, luego del procesamiento necesario para adecuar el conjunto de datos con los que se trabajó en Python.

5.3. Procesamiento de datos en Knime

En esta sección se presenta lo trabajado con la herramienta Knime, en la cual se utilizaron los datos preprocesados anteriormente a través de Python y las diferentes bibliotecas exploradas, a los cuales se les aplicó un procesamiento adicional para que puedan ser utilizados de entrada para los distintos modelos planteados.

A grandes rasgos, se puede apreciar en la Ilustración 40, que el conjunto de nodos utilizados en Knime para el procesamiento de datos y así dar el formato correcto a los mismos para ser utilizados como entrada para el modelo Random Forest, en el cual no se profundizó, ya que ha sido desarrollado en secciones anteriores de este trabajo. De igual manera se adecuaron los datos para ser procesados por un perceptrón multicapa y dos redes neuronales con una capa oculta, siendo una de ellas una red de tipo LSTM.

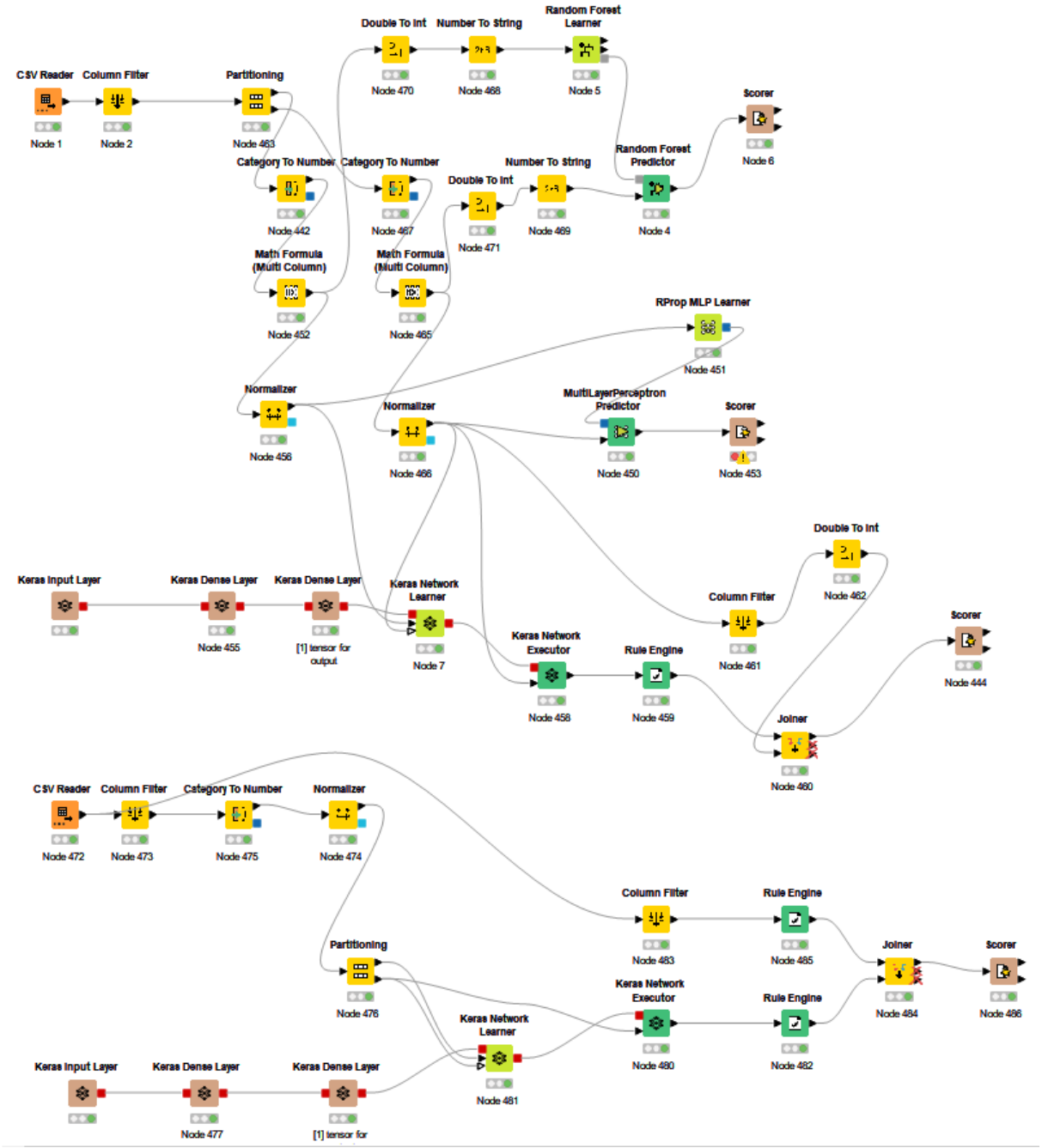

Ilustración 40 Workflow en Knime

5.3.1. Workflow #1: Perceptrón multicapa y red neuronal de una capa para predicción a 3hs

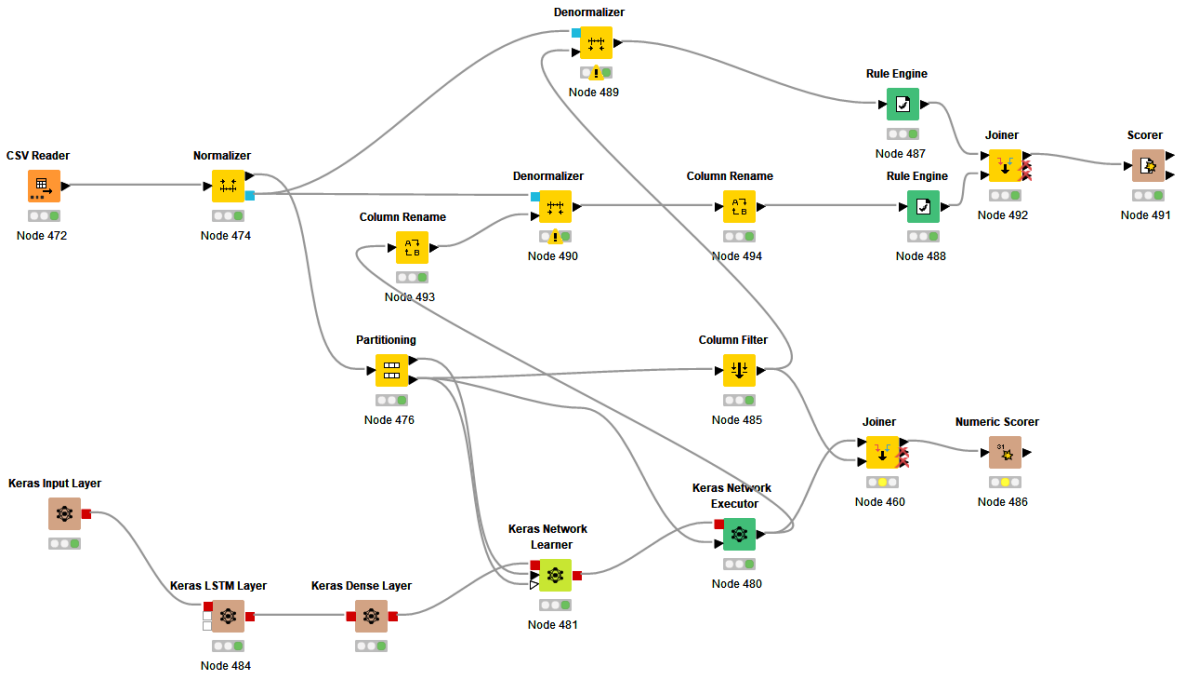

Ilustración 41 Flujo de Knime para perceptrón y red neuronal superficial

5.3.1.1. Preprocesamiento de datos

Como entrada para el *workflow* de Knime se utilizaron los datos obtenidos del preprocesamiento realizado anteriormente en este trabajo, posterior a recortar los periodos de tiempo que no son relevantes para la helada, así como también los horarios en las que estas no ocurren.

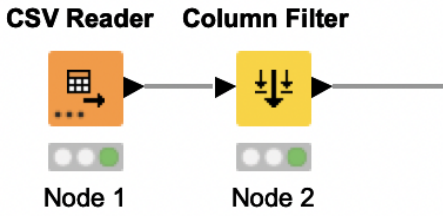

Ilustración 42 Nodos para leer datos y filtrar columnas, respectivamente.

Este trabajo fue realizado por el nodo de tipo *CSV Reader*, encargado de leer los archivos CSV, posteriormente el nodo *Column Filter* filtró las columnas que no eran necesarias para este análisis, como lo pueden ser la fecha y las columnas *lag_K* descritas en secciones anteriores.

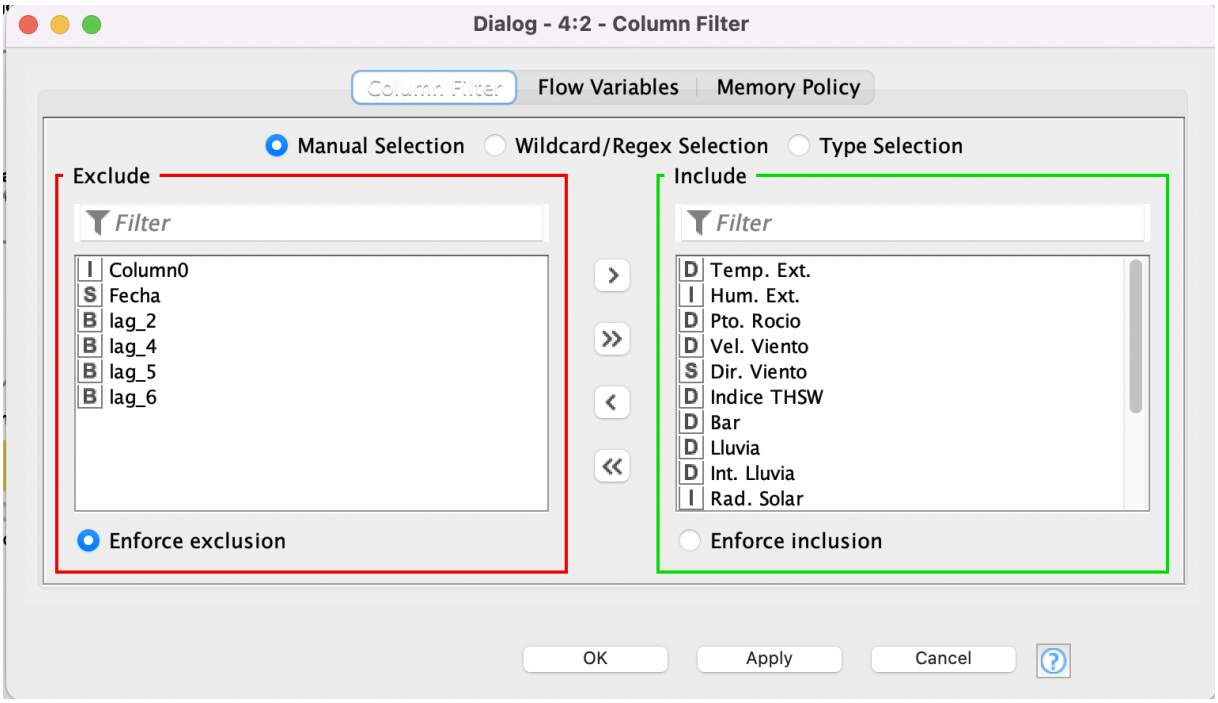

Ilustración 43 Configuración del nodo Column Filter

Una vez realizado esto, es necesario transformar todos los tipos de datos a flotantes, dado que así es como es requerida la entrada por parte de los nodos relacionados a las redes neuronales. Para ello, se utilizan dos tipos de nodos: *Category To Number* que permite transformar valores de tipo cadena a entero, lo que fue particularmente útil para transformar la columna *Dir. Viento*.

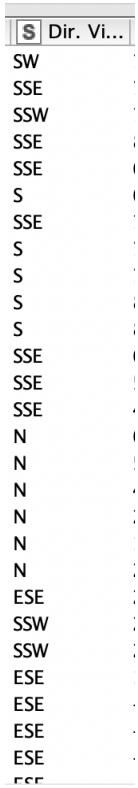

Ilustración 44 Valores de la columna dirección del viento, previo a transformación de datos

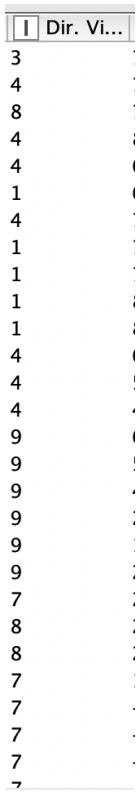

Ilustración 45 Valores de la columna dirección del viento, posterior a transformación de datos

En Ilustración 44 e Ilustración 45 se puede observar los valores para la columna en cuestión, antes y después del procesamiento del nodo *Category to Number*. El modo de funcionamiento

de este nodo es el siguiente: analiza una por una las diferentes cadenas de la o las columnas seleccionadas y les asigna un entero de forma incremental, con posibilidad de definir cuál es el primer valor y de cuanto es el incremento, como se muestra en la pantalla de Ilustración 46.

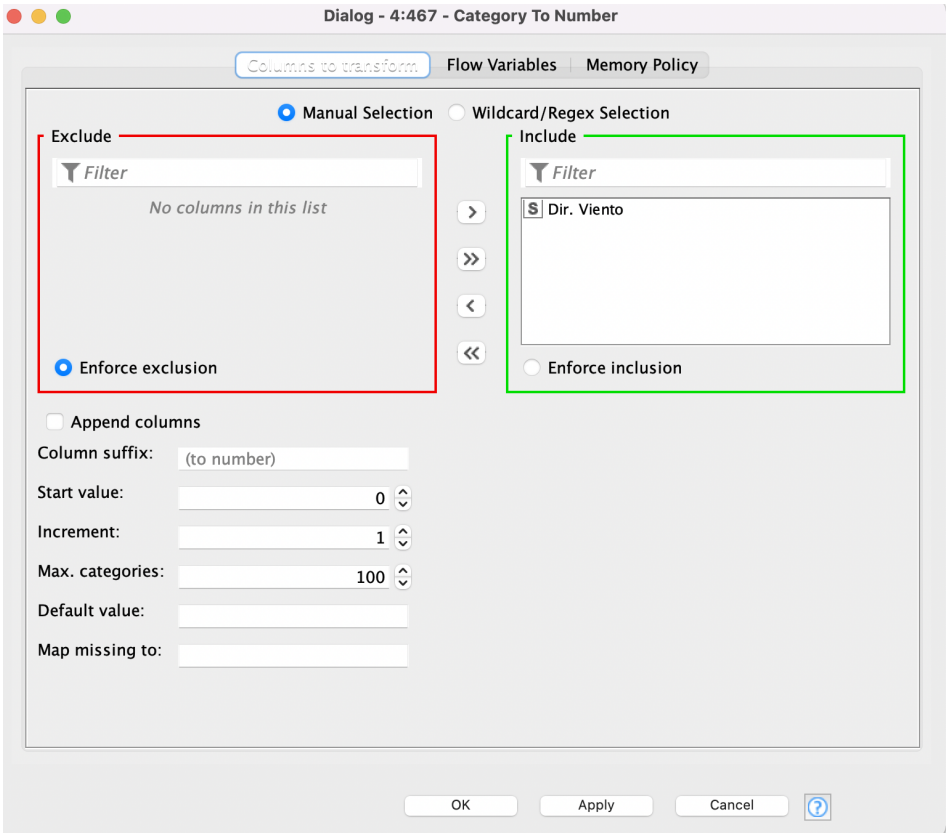

Ilustración 46 Pantalla de configuración del nodo Category To Number

El siguiente paso en lo que refiere a preprocesamiento es convertir los tipos de datos enteros en reales, esto se realiza mediante el nodo *Math Formula*, que, como el nombre lo indica, permite transformar los valores de los atributos aplicando fórmulas matemáticas. Sin embargo, en este caso no se requiere utilizar ninguna fórmula, tan solo convertir los valores enteros en reales, de forma que la fórmula resultará en este caso $f(x) = x$, pero la salida obtenida será el número real. En la Ilustración 47 se muestra la pantalla de configuración para este nodo.

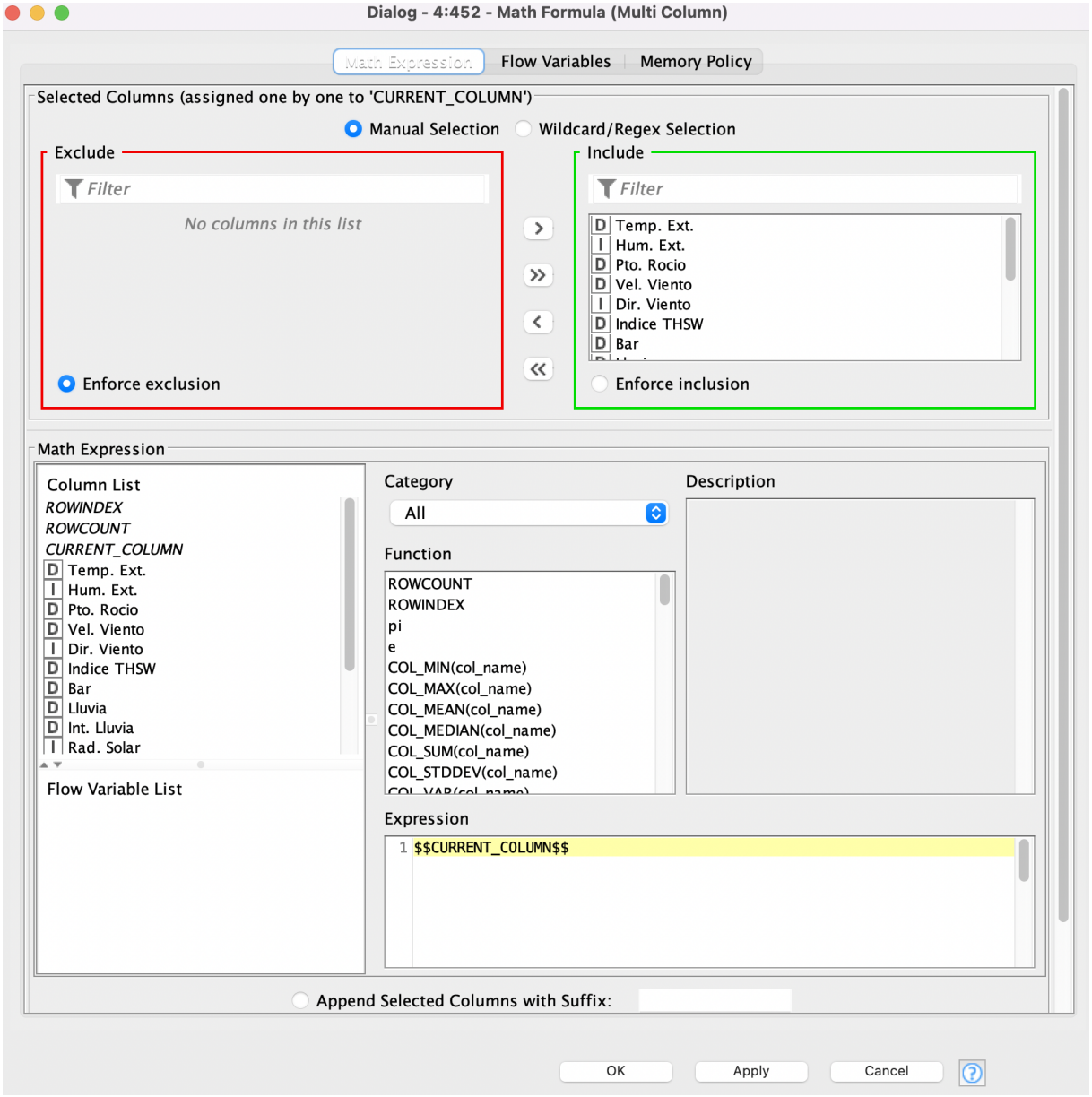

Ilustración 47 Pantalla de configuración del nodo Math Formula

Lo único que resta ahora en cuanto al preprocesamiento de los datos, es la normalización de los mismos para ser utilizados por los nodos de aprendizaje de las redes neuronales. Esto se logró mediante el nodo *Normalizer*, normalizando todos los valores para ubicarlos entre 0 y 1, aunque este nodo también da otras opciones de normalización.

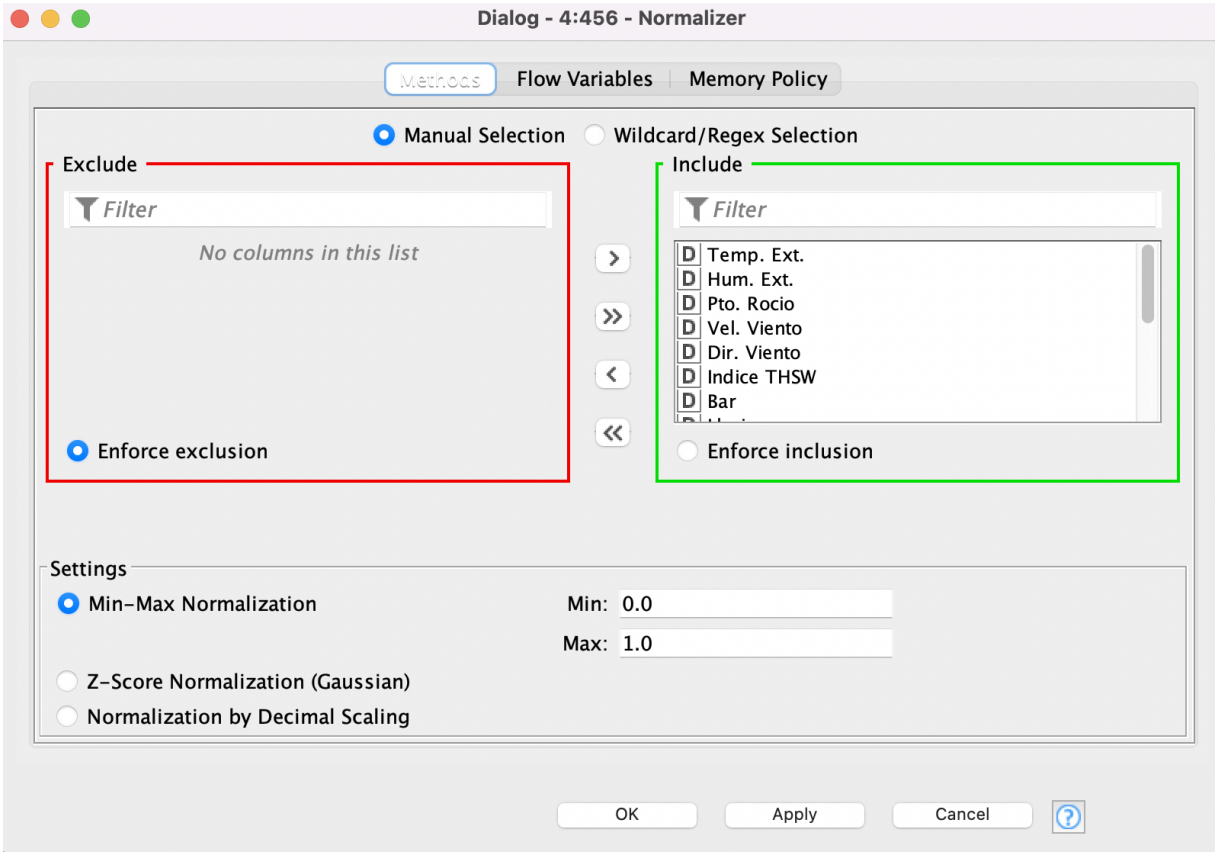

Ilustración 48 Pantalla de configuración del nodo Normalizer

Luego de haber realizado el preprocesamiento de estos datos, es necesario separar estos en datos de entrenamiento y de prueba para lo cual se usa el nodo *Partitioning*, el cual provee distintas formas alternativas para dividir en dos el conjunto de datos, indicando de qué tamaño debería ser cada uno de ellos. Para este trabajo, en esta instancia, se utilizó el muestreo aleatorio y se tomó un 70% de los datos para entrenamiento y un 30% para pruebas. En la Ilustración 49 se puede observar la pantalla de configuración del nodo en cuestión.

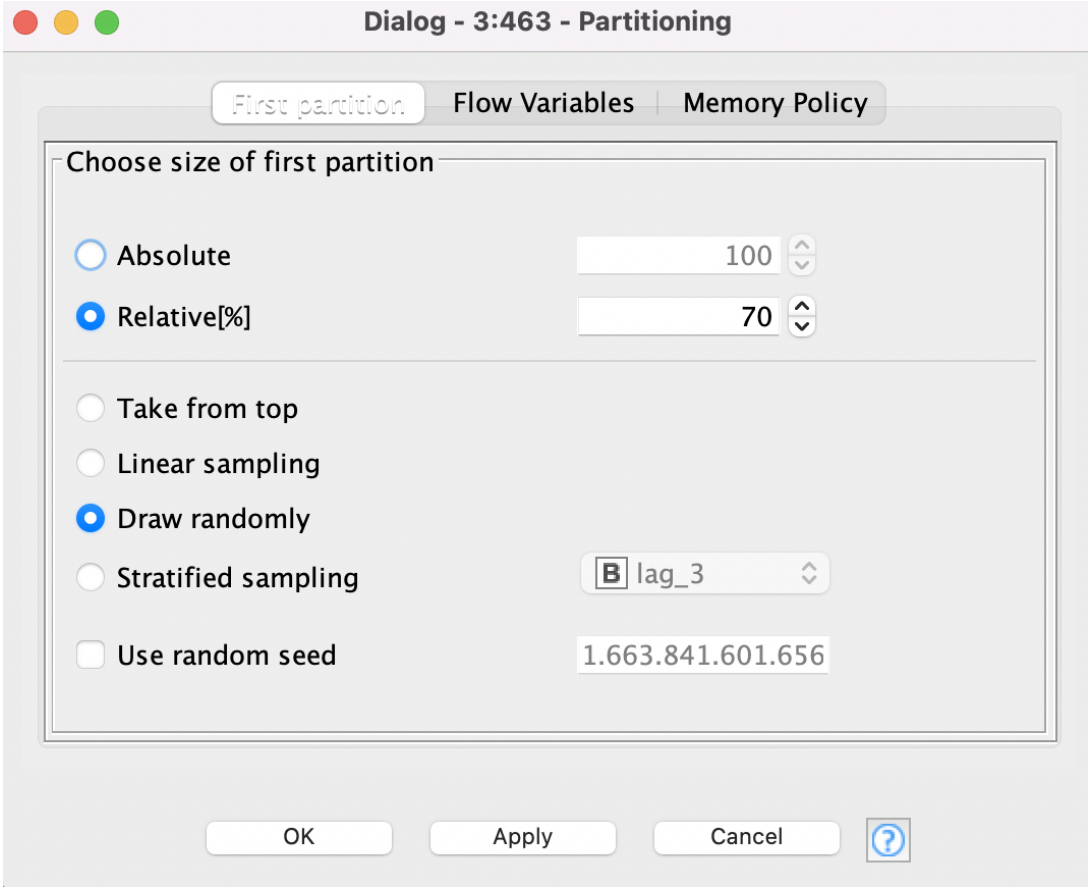

Ilustración 49 Pantalla de configuración del nodo Partitioning

5.3.1.2. Perceptrón Multicapa

Para entrenar al perceptrón multicapa, se utiliza el nodo *RProp MLP Learning*, que recibe como entrada el conjunto de datos de entrenamiento y que permite configurar los parámetros que son: cantidad máxima de iteraciones, cantidad de capas, cantidad de neuronas por capa y que variable se deberá predecir. Como se mencionó anteriormente, se predecirá la variable *lag_3* que indica la temperatura con una ventana de 3 horas.

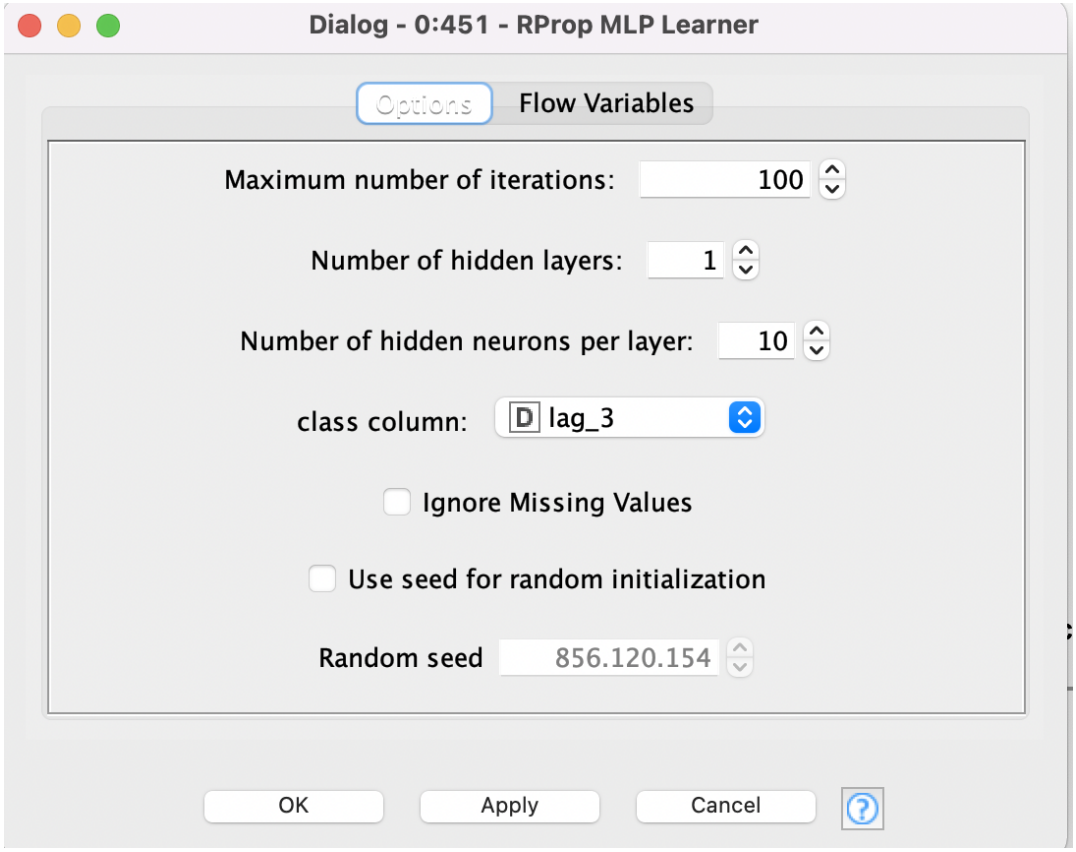

Ilustración 50 Pantalla de configuración del nodo RProp MLP Learning

Luego de haber entrenado al perceptrón, su salida se utilizará como entrada del nodo *MultilayerPerceptron Predictor*, junto con los datos de prueba para poder hacer las predicciones. Posteriormente se utiliza el nodo *Rule Engine*, necesario porque las predicciones realizadas por el perceptrón multicapa son números reales, y como en este caso se trabajó con números enteros - 0 y 1- que representaban la no ocurrencia y la ocurrencia de helada, respectivamente, es necesario asignar valores discretos P a estos resultados X, tal que si X < 0.5 P=0 y X>= 0.5 P= 1.

Luego, fue necesario medir la performance del modelo, utilizando el nodo *Scorer,* este nodo recibe como entrada una tabla con dos columnas de valores discretos y hace una comparación entre los valores de ambas columnas para una misma fila para determinar si son iguales y en base a esto calcular la matriz de confusión y generar las métricas que se muestran en la Ilustración 51.

Row ID	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
1	1586	1016	10012	655	0.708	0.61	0.708	0.908	0.655	?	?
0	10012	655	1586	1016	0.908	0.939	0.908	0.708	0.923	?	?
Overall	?	?	?	?	?	?	?	?	?	0.874	0.578

Ilustración 51 Métricas obtenidas del nodo Scorer

Para utilizar este nodo, fue necesario utilizar previamente los nodos *Column Filter*, descrito anteriormente, y el nodo *Double to Int*, para convertir los valores reales en enteros. Finalmente, el flujo de nodos para el perceptrón multicapa resultó como se muestra en la Ilustración 52.

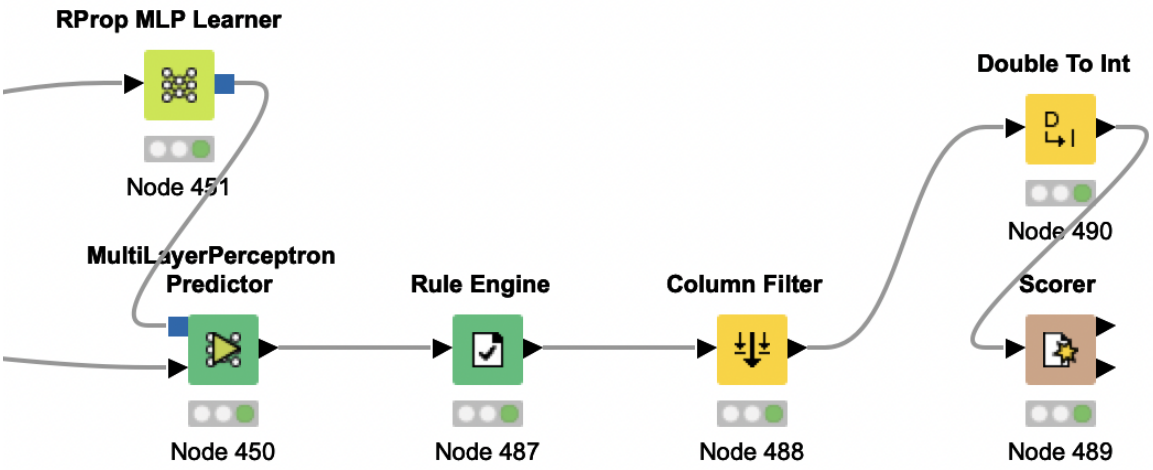

Ilustración 52 Flujo de Knime para perceptrón multicapa

5.3.1.2.1. Resultados

Tabla 15 Matriz de confusión de perceptrón multicapa

	Casos predichos	
Casos reales	Verdadero	Falso
Verdadero	1586	655
Falso	1016	10012

De un total de 2241 casos de heladas fueron predichas correctamente 1586, esto quiere decir, que, del total de heladas ocurridas, fue posible predecir con correctitud aproximadamente un 71% de las mismas. Evidenciando que el modelo posee una performance similar a la de modelos mostrados con anterioridad en el presente trabajo. Lo mismo puede observarse para las distintas métricas evaluadas en otros modelos.

Tabla 16 Métricas de performance por valor de K=3 para perceptrón multicapa

K	Valor	Precisión	Recall	F1 Score	Exactitud
3	Verdadero	0.61	0.71	0.65	0.87
	Falso	0.94	0.91	0.92	

5.3.1.3. Red neuronal FeedForward con una capa oculta

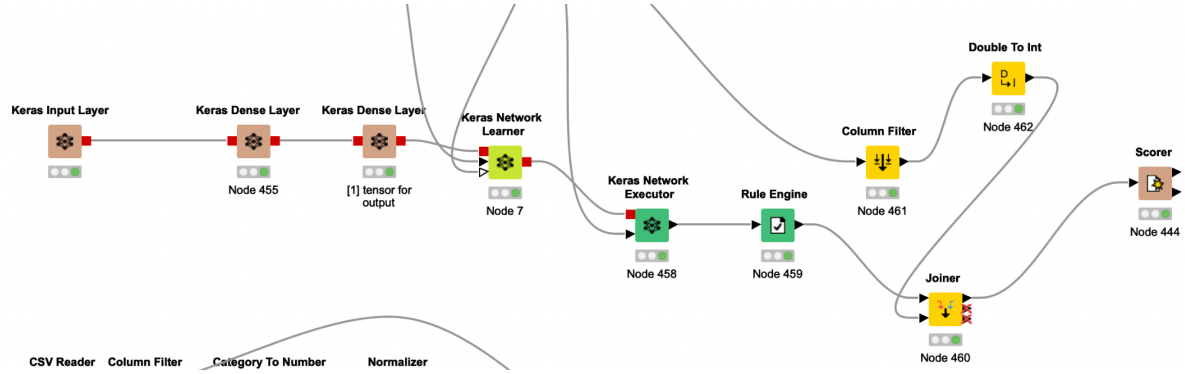

Ilustración 53 Flujo de Knime para red neuronal de una capa oculta

Para definir la red neuronal se utilizaron nodos de la integración de aprendizaje profundo, con Keras. Para este caso, se hizo uso de tres nodos en particular: *Keras Input Layer*, y dos *Keras Dense Layer*, tal como se muestra en la Ilustración 54.

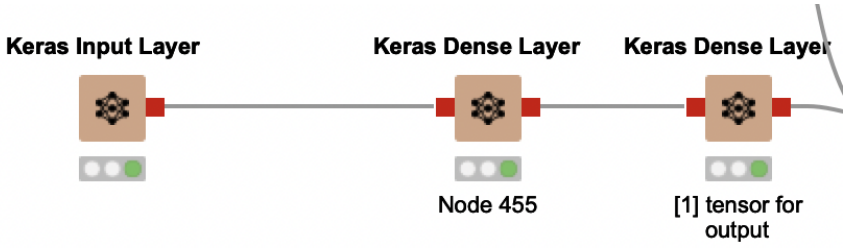

Ilustración 54 Red neuronal de una capa oculta en Knime

El primero es la capa de entrada. El nodo define cuál será la forma de la entrada, esto es, cuantos atributos o variables tendrá la entrada. Se definieron 15 entradas, una por cada variable meteorológica mencionada en la sección de Preprocesamiento.

El segundo nodo define una capa oculta. Permite configurar la función de activación y la cantidad de neuronas. La función elegida fue ReLU y la cantidad de neuronas, 15.

El tercer nodo define la capa de salida. Es un nodo del mismo tipo que el anterior, *Keras Dense Layer*, pero este tiene solo una neurona y la función de activación Sigmoid. Se eligió esta función dado que se quería representar, entre 0 y 1 la probabilidad de que ocurriera helada.

Se colocan tantos nodos como capas ocultas se vayan a utilizar o desde un solo nodo se puede especificar la cantidad de capas ocultas (en este caso todas iguales)

Una vez definida la red con los tres nodos mencionados, la red se entrenó mediante el nodo *Keras Network Learner*, que permite configurar cuál será la variable a predecir y que recibe como entrada los datos de entrenamiento. Posteriormente la red se ejecuta mediante el nodo *Keras Network Executor* que recibe como entrada los datos de prueba. Es importante aclarar que ambos nodos reciben los datos normalizados.

Luego, utilizando *Rule Engine* de la misma forma que se utilizó con Perceptrón Multicapa se utiliza el nodo *Scorer* se obtienen los resultados.

5.3.1.3.1. Resultados

Tabla 17 Matriz de confusión para red neuronal de una capa oculta

	Falso	Verdadero
Falso	9976	983
Verdadero	1052	1258

Tabla 18 Métricas para red neuronal de una capa oculta

K	Valor	Precisión	Recall	F1 Score	Exactitud
3	Verdadero	0.56	0.55	0.55	0.85
	Falso	0.95	0.91	0.91	

5.3.2. Workflow #2: Redes neuronales LSTM

Este *workflow* consistió en utilizar los datos obtenidos de la sección de Preprocesamiento de este trabajo, luego de haber recortado los periodos y horarios en los que no se producen heladas. Se utilizó un tipo de red neuronal que es capaz de recordar datos relevantes en la secuencia y preservarlos a través del tiempo, que son las redes LSTM.

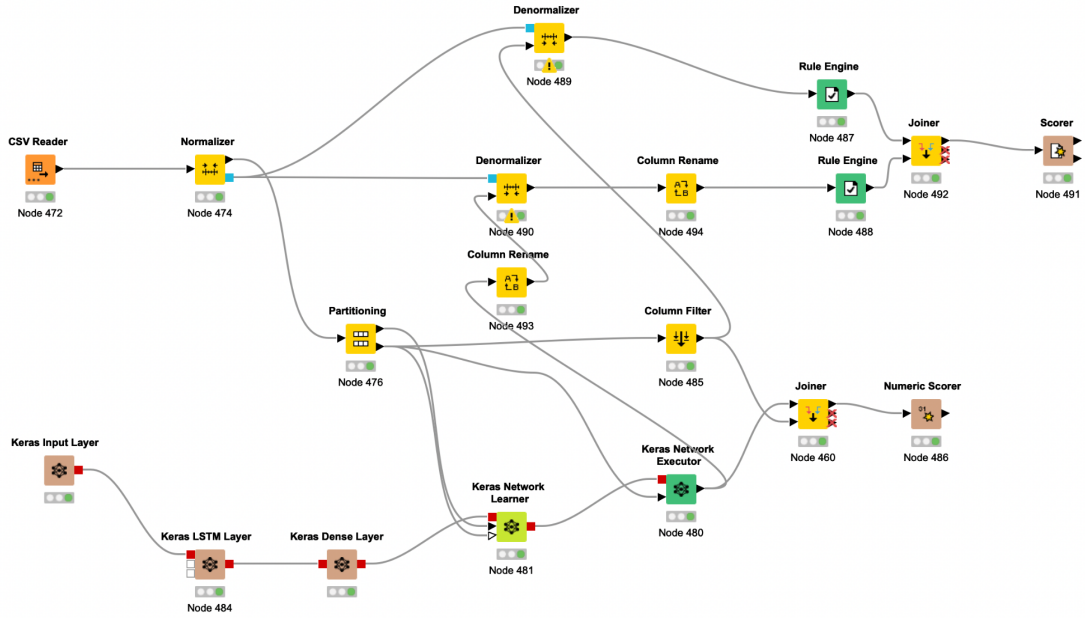

Ilustración 55 Flujo de Knime para una red neuronal de tipo LSTM

Como entrada del workflow, se leen los datos en formato CSV exportados del *Jupyter Notebook* donde se realizó el preprocesamiento a través del nodo *CSV Reader* y posteriormente los datos son normalizados. Luego se separaron en un conjunto de datos de entrenamiento, con un 70% de estos, y otro de pruebas, con el restante 30%.

La red neuronal se define de forma similar a la que de *workflow #1*, porque también es una red de una capa oculta, la diferencia es que esta está definida por un nodo de tipo *Keras LSTM Layer*, en vez de *Keras Dense Layer.* Similar a la anterior, tiene una entrada de 15 atributos. La capa oculta tiene la función de activación y de activación recurrente ReLU, y un *dropout* de 0. La capa de salida tiene la función de activación lineal. La Ilustración 56 muestra la pantalla de configuración de las opciones y parámetros del nodo *Keras LSTM Layer.*

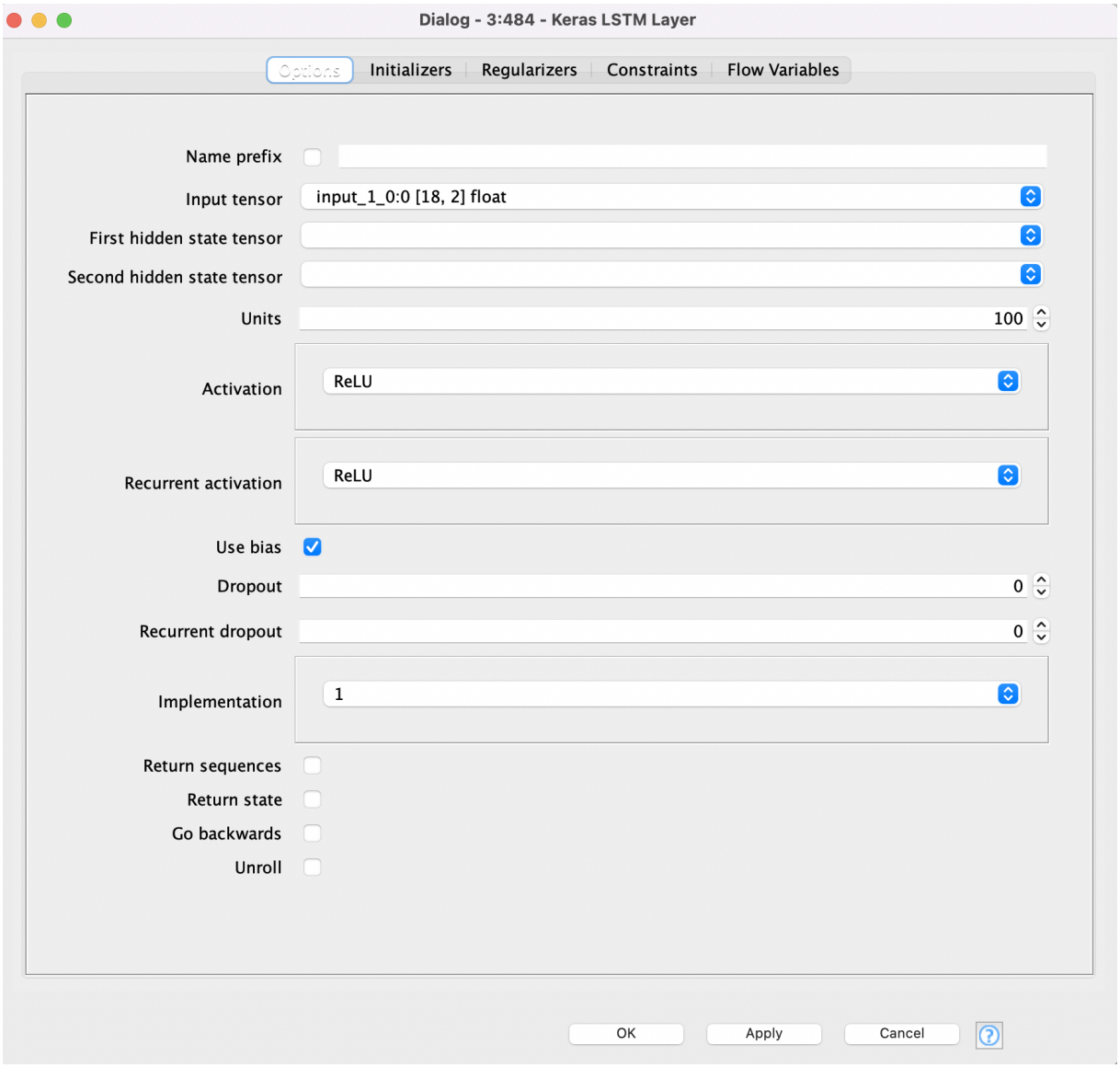

Ilustración 56 Pantalla de configuración para el nodo LSTM Keras Layer

5.3.2.1. Resultados

Tabla 19 Matriz de confusión obtenida para resultados de red LSTM

	Falso	Verdadero
Falso	8696	462
Verdadero	871	2174

Tabla 20 Métricas de rendimiento para resultados de red neuronal LSTM

K	Valor	Precisión	Recall	F1 Score	Exactitud
3	Verdadero	0.83	0.71	0.77	0.89
	Falso	0.91	0.95	0.93	

5.3.3. Tiempos de ejecución de redes neuronales: GPU vs. CPU

Las redes neuronales requieren de una alta capacidad de cómputo, en especial cuando se habla de redes neuronales de tipo LSTM, por lo que obtener los resultados mostrados anteriormente conlleva varios minutos de procesamiento y a medida que la red se complejiza y la cantidad de datos aumenta, estos tiempos de ejecución se incrementan al punto que entrenar una red puede demorar horas. Es por esto que las librerías de Python como Keras y Tensorflow tienen soporte para la ejecución del entrenamiento de estas redes sobre GPU, en vez de CPU.

5.3.3.1. Habilitación de la configuración en Knime

Knime, en su integración con el módulo de Deep Learning, basados en las mencionadas bibliotecas, posee una configuración para habilitar el procesamiento sobre GPU. Actualmente esta posibilidad solo está disponible en el sistema operativo Windows y con placas de video de tipo Nvidia. No funcionarán con procesamiento sobre GPU en otras GPU o en otros sistemas operativos y si bien la integración funcionará en Knime, sólo se podrá hacer el procesamiento sobre CPU.

Para habilitar esta configuración, se deben tener instalada la última versión de Python3 y poseer instaladas globalmente las siguientes bibliotecas:

- h5py (versión: 2.8)
- Numpy (versión: 1.15)
- Pyyaml (versión: 3.13)
- Scipy (versión: 1.1)
- Six (versión: 1.11)
- TensorFlow o tensorflow-gpu (versión: 1.12)

Posteriormente, será necesario instalar los siguientes paquetes de software:

- Nvidia GPU Drivers.
- CUDA Toolkit 10.0.
- cuDNN.

Después de las instalaciones, se debe ingresar a la sección de Python Deep Learning en la configuración de Knime y crear dos nuevos entornos virtuales de Python, uno para la biblioteca de Keras y otro para Tensorflow, como se muestra en la Ilustración 57.

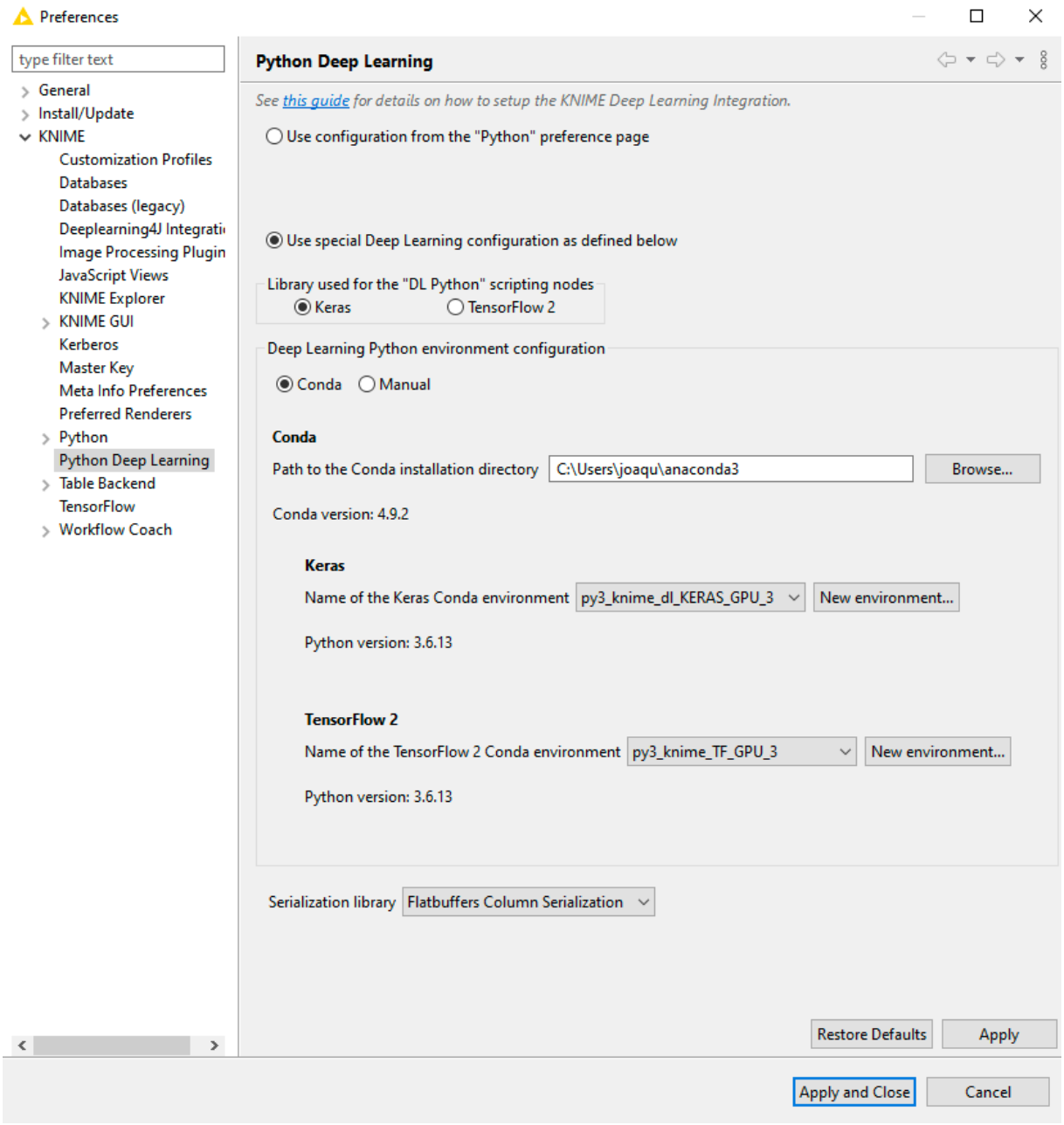

Ilustración 57 Pantalla de configuración del módulo de Deep Learning en Knime

Luego de esto, el paso final para habilitar esta configuración, fue ingresar al archivo *tensorflow_backend.py* presente en el directorio *\Lib\site-packages\keras\backend* de la carpeta de instalación de Python y modificar las líneas 179 a 188 del código, para que resulten como se indica en la Ilustración 58.

```
178     else:
179         if _SESSION is None:
180             if not os.environ.get('OMP_NUM_THREADS'):
181                 config = tf.ConfigProto(allow_soft_placement=True)
182                 config.gpu_options.allow_growth=True
183             else:
184                 num_thread = int(os.environ.get('OMP_NUM_THREADS'))
185                 config = tf.ConfigProto(intra_op_parallelism_threads=num_thread,
186                                         allow_soft_placement=True)
187                 config.gpu_options.allow_growth=True
188             _SESSION = tf.Session(config=config)
```

Ilustración 58 Líneas de código modificadas en Knime para permitir la ejecución sobre GPU

5.3.3.2. Resultados y tiempos

Para la red LSTM, mencionada en secciones anteriores, recordando sus características:

- Capa de entrada con 18 neuronas, una por cada uno de los atributos que se procesaron.
- Una capa oculta con 18 neuronas y función de activación ReLU.
- Se entrenó con un conjunto de datos de 48.800 registros.
- 50 Epochs.
- Tamaño de batch de entrenamiento de 800.
- Tamaño de batch de pruebas de 200.
- Optimizador: Adams.

Esta red, ejecutada sobre CPU demoró 6:08 (368s) minutos, mientras que sobre GPU exactamente la misma red demoró 1:53 minutos (113s). Esto implica una reducción en los tiempos de ejecución del 69,29%.

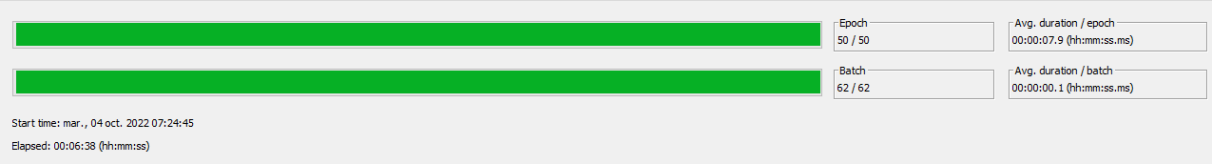

Ilustración 59 Tiempos de ejecución de red LSTM sobre CPU

Ilustración 60 Tiempos de ejecución de red LSTM sobre GPU

Por otro lado, tratada anteriormente, que contaba con las siguientes características:

- Capa de entrada con 15 neuronas.
- Una capa oculta con 15 neuronas y función de activación ReLU.
- Capa de salida con función de activación Sigmoid.

Y con los siguientes parámetros de entrenamiento:

- 53073 datos para entrenamiento.
- 150 Epochs.
- Tamaño de batch de entrenamiento: 1000.
- Tamaño de batch de validación: 1000.
- Optimizador: Stochastic gradient descent.

Esta red, ejecutada sobre CPU demoró 02:24 minutos (144s), mientras que sobre GPU exactamente la misma red demoró 02:15 minutos (135s). La diferencia del 6,25% es relativamente pequeña en comparación con la notable diferencia para la red anterior. Esto puede deberse a que en este caso se trata de una red mucho más simple, en cambio las redes LSTM demandan una gran cantidad de poder computacional en donde la GPU se torna necesaria.

Ilustración 61 Tiempos de ejecución de red de una capa oculta sobre CPU

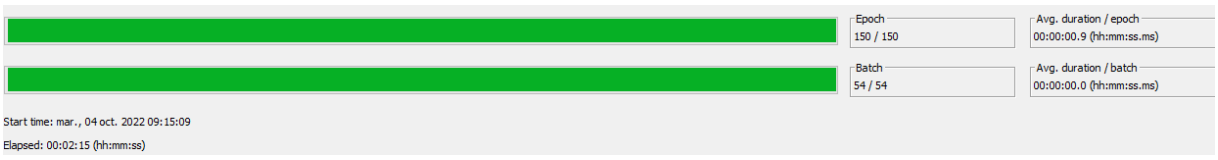

Ilustración 62 Tiempos de ejecución de red de una capa oculta sobre GPU

6. Conclusiones y futuros trabajos

6.1. Conclusiones

En este trabajo se logró predecir la ocurrencia de heladas, así como sus características de intensidad y duración mediante el uso de estrategias de Deep Learning y Machine Learning.

Para la predicción de ocurrencia de helada, se utilizaron las estrategias de regresión logística y Random Forest, donde ambos modelos arrojan resultados, en cuanto a exactitud, similares, pero centrándonos en la métrica de *recall*, que tiene mayor relevancia en esta ocasión, para la reducción de los falsos negativos, Random Forest tiene un rendimiento notablemente superior, donde es capaz de identificar el 79% de las heladas a una distancia de tres horas comparado con Regresión Logística, con un 71%, ventaja que se va ampliando conforme se incrementa la distancia de tiempo la cual se quiere predecir.

Tabla 21Métricas obtenidas para distintos modelos en la predicción de ocurrencia de heladas

Estrategia	Algoritmo	Valor	Métricas			
			Precisión	Recall	F1 Score	Exactitud
Ventanas de tiempo	Regresión logística	Falso	0.90	0.94	0.92	0.87
		Verdadero	0.64	0.50	0.56	
	Random Forest Clasificador	Falso	0.93	0.94	0.94	0.89
		Verdadero	0.68	0.68	0.68	
	Perceptrón Multicapa	Falso	0.94	0.94	0.92	0.87
		Verdadero	0.61	0.71	0.65	
	Red neuronal FF	Falso	0.95	0.91	0.91	0.85
		Verdadero	0.56	0.55	0.55	
	Red neuronal LSTM	Falso	0.91	0.95	0.93	0.89
		Verdadero	0.83	0.71	0.77	
Promedios meteorológicos diarios	Random Forest Clasificador	Falso	0.82	0.83	0.83	0.81
		Verdadero	0.80	0.79	0.79	
	Random Forest Regresor	Falso	0.84	0.80	0.82	0.80
		Verdadero	0.76	0.79	0.77	

Llegado a la distancia máxima propuesta, de 6 horas, la estrategia de Regresión Logística se comporta pobremente, identificando tan solo un 11% de las heladas, mientras que Random Forest mantiene su rendimiento, llegando a identificar un 39% de las heladas ocurridas a esta distancia de tiempo.

En la Tabla 21 se puede observar una comparación de las métricas para los resultados obtenidos con diferentes estrategias y modelos. Para la estrategia de ventanas de tiempo, se consideró una ventana de tres horas para establecer las comparaciones.

En cuanto a los modelos de Machine Learning utilizados, redes neuronales FeedForward (FF) y LSTM, la primera pudo identificar las heladas con una anticipación de 3 horas el 51% de los casos, mientras que la red LSTM, de mayor complejidad lo logró en el 71% de las ocasiones, utilizando la metodología de ventanas de tiempo. Cuando se compararon los tiempos de ejecución en CPU contra los tiempos en GPU, se encontró que para la red neuronal FF la reducción de la demora al entrenar el modelo fue del 6,25%, mientras que para la red LSTM esta reducción fue de un 69,29%. Esta gran diferencia en la reducción de tiempos de entrenamiento está relacionada a la complejidad de la red. La red neuronal LSTM es más compleja que la FF, esto es, requiere mucho mayor poder computacional, por lo tanto, tiene un grado mucho mayor de aprovechamiento de los recursos de la GPU. Esto se logró dado que el módulo de Knime para Deep Learning utilizados funciona con bibliotecas de Python como TensorFlow y Keras, que permiten aprovechar al máximo la capacidad de procesamiento masivo de las GPU para acelerar el tiempo de ejecución de las tareas de cómputo intensivo, como lo son las redes neuronales abordadas.

Respecto a la estrategia por promedios meteorológicos diarios, propuesta de forma autónoma en este trabajo, se observa que para la predicción de la ocurrencia de helada este modelo se comporta de forma altamente aceptable, logrando identificar el 83% de las heladas. De acuerdo a esta estrategia, esto quiere decir que el productor agropecuario podrá saber si ocurrirá una helada durante la noche, a las 20:00 horas de ese día, con la exactitud antes mencionada. En cuanto a la predicción de la duración e intensidad de la helada, se pudo observar que bajo esta estrategia tanto los modelos clasificadores como regresores de Random Forest arrojan predicciones no satisfactorias para la clasificación de estas características en las categorías leves, o cortas, y medias, pero sí pudo identificar las heladas extensas en un 77% de los casos, y las heladas intensas en un 83% de los mismos. De esto se desprende que los modelos son perfectibles en cuanto a la identificación de heladas de intensidad baja y media y de duración media y breve, probablemente debido a la relativa escasez de datos para estas categorías bajo

esta estrategia de agrupación de datos. En la Tabla 22 se observa una comparación de estas métricas mencionadas.

Tabla 22 Comparación de métricas de los resultados para la clasificación de intensidad y duración de heladas

Característica	Algoritmo	Valor	Métricas			
			Precisión	Recall	F1 Score	Exactitud
Intensidad	Random Forest Regresor	Bajo	0.34	0.28	0.31	060
		Medio	0.36	0.21	0.27	
		Alto	0.27	0.83	0.41	
	Random Forest Clasificador	Bajo	0.08	0.18	0.11	0.66
		Medio	0.27	0.26	0.26	
		Alto	0.71	0.70	0.71	
Duración	Random Forest Regresor	Bajo	0.75	0.16	0.26	0.27
		Medio	0.44	0.27	0.33	
		Alto	0.46	0.65	0.54	
	Random Forest Clasificador	Bajo	0.08	0.23	0.12	0.63
		Medio	0.33	0.30	0.31	
		Alto	0.86	0.77	0.81	

En lo particular, al que suscribe, desde tareas secuenciales y concatenadas que surgen del haber obtenido y trabajado sobre la temática de visualización de información mediante la utilización de bibliotecas del entorno de programación Python en una beca CIN y el posterior desarrollo del trabajo final sobre el área de conocimiento predicción de heladas permitió un crecimiento cualitativo y cuantitativo en la temática que se evidencia en diferentes publicaciones realizadas en el marco del proyecto “Evaluación de Visualizaciones Eficientes en Ciencia de Datos” a lo largo de trienio en que el mismo se ejecutó.

6.2. Futuros trabajos

Del desarrollo de este trabajo, se desprenden directamente dos propuestas de trabajo a futuro continuando la labor realizada en el presente trabajo.

La primera es la investigación e implementación nuevas estrategias de DL y ML para poder mejorar la calidad de las predicciones, haciendo énfasis en el *recall,* de la clasificación de la duración e intensidad de las heladas en las categorías bajas y medias. La segunda es la implementación de la combinación de los modelos de predicción a distancia de horas con el modelo de promedios meteorológicos diarios, aprovechando las ventajas de ambos: predecir exactamente en qué hora ocurrirá la helada, por parte del primero, y la mayor calidad de las predicciones, así como la posibilidad de identificar la duración e intensidad de la helada, por parte del segundo.

Por otro lado, considerando la dinámica exponencial del crecimiento de las herramientas de Inteligencia Artificial, así como de Aprendizaje Profundo, es posible realizar trabajos que de un paso más allá a la hora de solucionar las problemáticas de los agroproductores, integrando los métodos presentes en este trabajo así como las dos alternativas de trabajo futuro propuestas, con otras plataformas dedicadas a la automatización y optimización del uso del agua y al estudio de los suelos para la mejora del uso de fertilizantes con el fin de ofrecer al productor agropecuario una solución que de respuesta a sus diferentes necesidades de manera integral.

7. Referencias bibliográficas

Bibliografía

1. *Development of an index for frost prediction: Technique and validation.* **J. R. Rozante, E. R. Gutierrez, P. L. da Silva Dias, A. de Almeida Fernandes, D. S. Alvim, and V. M. Silva,.** s.l. : Meteorol Appl., 2019.

2. *Frost Prediction using Machine Learning and Deep.* **Carl J. Talsmaad, Kurt C. Solandera, Maruti K. Mudunurub , Brandon Crawforda.** Bloomington, IN : C.J.T., K.C.S., and B.C. Authors are with the Earth and Environmental Sciences Division, Los Alamos National Laboratory, MS T003, Los Alamos, NM 87545 (Corresponding author, Carl Talsma: talsmac83@lanl.gov). 5 b M.K.M. is with the Watershed & Ecosystem , 2022.

3. *Prediction of Frost Events using Bayesian networks and Random Forest.* **Ana Laura Diedrichs, Facundo Bromberg, Diego Dujovne, Keoma Brun-Laguna, Thomas Watteyne.** s.l. : IEEE internet of things journal, 2018.

4. *Técnicas de balanceo de datos para predecir la ocurrencia del fenómeno meteorológico de la helada.* **María Isabel Masanet, Raúl Oscar Klenzi, Flavio Capraro.** 2021, Facultad de Ciencias Exactas, Físicas y Naturales , Instituto de Automática (INAUT) , Universidad Nacional de San Juan.

5. *PROCESAMIENTO DE DATOS METEOROLÓGICOS PARA DETERMINAR LA OCURRENCIA DE HELADAS EN LA AGRICULTURA.* **María Masanet, Flavio Capraro, Raúl Klenzi, Martín Muñoz.** San Juan, Argentina : Instituto de Informática / Departamento de Informática /Facultad de Ciencias Exactas Físicas y Naturales / Universidad Nacional de San Juan, 2021.

6. *Forecasting Severe Weather with Random Forests.* **Aaron J. Hill, Gregory R. Herman, Russ S. Schumacher.** Colorado : Department of Atmospheric Science, Colorado State University, Fort Collins, Colorado, 2020.

7. *Procesamiento de datos meteorológicos para determinar la ocurrencia, intensidad y duración de heladas.* **Joaquin Cortez, María Masanet, Raul Klenzi, Manuel Ortega.** San Juan : 51 JAIIO, 2022.

8. **AWS.** What is data science? [En línea] AWS. [Citado el: 19 de 10 de 2022.] https://aws.amazon.com/what-is/data-science/.

9. **Universidad Autónoma de Madrid.** La metodología CRISP-DM en ciencia de datos. *IIC UAM.* [Online] [Cited: 02 16, 2023.] www.iic.uam.es/innovacion/metodologia-crisp-dm-ciencia-de-datos/.

10. **IBM.** What is Artificial Intelligence? [Online] IBM, 3 6, 2020. [Cited: 10 27, 2022.] https://www.ibm.com/cloud/learn/what-is-artificial-intelligence.

11. **UC Berkeley.** What is Machine Learning. [Online] UC Berkeley, 06 26, 2020. [Cited: 10 27, 2022.] https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/.

12. **Peter Norvig, Stuart J. Russell.** *Artificial Intelligence: A Modern Approach 3rd Edition.* s.l. : Pearson, 2009.

13. **IBM.** What is Machine Learning. [Online] IBM, 07 15, 2020. [Cited: 10 26, 2022.] https://www.ibm.com/cloud/learn/machine-learning.

14. **Tableau.** Tableau Articles. *Time Series Analysis: Definition, Types, Techniques, and When It's Used.* [Online] [Cited: 02 14, 2022.] https://www.tableau.com/learn/articles/time-series-analysis. .

15. **Peixeiro, Marco.** Medium. *The Complete Guide to Time Series Analysis and Forecasting.* [Online] 08 07, 2019. [Cited: 02 14, 2022.] https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775.

16. **Géron, Aurélien.** *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 2ed.* s.l. : O'Reilly Media, 2019.

17. **IBM.** What is Random Forest. [Online] IBM, 12 07, 2020. [Cited: 10 28, 2022.] https://www.ibm.com/cloud/learn/random-forest.

18. *Comparison of Cloud-Filling Algorithms for Marine Satellite Data.* **Stock, A., et al.** s.l. : Remote Sens, 2020.

19. **IBM.** What are Neural Networks? [Online] IBM, 08 17, 2020. [Cited: 11 01, 2022.] https://www.ibm.com/cloud/learn/neural-networks.

20. **Baheti, Pragati.** Activation Functions in Neural Networks [12 Types & Use Cases]. *V7 Labs.* [Online] 02 03, 2023. [Cited: 02 15, 2023.] https://www.v7labs.com/blog/neural-networks-activation-functions.

21. **Aggarwal, Charu C.** *Neural Networks and Deep Learning: A Textbook 1ed.* s.l. : Springer, 2018.

22. **Colah.** Understanding LSTM Networks. [Online] 08 27, 2015. [Cited: 11 9, 2022.] https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

23. **Barlas, Gerassimos.** *Multicore and GPU Programming: An Integrated Approach.* s.l. : Morgan Kaufmann , 2014.

24. **Intel.** What Is a GPU? [Online] 10 03, 2021. [Cited: 11 10, 2022.] https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html.

25. **Dsouza, Jason.** What is a GPU and do you need one in Deep Learning? [Online] Medium, 04 25, 2020. [Cited: 11 10, 2022.] https://towardsdatascience.com/what-is-a-gpu-and-do-you-need-one-in-deep-learning-718b9597aa0d.

26. **Lucía Guadalupe Matías Ramírez, Óscar Arturo Fuentes Mariles y Fermín García Jiménez.** *Heladas.* México DF : CENAPRED, 2001.

27. **Richard L Snyder, J. Paulo de Melo-Abreu.** *Protección contra las heladas: fundamentos, práctica y economía.* Roma : FAO, 2010.

28. **TIOBE.** TIOBE Index for November 2022. [Online] TIOBE, 11 01, 2022. [Cited: 11 24, 2022.] https://www.tiobe.com/tiobe-index/.

29. **Octoverse.** The top programming languages in Github. [Online] Octoverse, 01 01, 2022. [Cited: 11 24, 2022.] https://octoverse.github.com/2022/top-programming-languages.

30. **Ruiz, Pablo.** ML Approaches for Time Series . *Medium.* [Online] 05 19, 2019. [Cited: 02 19, 2022.] https://towardsdatascience.com/ml-approaches-for-time-series-4d44722e48fe.

31. **Snyder, Richard L.** *Protección contra las heladas: fundamentos, práctica y economía.* 2010.

32. **Konca, Mustafa Serdar.** Data Science Life Cycle. [Online] 05 18, 2022. [Cited: 10 24, 2022.] https://mustafaserdarkonca.medium.com/data-science-life-cycle-e4d74afe4bf5#fd80.

33. **Microsoft.** Learn to code with Visual Studio Code . [Online] VSCode, 05 17, 2022. [Cited: 11 24, 2022.] https://code.visualstudio.com/learn.

34. **Ramalho, Luciano.** *Fluent Python* . s.l. : O'Reilly Media, 2015.

35. **Rosario Silipo, Jeanette Prinz.** *KNIME Advanced Luck A Guide to KNIME Analytics Platform for Advanced Users.* s.l. : KNIME Press, 2019.

36. **Silipo, Rosario.** *KNIME Beginners Luck.* s.l. : KNIME Press, 2018.

37. **Goodfellow, Ian.** *Deep Learning (Adaptive Computation and Machine Learning series)* . s.l. : The MIT Press, 2016.