DESARROLLO DE MODELOS PREDICTIVOS PARA LA ESTIMACIÓN DE LA TEMPERATURA SUPERFICIAL DE LOS DETECTORES DEL OBSERVATORIO PIERRE AUGER

ANDRES FELIPE RODRIGUEZ GRANADOS
LIBARDO LUIS MUÑOZ MARTINEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICO-MECANICAS
ESCUELA DE INGENIERIA DE SISTEMAS E INFORMATICA
BUCARAMANGA
2018

DESARROLLO DE MODELOS PREDICTIVOS PARA LA ESTIMACIÓN DE LA TEMPERATURA SUPERFICIAL DE LOS DETECTORES DEL OBSERVATORIO PIERRE AUGER

ANDRES FELIPE RODRIGUEZ GRANADOS
LIBARDO LUIS MUÑOZ MARTINEZ

Trabajo de grado para optar el título de Ingeniero de Sistemas

DIRECTOR
RAUL RAMOS POLLAN
PhD. en Ingeniería en Informática

CODIRECTOR
LUIS ALBERTO NUÑEZ DE VILLAVICENCIO MARTINEZ
PhD. en Ciencias

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICO-MECANICAS
ESCUELA DE INGENIERIA DE SISTEMAS
BUCARAMANGA
2018

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

GLOSARIO

DATASET: Es una colección de datos tabulada que corresponde a una única base de datos o a una única matriz de datos, donde cada columna de la tabla representa una variable en particular, y cada fila representa a un miembro determinado del conjunto de datos.

DETECTOR CHERENKOV: Es un detector construido con fotomultiplicadores, un panel solar y plaquetas electrónicas de adquisición de datos, que tiene como fin registrar la caída de rayos cósmicos mediante el efecto Cherenkov, el cual se produce cuando una partícula cargada se mueve en un medio transparente con una velocidad mayor que la que la luz tendría en ese medio.

MACHINE LEARNING: Rama de la inteligencia artificial que tiene como objetivo desarrollar técnicas que le permiten a las máquinas aprender sin ser específicamente programadas, mediante un proceso de inducción. En un programa de aprendizaje automático se provee de un conjunto de datos de ejemplo (experiencia) a partir de los cuales se construye un modelo que calcula un resultado aproximado a los datos reales.

PYTHON: Es un lenguaje de programación de alto nivel, interpretado y de código abierto ampliamente usado. Cuenta con una gran cantidad de librerías que lo hacen una herramienta adaptable a las necesidades de los desarrolladores.

TEMPERATURA SUPERFICIAL: Es una medida que representa el calentamiento directo de la superficie terrestre, donde los rayos del sol son absorbidos y reemitidos.

RESUMEN

TITULO: DESARROLLO DE MODELOS PREDICTIVOS PARA LA ESTIMACIÓN DE LA TEMPERATURA SUPERFICIAL DE LOS DETECTORES DEL OBSERVATORIO PIERRE AUGER*

AUTORES: MUÑOZ MARTÍNEZ, Libardo Luis
RODRÍGUEZ GRANADOS, Andrés Felipe**

PALABRAS CLAVE: Machine Learning, Detector Cherenkov, Pierre Auger.

DESCRIPCIÓN:

El observatorio Pierre Auger ubicado en la provincia de Mendoza, Argentina, tiene como finalidad el estudio de partículas subatómicas provenientes del espacio exterior denominadas rayos cósmicos las cuales se le desconoce su naturaleza, su origen y el mecanismo que le imparte sus velocidades. El estudio de estas, es posible gracias a que el observatorio cuenta con un arreglo de 1660 detectores Cherenkov distribuidos en un área de 3000 Kilómetros cuadrados el cual está dividido en 4 zonas conocidas como Los Leones, Loma Amarilla, Los Morados y Coihueco. Cada una de estas tiene a su disposición una estación meteorológica que tiene como objetivo el monitoreo en tiempo real de la temperatura superficial de cada una de estas, lo que posibilita el estudio respectivo que se lleva a cabo en el observatorio. Por otro lado, a pesar de que los detectores posean una amplia gama de sensores, prescinden de aquellos que les permitan medir la temperatura del aire sobre estos mismos. Por esta razón, utilizando datos de algunos sensores de los detectores más cercanos a cada una de las distintas estaciones y los datos de temperatura de estas mismas, es posible llevar a cabo el ejercicio de exploración de modelos predictivos para estimar la temperatura superficial de cada detector.

* Trabajo de grado
** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Ph.D Raúl Ramos Pollan. Codirector:Ph.D Luis Alberto Núñez.

ABSTRACT

TITLE: DEVELOPMENT OF PREDICTIVE MODELS FOR THE ESTIMATION OF THE SURFACE TEMPERATURE OF DETECTORS IN THE PIERRE AUGER OBSERVATORY*

AUTHORS: MUÑOZ MARTÍNEZ, Libardo Luis
RODRÍGUEZ GRANADOS, Andrés Felipe**

KEYWORDS: Machine Learning, Cherenkov Detector, Pierre Auger.

DESCRIPTION:

The Pierre Auger Observatory located in the province of Mendoza, Argentina, has as its purpose the study of subatomic particles coming from outer space called cosmic rays which are unknown its nature, its origin and the mechanism that imparts its speeds. The study of these, is possible thanks to the observatory has an array of 1660 Cherenkov detectors distributed in an area of 3000 square kilometers which is divided into 4 areas known as Los Leones, Loma Amarilla, Los Morados and Coihueco. Each one of them has at its disposal a meteorological station that has the objective of real-time monitoring of the surface temperature of each of these, which makes possible the respective study that is carried out in the observatory. On the other hand, in spite of the fact that the detectors have a wide range of sensors, they do without those that allow them to estimate the air temperature over them. For this reason, using data from some sensors of the detectors closest to each of the different stations and the temperature data of these, it is possible to carry out the exercise of scouting of predictive models to estimate the surface temperature of each detector.

* Bachelor Thesis

** Faculty of Physical-Mechanical Engineering. School of Engineering and Computer Science. Director:Ph.D Raúl Ramos Pollan. Codirector:Ph.D Luis Alberto Núñez

INTRODUCCIÓN

El análisis de datos es una de las áreas en auge con mayor presencia en la actualidad, involucrándose en campos como la economía para predecir el comportamiento de los mercados; las redes sociales para sugerir contenido en función de los gustos de sus usuarios; la biología para predecir los ritmos de crecimientos de determinados seres vivos o incluso en ciencias del comportamiento para analizar la toma de decisiones con las que se enfrentan las personas día a día.

Mientras se disponga de suficientes datos para entrenar los modelos, la aplicación de análisis de datos utilizando algún tipo de enfoque como el de aprendizaje automático (*Machine Learning*), pueden ser utilizados en cualquier área posible. Por esta razón, al reflexionar acerca del inconveniente con el que lidia un observatorio en Argentina, se concluyó que era necesario explorar modelos basados en estos principios anteriormente expuestos para poder aportar soluciones.

El observatorio Pierre Auger, situado en la ciudad de Malargüe, en la provincia de Mendoza, Argentina, funciona gracias a la colaboración de 16 países (al menos unos 500 científicos de 100 instituciones, siendo una de esas la Universidad Industrial de Santander), como una iniciativa que tiene como fin investigativo la detección de partículas subatómicas provenientes del espacio exterior, denominadas rayos cósmicos. Para esto, se vale del uso de un arreglo de 1660 detectores Cherenkov, los cuales están ubicados estratégicamente en toda la zona. Estos detectores, tienen diversos sensores utilizados para el monitoreo general de su buen funcionamiento y estado.

Aunque todos estos detectores tienen sensores de temperatura de la electrónica interna del mismo (baterías, fotomultiplicadores, etc), no existe una relación directa

entre la temperatura real del desierto, la cual se mide gracias a varias estaciones meteorológicas situadas en los extremos del desierto, con la temperatura obtenida por los detectores Cherenkov. Esto, debido a los múltiples componentes electrónicos internos de cada detector, y a la distancia existente de varios kilómetros que separa las estaciones meteorológicas con los detectores, los cuales afectan a la precisión de monitoreo de la temperatura atmosférica real.

El proyecto está enfocado en el desarrollo de modelos de *Machine Learning*, que permitan estimar la temperatura de todos los detectores del arreglo del observatorio Pierre Auger, usando datos de monitoreo provenientes de las estaciones meteorológicas, los cuales se toman de referencia para entrenar los datos de los detectores. En las notas del proyecto se describen los datos utilizados en los ensayos, las metodologías de *Machine Learning* diseñadas y aplicadas, las configuraciones de pruebas y finalmente los resultados predictivos obtenidos junto a los trabajos posteriores a abordar.

1. DEFINICIÓN DEL PROBLEMA

En el observatorio Pierre Auger, se llevan a cabo diferentes investigaciones científicas, que requieren de un conocimiento en tiempo real sobre diversos parámetros externos. Esto, con el fin de evitar alguna afectación causada por dichos parámetros, de los cuales, uno de los más influyentes es la temperatura superficial, definida como una medida que representa el calentamiento directo de la superficie terrestre, donde los rayos del sol son absorbidos y remitidos.

Debido a que la ubicación del observatorio es una zona desértica muy extensa, la temperatura superficial puede presentar cambios muy bruscos en el transcurso del día. Aun así, gracias al uso de unas pocas estaciones meteorológicas situadas en los extremos del desierto, se logra registrar la temperatura con precisión de un grado, en intervalos de 300 segundos.

Los 1660 detectores Cherenkov utilizados para las investigaciones, por su parte, registran datos usando sus diferentes sensores, en intervalos de 400 segundos. Estas mediciones están relacionadas con la temperatura de la superficie, pero esta relación no es obvia, ya que diferentes condiciones podrían introducir desplazamientos entre las condiciones de aire externo y la temperatura de alguno de los componentes internos.

En el proyecto se aborda la idea de determinar una relación precisa entre la temperatura del aíre local registrada por las estaciones meteorológicas en los extremos del arreglo y los datos obtenidos de los sensores de cada uno de los detectores Cherenkov, a lo largo de toda la superficie del arreglo. Se propuso comprender esta relación y crear modelos de *Machine Learning*, para predecir la temperatura de la superficie entrenando datos de monitoreo de los detectores,

usando registros obtenidos por las estaciones meteorológicas. Teniendo en cuenta, de que existen elecciones a realizar, con el fin de parametrizar la metodología aplicada en el proyecto, desde la cantidad y el tipo de datos a usar para entrenar los modelos predictivos, hasta el tiempo y el tipo de entrenamiento aplicado.

El proyecto es aplicable a largo plazo en el mejoramiento de los cálculos de los efectos meteorológicos actuales mediante el uso de las condiciones atmosféricas locales, y aporta en aplicaciones de investigación en otras áreas relacionadas muy probablemente al clima y la temperatura.

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Explorar la creación de modelos predictivos para la estimación de variables atmosféricas en el arreglo de detectores del observatorio Pierre Auger, partiendo de datos de monitoreo de los mismos.

2.2 OBJETIVOS ESPECÍFICOS

- Generar datasets consolidados desde los ficheros de log de monitoreo del arreglo para el entrenamiento y validación de modelos predictivos.

- Definir distintos diseños experimentales que evidencien la capacidad de reutilización de modelos creados con datos de distintas partes del arreglo.

- Entrenar distintos algoritmos y procesos de *Machine Learning* para cada diseño experimental.

- Evaluar el desempeño predictivo para cada diseño experimental y modelo entrenado.

3. MARCO TEÓRICO

3.1 OBSERVATORIO PIERRE AUGER

El Observatorio Pierre Auger, situado en la ciudad de Malargüe, en la provincia de Mendoza, Argentina, es una iniciativa conjunta de 16 países en la que colaboran unos 450 científicos de 60 instituciones, con la finalidad de detectar partículas subatómicas que provienen del espacio exterior denominadas rayos cósmicos.

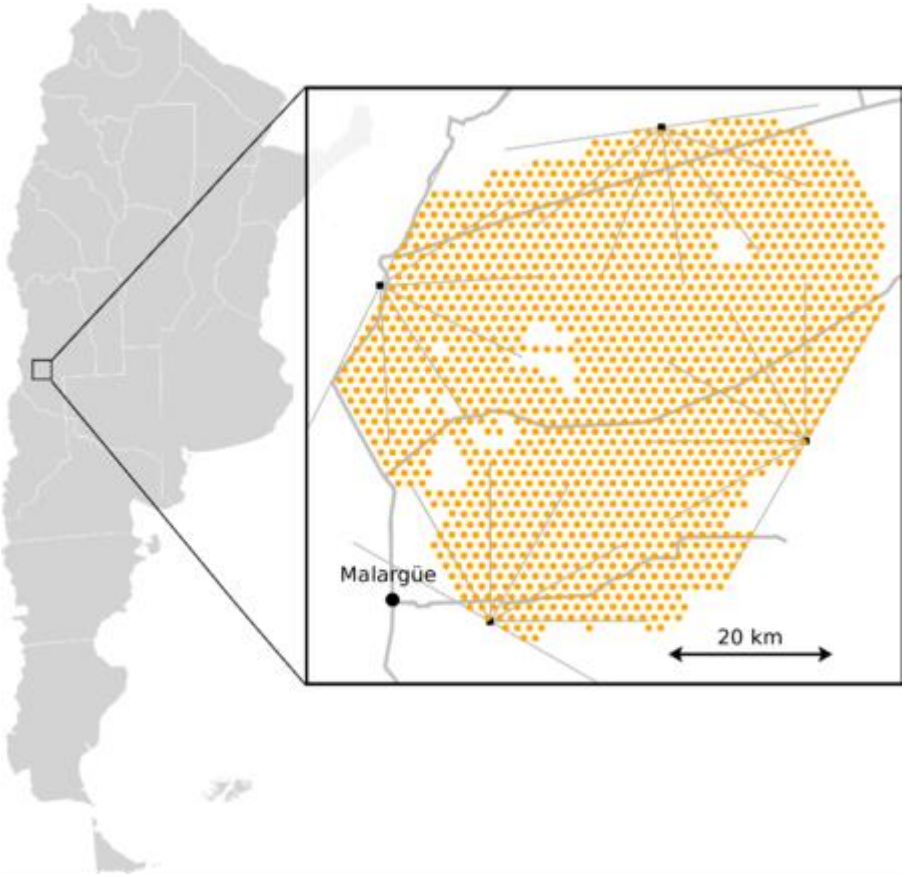
Figura 1. Países que colaboran en el observatorio Pierre Auger

Fuente: AUGER Country [en línea] disponible en: https://www.auger.org/index.php/about-us/institutions#per-country

Los rayos cósmicos son partículas cargadas provenientes del espacio, que caen permanentemente sobre la Tierra. Estos rayos tienen energías anormalmente

superiores a los que usualmente bombardean la Tierra y producen un efecto llamado lluvia cósmica o cascada atmosférica extensa. Para registrar una gran cantidad de estos rayos cósmicos, el área de detección es de 3000 km^2, convirtiéndose en el Observatorio de rayos cósmicos de mayor tamaño en el planeta. Esta región desértica, se divide en cuatro zonas las cuales han sido denominadas: Los Leones, Loma Amarilla, Los Morados, y Coihueco[1].

Figura 2. Ubicación del observatorio Pierre Auger en Malargüe, Mendoza



Fuente: CNEA [en línea] disponible en: http://fisica.cab.cnea.gov.ar/particulas/html/labdpr/auger.php

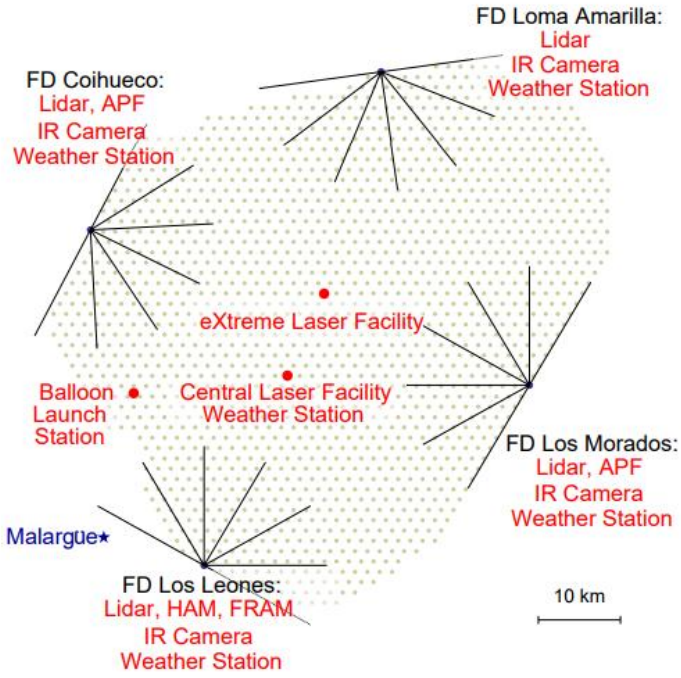
3.1.1 Estaciones Meteorológicas. En el observatorio Pierre Auger operan unas estaciones meteorológicas que son utilizadas para el monitoreo y control de la

[1] CNEA Laboratorio Detección De Partículas y Radiación [en línea]. <http://fisica.cab.cnea.gov.ar/particulas/html/labdpr/auger.php> [citado en 3 de enero del 2018]

región. Estas estaciones, son instalaciones equipadas con sensores de temperatura, presión, humedad y velocidad del viento, los cuales registran datos en intervalos de 300 segundos.

En toda la región del observatorio, hay un total de cinco estaciones meteorológicas, de las cuales hay cuatro ubicadas en cada una de las zonas del observatorio (Los Leones, Loma Amarilla, Los Morados, Coihueco), y la quinta ubicada en el centro de la región. Los datos recolectados por cada una de las estaciones meteorológicas son transferidos, procesados y finalmente almacenados, en las bases de datos para información de monitoreo atmosférico.

Figura 3. Mapa de las estaciones meteorológicas



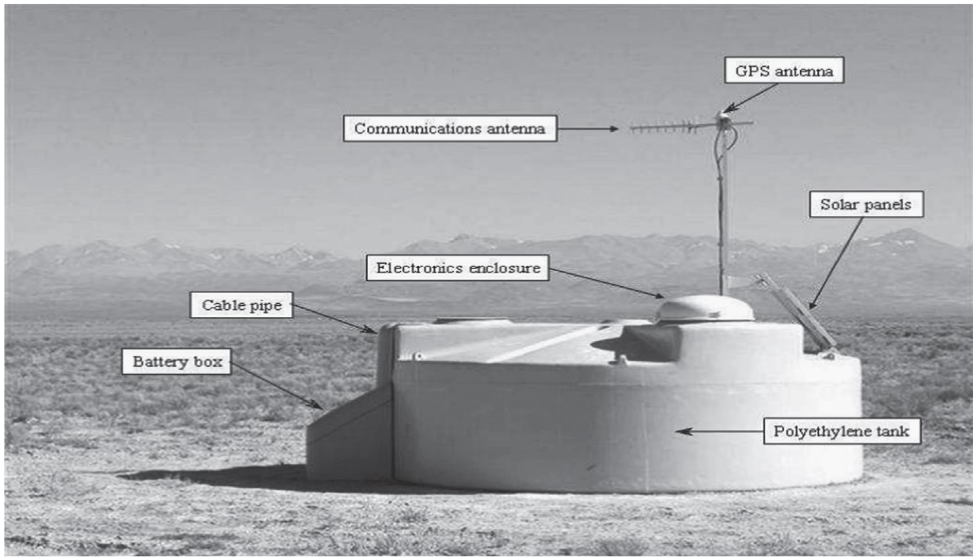
Fuente: ARXIV [en línea] disponible en: https://arxiv.org/ftp/arxiv/papers/1107/1107.4806.pdf

3.1.2 Detectores Cherenkov. Detectores que se basan en el efecto Cherenkov, el cual se produce cuando una partícula cargada se mueve en un medio transparente

con velocidad mayor que la que la luz tendría en ese medio, tienen por finalidad registrar la caída de rayos cósmicos.

Cada detector consiste en un tanque de agua con 3.6 m de diámetro y 1.2 m de altura, que contiene un revestimiento sellado con una superficie interior reflectante; el cual contiene 12.000 litros de agua purificada. De manera distribuida, el detector Cherenkov tiene tres fotomultiplicadores *PhotonisXP1805/D1* (*PMT*) que se encargan de registrar la luz Cherenkov producida por el paso de partículas cargadas a través del agua. Tiene además unos componentes electrónicos como un procesador, un receptor GPS, un transceptor de radio y un controlador de potencia. Cada detector es autónomo gracias a un sistema de energía solar, que proporciona un promedio de 10W para los diferentes componentes electrónicos. Las baterías se instalan en una caja de polietileno resistente a la luz solar directa, ya que el aumento de la temperatura reduce drásticamente el tiempo de vida de las baterías. Los paneles solares de cada detector están instalados sobre soportes de aluminio, los cuales están diseñados para resistir vientos de 160 km/h, debido a que en su parte superior también sostienen las antenas para la comunicación de radio y la recepción del GPS. Los revestimientos de cada detector son cilindros circulares hechos de un material plástico flexible que se adapta a la superficie interior del detector, brindándole un ambiente hermético. Los sensores del detector proporcionan información del estado de cada uno de los componentes internos del mismo, esta información es enviada mediante un registro de datos en intervalos de 400 segundos.

Figura 4. Detector Cherenkov con sus principales componentes



3.1.2.1 Operación y Mantenimiento. Actualmente hay 1660 detectores Cherenkov, de los cuales se han detectado muy pocos fallos, gracias a que la electrónica del detector funciona con energía solar, y un sistema de control de potencia incorpora circuitos de protección, acondicionamiento de señal para el monitoreo del sistema de energía solar, y un circuito que permite el apagado y encendido ordenado en caso de un periodo prolongado donde la energía solar no pueda ser reabastecida eficientemente, de manera que no se afecte la operatividad continua del detector.

Los fotomultiplicadores (*PMT*) sin embargo, son los elementos más críticos del detector, debido a que están sujetos a condiciones ambientales muy severas como variaciones de temperatura, humedad, salinidad y polvo. Presentando una tasa de fallas de 20 por año (aproximadamente 0.5%), se ha creado una gran reserva en la que se estima que la cantidad de *PMT* para repuesto es suficiente para 10 años más de operación.

Aunque la mayoría de los problemas de los detectores deben repararse mediante el uso de repuestos, algunos problemas se reparan simplemente refinando el software.

El objetivo de mantenimiento es no tener más de 20 detectores fuera de servicio en un mismo instante. Actualmente, el número alcanzado es inferior a 10 detectores fuera de servicio[2].

3.2 TEMPERATURA SUPERFICIAL

La temperatura superficial se define como una medida que representa el calentamiento directo de la superficie terrestre, donde los rayos del sol son absorbidos y reemitidos. Esta superficie, se define como el espacio existente entre la atmósfera y el suelo. Por lo tanto, se puede conocer que tan caliente se siente la superficie de la Tierra al tacto en un lugar determinado.

La temperatura superficial es un parámetro clave en muchas aplicaciones de energía, como los modelos de evaporación, los modelos climáticos y los modelos de transferencia radiante. El uso de la superficie de la Tierra puede afectar el clima circundante, como los vapores de agua, las moléculas de gas y la formación de nubes. De esta manera, puede afectar a todo un ecosistema generando que empiece a presentar fluctuaciones muy fuertes en la temperatura de la superficie. Cuando se trata de crear predicciones meteorológicas, es importante que la parametrización sea altamente integrada de la superficie de la tierra, como la

[2] THE PIERRE AUGER COSMIC RAY OBSERVATORY. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associate Equipment, North-Holland, 8 de julio del 2015 [en línea]. <https://www.sciencedirect.com/science/article/pii/S0168900215008086> [citado en 4 de enero del 2018]

humedad del suelo y la temperatura superficial, para poder generar que dichas predicciones sean lo más exactas posibles[3].

3.3 MACHINE LEARNING

Machine Learning se define como la ciencia y el arte de la programación de computadoras, de manera que estas puedan aprender a partir de los datos. Una definición más orientada a la ingeniería, expresa que un programa de computadora aprende de la experiencia E con respecto a alguna tarea T y alguna medida de rendimiento P, si su desempeño en T, medido por P, mejora con la experiencia E.

Por ejemplo, el filtro de spam es un programa de aprendizaje automático que puede aprender a marcar correo no deseado con ejemplos de correos electrónicos no deseados y ejemplos de correos regulares. Los ejemplos que el sistema usa para aprender se llaman conjunto de entrenamiento. Cada ejemplo de capacitación se denomina instancia de capacitación (o muestra). En este caso, la tarea T es marcar spam para nuevos correos electrónicos, la experiencia E es la información de entrenamiento y la medida de rendimiento P necesita ser definida; por ejemplo, puede usar la proporción de correos electrónicos correctamente clasificados. Esta medida de rendimiento particular se llama precisión y se usa a menudo en tarea de clasificación.

Cuando el problema no es trivial, y éste se programa con un enfoque tradicional, el programa se convertirá en una larga lista de reglas complejas, bastante difícil de mantener. Por el contrario, mediante aprendizaje automático el programa resulta ser más corto, más fácil de mantener, y con toda probabilidad, más preciso.

[3] LAND SURFACE TEMPERATURE AND SCALING Factors for Different Satellites Datasets [en línea]. <https://www.omicsonline.org/open-access/land-surface-temperature-and-scaling-factors-for-different-satellitesdatasets-jgg-1000223.pdf> [citado en 4 de enero del 2018]

Figura 5. Enfoque tradicional

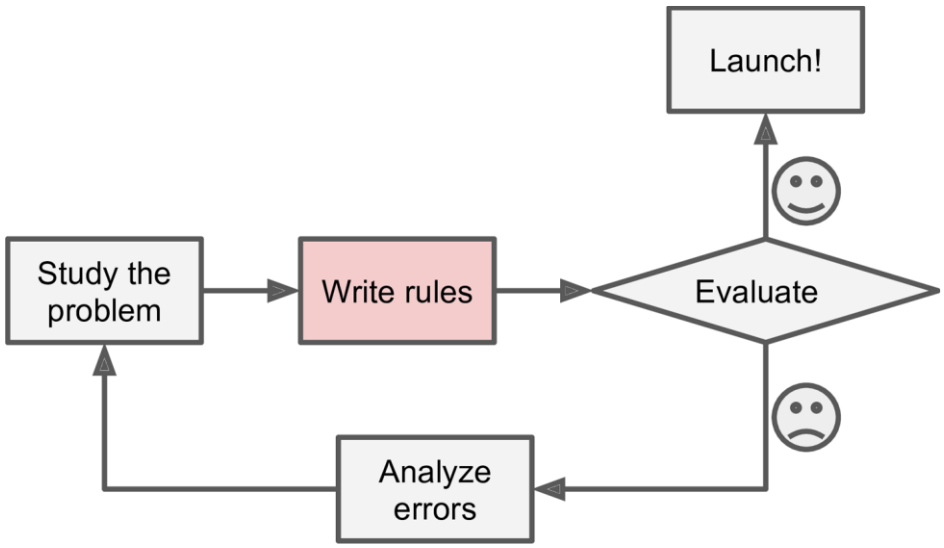
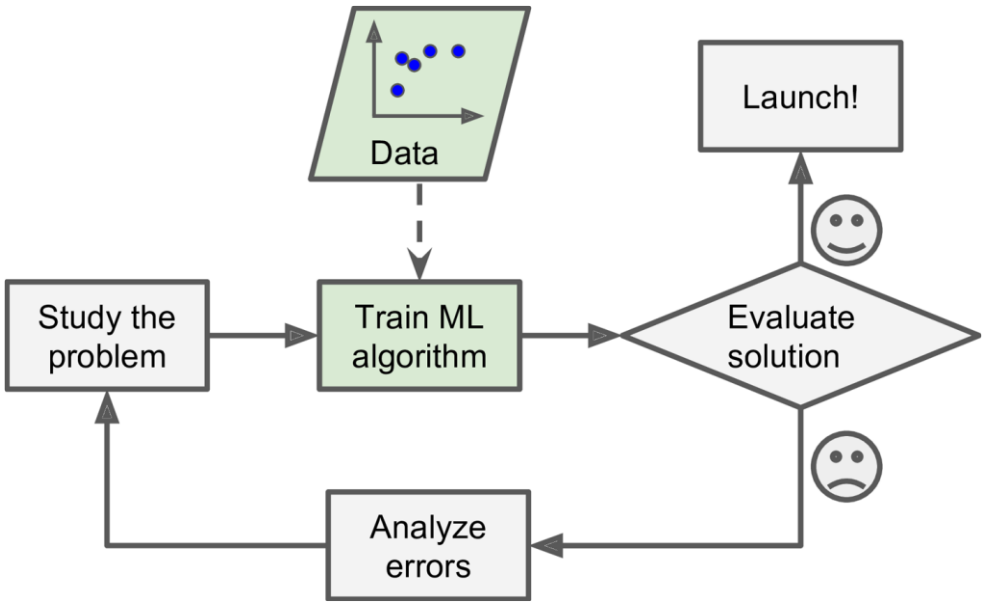


Figura 6. Enfoque de aprendizaje automático



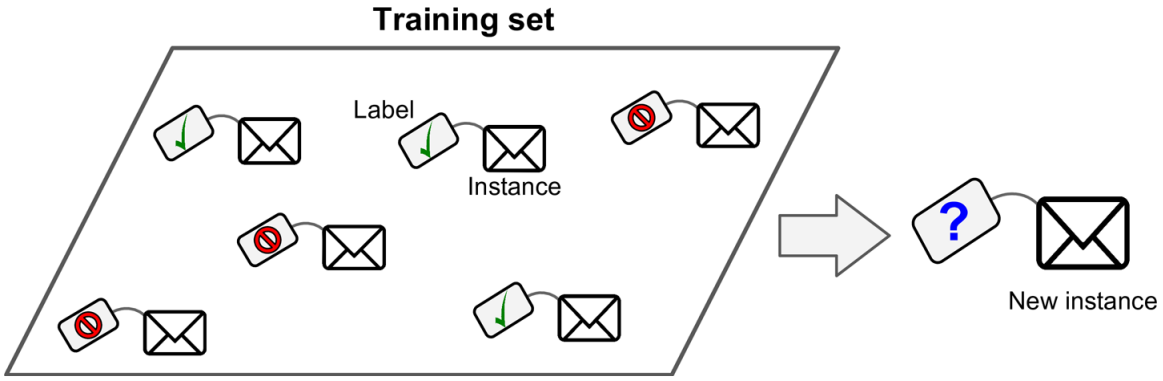
Hay tantos tipos diferentes de sistemas de aprendizaje automático que es útil clasificarlos en categorías amplias en función de:

- Si están o no entrenados con supervisión humana.

- Si pueden o no aprender incrementalmente sobre la marcha.
- Si funcionan simplemente comparando nuevos puntos de datos con puntos de datos conocidos, o en su lugar, detectar patrones en los datos de entrenamiento y construir un modelo predictivo.

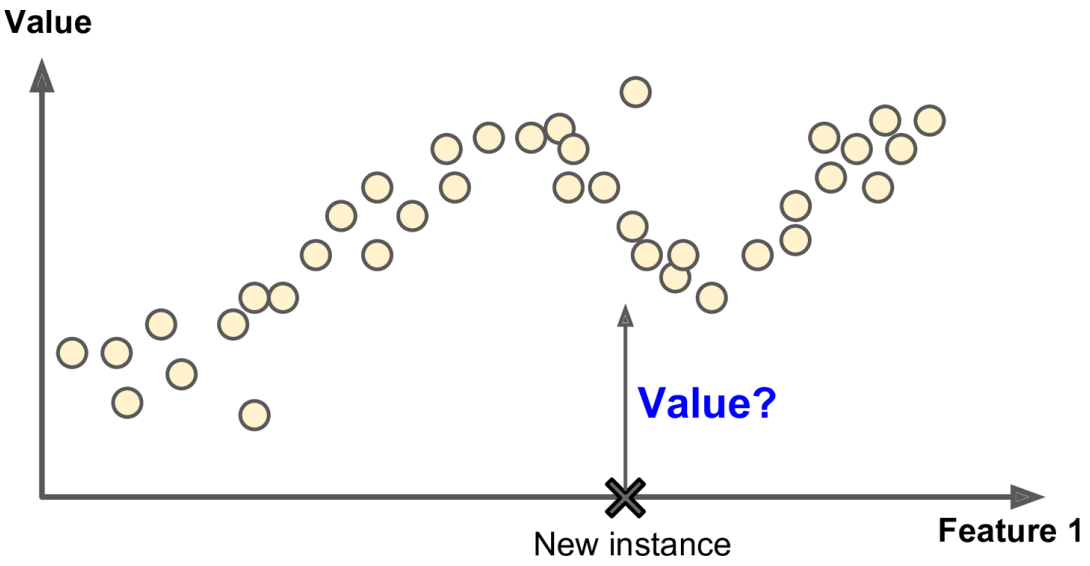
3.3.1 Aprendizaje Supervisado En el aprendizaje supervisado, los datos de entrenamiento que alimenta al algoritmo incluyen las soluciones deseadas, denominadas etiquetas. Una típica tarea de aprendizaje automático supervisado es la clasificación. El filtro de spam es un buen ejemplo de esto: está entrenando con muchos correos electrónicos de ejemplo junto con su clase (es spam, o no lo es), y debe aprender a clasificar los nuevos correos electrónicos.

Figura 7. Un conjunto de capacitación etiquetado para el aprendizaje supervisado (por ejemplo, clasificación de spam)



Otra tarea típica es predecir un valor numérico objetivo, como el precio de un automóvil, dado un conjunto de características (kilometraje, antigüedad, marca, etc.) llamadas predictores. Este tipo de tarea es llamada regresión. Para entrenar el sistema, debe darle muchos ejemplos de automóviles, incluidos sus predictores y sus etiquetas (es decir, sus precios).

Figura 8. Regresión



3.3.1.1 Overfitting y Underfitting El *Overfitting* o sobreajuste de datos, se refiere a que el modelo funciona bien en los datos de entrenamiento, pero no se generaliza bien. El *Overfitting* ocurre cuando el modelo es demasiado complejo en relación con la cantidad de datos de entrenamiento, generando un modelo ruidoso.

El modelo puede detectar patrones en el ruido en sí, los cuales no se van a instanciar de nuevo, pero al momento de predecir usando nuevos datos, el modelo fallará en la predicción.

Las posibles soluciones al *Overfitting* son:

- Simplificar el modelo seleccionando uno con menos parámetros (por ejemplo, un modelo lineal en lugar de un modelo polinómico de alto grado), reduciendo el número de atributos en los datos de capacitación o restringiendo el modelo.
- Reunir más datos de entrenamiento reducen el ruido en los datos de entrenamiento (por ejemplo, corregir errores de datos y eliminar valores atípicos).

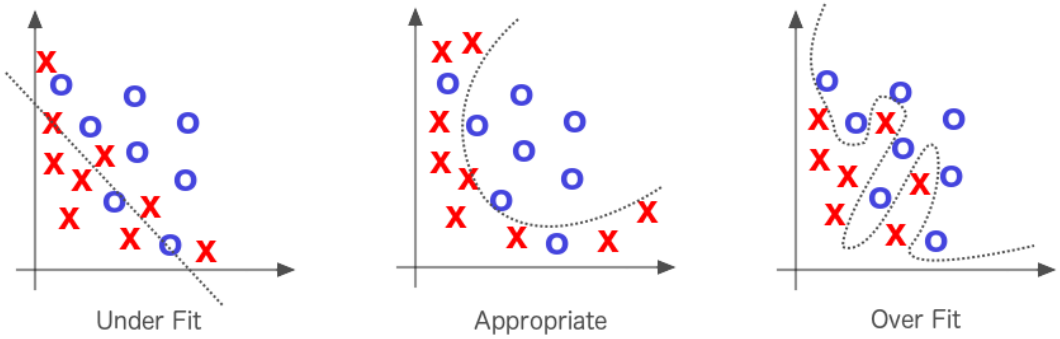
Por el contrario, el *Underfitting* o el ajuste insuficiente de datos ocurre cuando el modelo es demasiado simple para conocer la estructura subyacente de los datos.

Por ejemplo, un modelo lineal con satisfacción con la vida es propenso a ser inadecuado; la realidad es simplemente más compleja que el modelo, por lo que sus predicciones serán inexactas, incluso en los ejemplos de entrenamiento. Las principales opciones para solucionar este problema son:

- Seleccionar un modelo más poderoso, con más parámetros.
- Alimentando mejores características al algoritmo de aprendizaje.
- Reducir las restricciones en el modelo (por ejemplo, reducir el hiperparámetro de regularización.)

En la siguiente figura parte (a), se observa que una línea recta no es la adecuada para dividir las dos clases, generando *Underfitting* por causa del poco entrenamiento que tuvo el modelo, perdiendo precisión. Por el contrario, en la parte (c), se observa que la frontera se ajusta para todos los puntos del conjunto de entrenamiento y que es bastante ondulada, algo poco probable, se establece así una clasificación que no va conforme a un posible patrón o naturaleza de los datos. Y finalmente, en la parte (b), se observa el modelo apropiado, ya que la frontera de clasificación se asemeja a una función polinómica, gracias a que se tienen suficientes datos de entrenamiento y datos de testeo, logrando optimizar el rendimiento del modelo.

Figura 9. Ajuste de la complejidad de un modelo



Fuente: SAFARI BOOKS ON LINE [en línea] disponible en: https://www.safaribooksonline.com/library/view/deep-learning/9781491924570/ch01.html

3.3.1.2 Hiperparámetros Los hiperparámetros son parámetros de un algoritmo de aprendizaje (no del modelo). Como tal, no se ve afectado por el algoritmo de aprendizaje en sí mismo; debe establecerse antes del entrenamiento y permanecer constante durante el entrenamiento. Ajustar los hiperparámetros es una parte importante de la construcción de un sistema de aprendizaje automático.

La optimización o ajuste de hiperparámetros es el problema de elegir un conjunto de hiperparámetros óptimos para un algoritmo de aprendizaje. El mismo tipo de modelo de aprendizaje automático puede requerir diferentes restricciones, ponderaciones o tasas de aprendizaje para generalizar diferentes patrones de datos. La optimización de hiperparámetros encuentra una tupla de hiperparámetros que produce un modelo óptimo que minimiza una función de pérdida predefinida en datos independientes dados. La función objetivo toma una tupla de hiperparámetros y devuelve la pérdida asociada. La importancia de la optimización, es que el tiempo requerido para entrenar y probar un modelo puede depender de la elección de sus hiperparámetros[4].

[4] CSAIL Random Search for Hyper-Parameter Optimization [en línea]. <http://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf> [citado en 6 de enero del 2018]

3.3.2 Algoritmos de Predicción Los algoritmos de predicción usados en el trabajo se explicarán a continuación, mostrando su lógica, y planteamiento matemático.

3.3.2.1 Linear Regression En español denominado como regresión lineal, es un modelo lineal que hace una predicción simplemente calculando una suma ponderada de las características de entrada, más una constante llamada término de sesgo (también llamado término de intercepción).

La ecuación de la predicción del modelo de *Linear Regression* se muestra a continuación:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

- $\hat{y}$ es el valor predicho.
- n es la cantidad de características.
- x_i es el i-ésimo valor de la característica.
- θ_j es el j-ésimo parámetro del modelo.

Esto se puede escribir de una manera mucho más concisa usando una forma vectorizada, como se muestra a continuación:

$$\hat{y} = h_\theta(\mathbf{x}) = \theta^T \cdot \mathbf{x}$$

- θ es el vector de parámetros del modelo.
- θ^T es la transpuesta de θ.
- $\mathbf{x}$ es el vector característica de la instancia.
- $\theta^T \cdot \mathbf{x}$ es el producto escalar de θ^T y $\mathbf{x}$.
- h_θ es la función de hipótesis, usando los parámetros del modelo θ.

Para entrenar un modelo se debe establecer una medida de rendimiento que se ajuste a los datos de entrenamiento. La medida de rendimiento más común de un modelo de regresión es el Error Cuadrático Medio (*RSME*, por sus siglas en inglés)

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

- *m* es el número de instancias en el conjunto de datos en el que se está midiendo el RMSE.
- $\mathbf{x}^{(i)}$ es un vector de todos los valores de las características de la i-ésima instancia en el conjunto de datos.
- $y^{(i)}$ es su etiqueta (el valor de salida deseado para esa instancia).
- *X* es una matriz que contiene todos los valores de característica de todas las instancias en el conjunto de datos.
- *h* es la función de predicción de su sistema, también llamada hipótesis.
- *RMSE* (X, *h*) es la función costo medida en el conjunto de ejemplos que utiliza la hipótesis h.

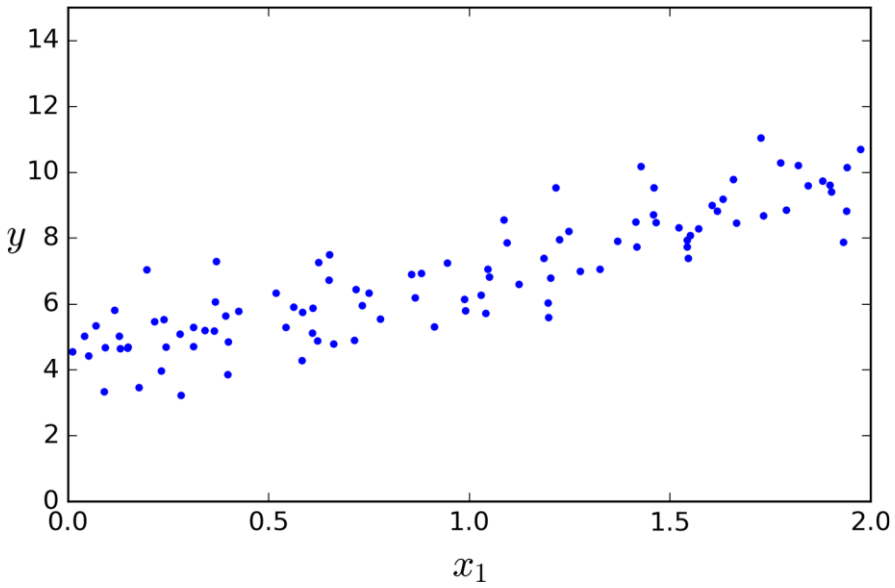
Por lo tanto, para entrenar un modelo de *Linear Regression*, se necesita encontrar el valor de θ que minimice RMSE. Para encontrar el valor de θ que minimiza la función costo, hay una solución cerrada, es decir, una ecuación matemática que da el resultado directamente. Esta se conoce como ecuación normal.

$$\hat{\theta} = \left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

- $\hat{\theta}$ es el valor de θ que minimiza la función de costo.
- y es el vector objetivo que contienen $y^{(1)}$ a $y^{(m)}$

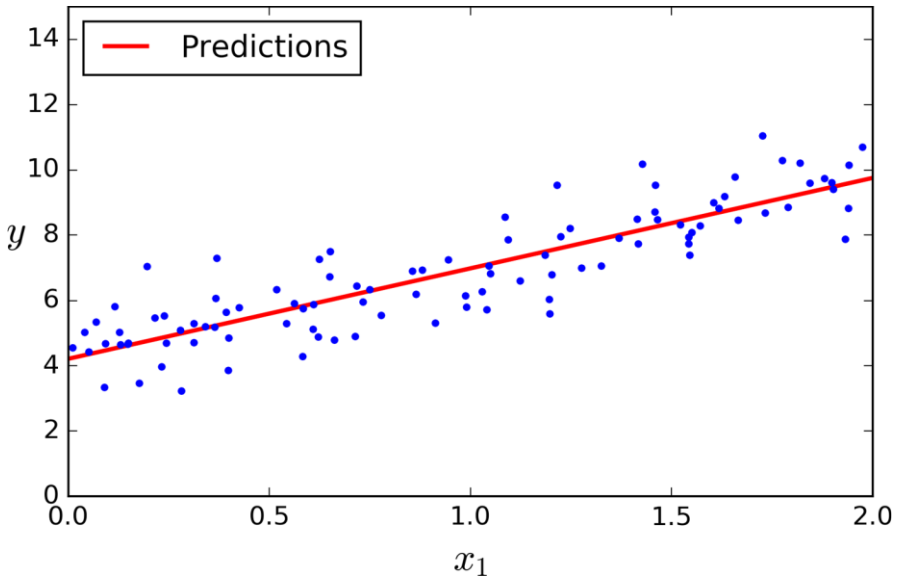
La ecuación normal es lineal con respecto al número de instancias es el conjunto de entrenamiento, por lo que maneja conjuntos de entrenamiento grandes de manera eficiente. Una vez que se haya entrenado el modelo de *Linear Regression* (usando la ecuación normal o cualquier otro algoritmo), las predicciones son muy rápidas: la complejidad computacional es lineal con respecto al número de instancias en las que desea hacer predicciones y el número de características[5].

Figura 10. **Conjunto de datos lineales generados aleatoriamente**



[5] GERÓN, Aurélien. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly, 2017

Figura 11. *Linear Regression* aplicada a un conjunto de datos

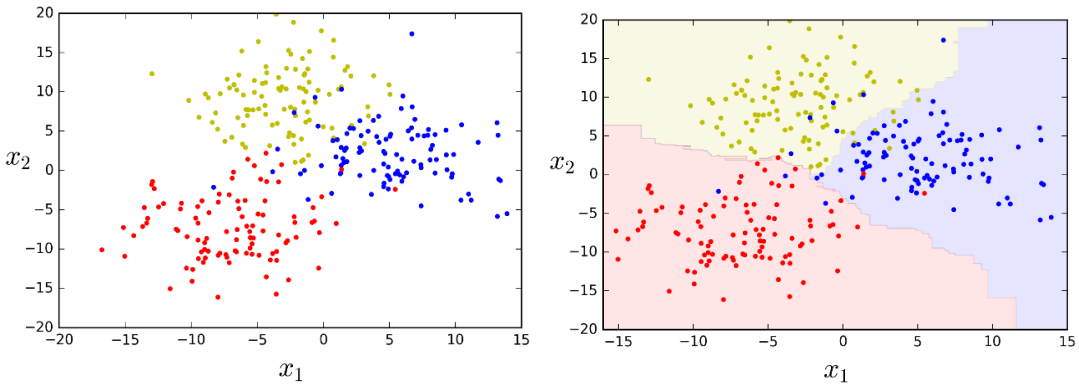


3.3.2.2 Random Forest *Random Forest* es un algoritmo para clasificación y regresión que presenta un excelente rendimiento para datos de alta dimensionalidad.

Inicialmente se crea de manera aleatoria un conjunto de datos de entrenamiento de igual tamaño que el conjunto de datos original. La aleatoriedad asegura que habrá datos que no harán parte del conjunto de entrenamiento, éstos forman un nuevo conjunto de datos de validación denominado '*out of bag data*' (*OOB data*). Los datos OBB tienen como fin determinar la impureza en cada una de las divisiones del árbol, y la suma de estas impurezas permite determinar la impureza del árbol.

La importancia de la división de los datos de entrenamiento, obliga a encontrar la mejor variable a partir de subconjuntos creados de forma aleatoria. Este proceso repetitivo genera al final, un conjunto de árboles entrenados sobre diferentes conjuntos de datos y de atributos. Logrando así, un algoritmo capaz de evaluar cada nueva entrada, clasificando el dato dependiendo del voto mayoritario del conjunto de árboles.

Figura 12. **Conjunto de datos antes y después de entrenar con *Random Forest***



Fuente: LUIS VALE SILVA [en línea] disponible en: *http://luisvalesilva.com/datasimple/random_forests.html*

Para medir el error del algoritmo, es recomendable utilizar un método denominado '*out of the bag error'*, el cual consiste en que para cada árbol se utiliza el conjunto de datos que no son clasificados en el árbol usando el algoritmo de entrenamiento. Y promediando sobre el conjunto de árboles, se puede estimar el error del algoritmo.

El número de árboles puede incrementar, permitiendo una mayor diversidad de los mismos, y a su vez reduciendo el error OBB. Cuando está creciendo un árbol nuevo, al usar variables aleatorias para cada característica en lugar de usar las mejores variables posibles, permite mayor aleatoriedad en los árboles, además, que los árboles adicionales sean mucho más rápidos de entrenar, debido a que buscar la mejor variable para cada característica en cada nodo es una de las tareas que más tiempo consume para crear un árbol.

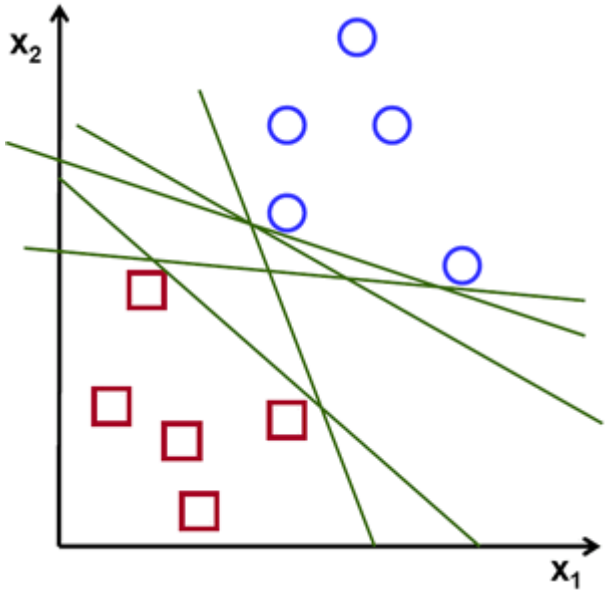
Otra ventaja, es que el método *Random Forest*, permite la paralelización, gracias a que cada árbol se puede construir de manera independiente a los otros árboles. Esta ventaja lo hace altamente recomendable para entornos paralelos[6].

[6] SPRINGER Random Forest [en línea]. <https://link.springer.com/content/pdf/10.1023%2FA%3A1010933404324.pdf> [citado en 7 de enero del 2018]

3.3.2.3 Support Vector Machine Por sus siglas en inglés, *SVM*, *Support Vector Machine*. Es un algoritmo de aprendizaje automático muy potente y versátil, capaz de realizar tanto clasificaciones lineales o no lineales como regresiones. Este algoritmo es particularmente adecuado para la clasificación de conjuntos de datos complejos pequeños o medianos. Los *SVM* usan modelos lineales para representar una frontera de decisión no lineal en el espacio origen, construyendo un modelo clasificador binario no probabilístico capaz de clasificar nuevas muestras de datos.

Para clasificar la muestra de datos, se determina un hiperplano que maximice la distancia con el espacio del conjunto de datos. Suponiendo que se tiene un conjunto de datos separable en dos clases, si se intenta clasificar mediante múltiples líneas, la solución resultará ser mala, porque la línea pasa demasiado cerca de los puntos, siendo sensible al ruido y no generalizará correctamente.

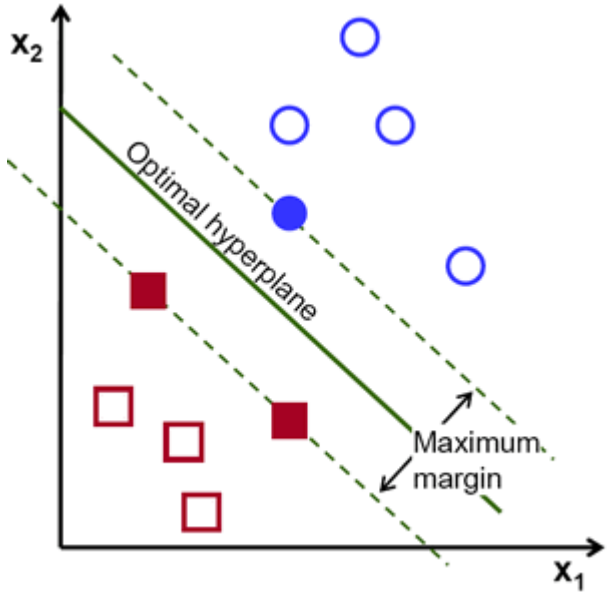
Figura 13. **Posibles hiperplanos que separan dos clases de datos**



Al aplicar el algoritmo *SVM*, éste calcula el hiperplano que da la distancia mínima más grande al conjunto de entrenamiento. En la teoría de *SVM* esta distancia recibe

el nombre de margen. Por lo tanto, el hiperplano de separación óptimo maximiza el margen de los datos del conjunto de entrenamiento.

Figura 14. **Hiperplano con margen máximo al conjunto de datos**



El algoritmo *SVM*, hace uso de funciones kernel, las cuales permiten optimizar el costo computacional de los datos, sin afectarlos de manera directa.

Cada función kernel se expresa de la siguiente manera:

$$K(x,x') = < \varphi(x), \varphi(x') >$$

- x, x' son datos de entrada representados en vectores.
- φ es una transformación que se realiza en los datos.
- $\langle x, x' \rangle$ es el producto punto entre la matriz x y su transpuesta.

La función kernel, puede variar dependiendo del comportamiento y conjunto de datos que se vayan a entrenar.

El kernel lineal representa un producto punto el cual se suma a una constante opcional c. El kernel lineal se debe usar cuando la cantidad de características es mayor que la cantidad de observaciones. Este kernel se expresa de la siguiente manera:

$$K(x,x') = \gamma + c$$

Por otra parte, el kernel polinomial no solo toma en cuenta las características dadas de la muestra, sino también las combinaciones de éstas. El kernel polinomial se representa en términos de monomios. Este kernel se expresa de la siguiente manera:

$$K(x,x') = (\gamma * + r)^d$$

Donde d es el grado las variables polinomiales.

Y finalmente, el kernel gaussiano se usa cuando la cantidad de observaciones es mayor que la cantidad de características. El kernel gaussiano, garantiza un predictor globalmente óptimo que minimiza los errores de estimación y aproximación de un clasificador. Éste kernel es altamente recomendable para modelos infinitamente complejos, ya que es el kernel más flexible. El kernel Gaussiano se representa de la siguiente manera:

$$K(x,x') = e^{-\gamma|x-x'|^2}$$

Donde γ, es una variable llamada gamma que debe ser mayor que cero, y define el alcance de la influencia de un único ejemplo de entrenamiento. Con valores pequeños, que representan un largo alcance, y valores grandes que representan un corto alcance[7].

[7] SCIKIT-LEARN Support Vector Machines. 1.4 Support Vector Machines – Scikit-Learn 0.19.1 Documentatio [en línea]. <http://scikit-learn.org/stable/modules/svm.html> [citado en 7 de enero del 2018]

3.3.3 Series Temporales Se define como una secuencia de datos experimentales ordenados en el tiempo, que se diferencia de otros conjuntos de datos debido a que el ordenamiento temporal crea una correlación entre valores sucesivos. Es decir, el valor presente puede depender de los valores pasados.

Las series temporales son altamente recomendables en el estudio de un fenómeno, ya que se puede conocer las causas subyacentes en su comportamiento, se puede comparar con otros fenómenos, y se puede predecir su comportamiento futuro[8].

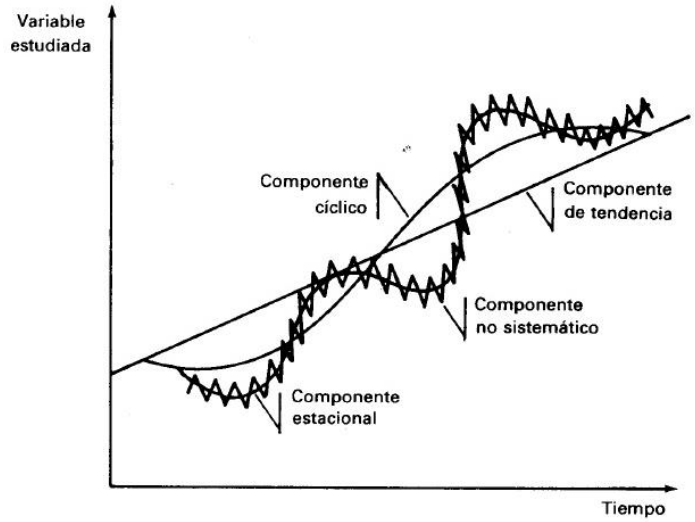
Las series temporales se expresan de la siguiente manera:

$$X_t = T_t + S_t + C_t + R_t$$

- T_t es un parámetro denominado tendencia, y representa el incremento o decremento a largo plazo en los datos.
- S_t es denominado estacionalidad, y es un patrón repetitivo con un periodo definido.
- C_t es denominado ciclicidad, y representa los cambios cíclicos sin un periodo definido.
- R_t son las fluctuaciones irregulares e impredecibles, y se denomina residuo.

[8] ESCUELA VERANO Análisis de Series de Tiempo [en línea]. <http://www.escuela-verano.otrasenda.org/wp-content/uploads/2015/06/curso_series.pdf> [citado en 15 de enero del 2018]

Figura 15. **Comportamiento de los componentes de una serie de tiempo**



Fuente: PROYECTOS INGENOTAS Modelos de series de tiempo [en línea] disponible en: http://proyectos.ingenotas.com/2012/07/modelos-de-series-de-tiempo-ii.html

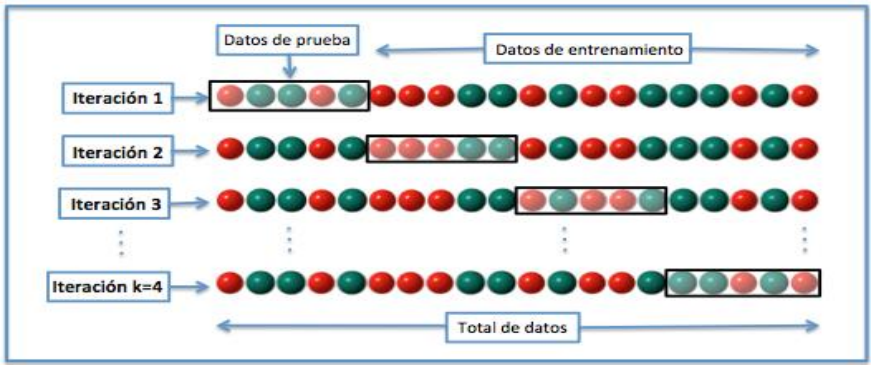
3.3.4 Validación de Modelos Al finalizar de construir un modelo, es altamente recomendable realizar medidas de evaluación, y así poder determinar la exactitud de su predicción. Existen diversas métricas que pueden utilizarse para medir el rendimiento, las cuáles deben aplicarse a la totalidad del conjunto de datos, es decir, no solamente a los datos que se entrenaron, sino también a los datos que se excluyeron en la fase de entrenamiento. La importancia de validar un modelo, es que permite optimizarlo mediante una disminución de ruido y un aumento de precisión; lo que lo hace más eficiente en la predicción de nuevas observaciones.

3.3.4.1 Cross Validation En español denominada validación cruzada, es una técnica de evaluación de modelos, que optimiza los parámetros del modelo garantizando un ajuste independiente de los datos de entrenamiento y de prueba. Funciona mediante repeticiones, en las cuales se dividen los datos de entrenamiento para simular la predicción fuera del conjunto de datos. Por lo tanto,

se realiza un ajuste a un hipotético conjunto de datos de prueba, debido a que no se dispone realmente de dichos datos.

El enfoque más común de la validación cruzada, divide el conjunto de datos de entrenamiento en k conjuntos más pequeños, llamados *k-folds*. El modelo toma como conjunto de datos *k-1* y como modelo de prueba el *fold* restante para medir la calidad de las predicciones en el conjunto de datos. Dependiendo del valor de *k,* se harán iteraciones, hasta que cada *fold* haya sido medido como conjunto de pruebas. Este enfoque de la validación cruzada da como resultado, los promedios y las desviaciones estándar de cada uno de los conjuntos de entrenamiento y de prueba que fue generado a través de cada iteración[9].

Figura 16. **Validación cruzada usando el enfoque *k-fold***



Fuente: WIKIMEDIA [en línea] disponible en: https://commons.wikimedia.org/wiki/File:K-fold_cross_validation.jpg

[9] CROSS-VALIDATION: Evaluating Estimator Perfomance. 3.1. Cross-Validation: Evaluating Estimator Perfomance – Scikit-Learn 0.19.1 Documentation. [en línea]. <http://scikit-learn.org/stable/modules/cross_validation.html> [citado en 19 de enero del 2018]

3.4 TRATAMIENTO DE DATOS

Durante el transcurso de análisis y modelado de datos, se dedica una gran cantidad de tiempo al tratamiento de datos: carga, limpieza, transformación y reorganización. Con frecuencia, la forma en que los datos se almacenan en archivos o bases de datos no está en el formato correcto para una tarea en particular. Este proceso de tratamiento de datos es de vital importancia, para lograr una manipulación correcta de los datos, debido a que los resultados que se obtengan al final, dependerán en gran parte al tipo de tratamiento que se les den a los datos.

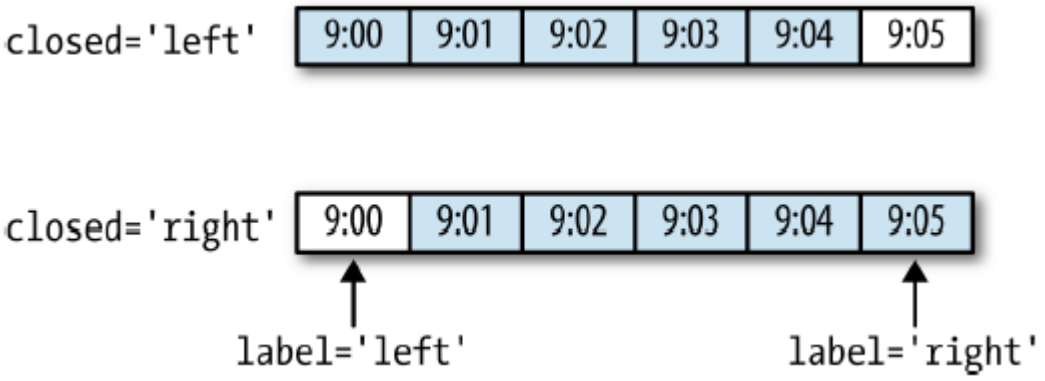
3.4.1 Resampling El *resampling* o remuestreo en español, se refiere al proceso de conversión de una serie temporal de una frecuencia a otra. Es altamente recomendable para procesar series de tiempo muy grandes. La agregación de datos de una frecuencia más alta a frecuencias más bajas se conoce como *downsampling* o disminución de muestreo; mientras que convertir frecuencias más bajas a frecuencias más altas se conoce como *upsampling* o aumento de muestreo.

El *downsampling* es una tarea bastante común de la serie temporal. En la que se agregan nuevos datos que no necesitan ser arreglados frecuentemente, ya que la frecuencia deseada es capaz de definir las divisiones de las series temporales, desde donde empieza cada división, hasta donde termina. Cada intervalo está medio abierto; donde una cantidad determinada de datos puede permanecer a un intervalo, y la unión de los intervalos debe conformar el marco de tiempo completo.

En el *downsampling* es importante definir cuál lado de cada intervalo estará cerrado, para esto se asume un valor definido como '*closed'*, haciendo que el intervalo se cierre en el lado derecho o en el lado izquierdo. De igual manera, es importante etiquetar el intervalo al momento de cerrarlo, por lo que se hace uso de un argumento denominado '*label'*. La importancia de esto, es que se tenga en cuenta exactamente la manera en que se segmentan los datos.

Contrariamente, el *upsampling* al ser más matizado, no necesita agregación. El *upsampling* define en qué parte del intervalo de tiempo coloca los valores antes de hacer el *resampling*.

Figura 17. **Intervalos de tiempo que se remuestrean a una frecuencia de cinco minutos**



3.4.2 Pandas. Pandas es una librería de código abierto de alto rendimiento, para el análisis de datos en el lenguaje de programación Python. Fue desarrollada en el año 2008, y desde entonces se ha convertido en la librería estándar para el análisis de datos estadísticos. Pandas puede procesar una variedad de conjuntos de datos en diferentes formatos, facilitando la carga/ importación de datos desde diversas fuentes como *CSV* y *DB/SQL* (bases de datos).

La característica distintiva de Pandas es la representación sencilla y adecuada de los datos, ya que hacer un análisis de datos equivalente en otros lenguajes de programación como Java o C++ requeriría de muchas más líneas de código[10].

3.4.3 CSV. El formato denominado *CSV* (*Comma Separated Values*), en español definido como valores separados por comas, es el formato de importación y

[10] ANTHONY, Femi. Mastering Pandas: Master the Features and Capabilities of Pandas, a Data Analysis Toolkit for Python. Pack Publishing Ltd., 2015

exportación más común para hojas de cálculo y bases de datos. No existe un estándar, por lo que el formato se define operativamente por las numerosas aplicaciones que lo leen y lo escriben. La falta de un estándar significa que a menudo existen diferencias sutiles en los datos producidos y consumidos por diferentes aplicaciones. Estas diferencias pueden hacer que sea molesto procesar archivos *CSV* de múltiples fuentes. Sim embargo, aunque los delimitadores y los caracteres de comillas varían, el formato general es lo suficientemente similar como para que sea posible escribir un único módulo que pueda manipular eficazmente dichos datos, ocultando los detalles de lectura y escritura de los datos del programador. *CSV* implementa clases para leer y escribir datos tabulares, seleccionando secuencias de lectura y escritura. Además, permite a los programadores describir los formatos *CSV* entendidos por otras aplicaciones o definir sus propios formatos *CSV*[11].

[11] CSV - CSV FILE READING AND WRITING. 13.1. Csv – CSV File Reading and Writing – Python 2.7.14 Documentation [en línea]. <https://docs.python.org/2/library/csv.html> [citado en 16 de enero del 2018]

4. ESTADO DEL ARTE

Tradicionalmente, el estudio de la temperatura atmosférica se ha venido realizando debido a su gran relevancia en muchas áreas de investigación. Siendo la predicción de ésta uno de los objetivos principales de estudio, se exponen a continuación proyectos tecnológicos que han intentado cumplir con este objetivo.

Un modelo de red neuronal artificial (*ANN*), se desarrolló para predecir temperaturas ambientales diarias en Denizli, al sudoeste de Turquía. Con el fin de capacitar al modelo, se utilizaron valores de temperatura medidos por el Servicio Meteorológico Estatal de Turquía durante tres años (2002-2005) los cuales se utilizaron como datos de entrenamiento, mientras que los registros de temperatura del año 2006, se tomaron como datos de prueba. Para determinar la arquitectura de red óptima, se diseñaron varias arquitecturas de red, también se usaron diferentes algoritmos de entrenamiento, y fue necesario variar el número de neuronas y capas a utilizar. Al final, los resultados demostraron que el enfoque *ANN* es un modelo confiable para la predicción de la temperatura ambiente[12].

En Vietnam se desarrolló un modelo de predicción estacional, el cual es crucial para la planificación socio-económica, así como para la prevención de desastres naturales. Las predicciones estacionales se pueden realizar por métodos estadísticos y dinámicos, y permiten predecir las evoluciones del sistema climático con varios meses de anticipación. El simulador *RegCM4.2 (Regional Climate Model*

[12] DAILY MEANS AMBIENT TEMPERATURA PREDICTION Using Artificial Neural Network Method: A Case Study of Turkey. Renewable Energy, Pergamon, 6 de septiembre del 2008 [en línea]. <https://www.sciencedirect.com/science/article/pii/S0960148108002851> [citado en 25 de enero del 2018]

Versión 4.2), se encarga de realizar las predicciones respectivas, las cuales se comparan con el *Sistema de pronóstico Climático (CFS,* por sus siglas en inglés)[13].

Para satisfacer la necesidad existente de un sistema que proporcione datos ambientales locales en tiempo real en los campos de cultivos rurales para la detección y el manejo de enfermedades fúngicas. Se diseñó un sistema de Internet de las cosas (*IoT*), que consiste en un dispositivo capaz de enviar datos ambientales en tiempo real al almacenamiento en la nube, y usando un algoritmo de aprendizaje automático se logran predecir las condiciones ambientales para la detección y prevención de hongos. Los datos ambientales almacenados sobre condiciones tales como la temperatura del aire, la humedad relativa del aire, la velocidad del viento y la caída de la lluvia son accedidos y procesados por una computadora remota que ejecuta un algoritmo de *Support Vector Machine (SVM),* prediciendo a corto plazo la temperatura del aíre, y la propagación de las enfermedades fúngicas perjudiciales a través del campo de cultivo local[14].

[13] VAN, TAN PHAN, et al. Seasonal Prediction of Surface Air Temperature across Vietnam Using the Regional Climate Model Version 4.2 (RegCM4.2).

[14] AN IOT ENVIRONMENTAL DATA COLLECTION System for Fungal Detection in Crop Fields – IEEE Conference Publication [en línea]. <http://ieeexplore.ieee.org/abstract/document/7946787/?reload=true> [citado en 25 de enero del 2018]

5. METODOLOGÍA

Este proyecto se desarrolló guiado por los principios de la metodología experimental ágil, logrando un completo análisis, desarrollo y utilización de un framework de aprendizaje automático capaz de predecir la temperatura superficial en el observatorio Pierre Auger, ubicado en la provincia de Mendonza, Argentina.

El proyecto es desarrollado en función de brindar una alternativa en la estimación de la temperatura, diferente a lo que la comunidad científica Pierre Auger ha venido realizando durante los últimos años. Los datos fueron proporcionados por el observatorio, a través del planetario de investigación Halley, perteneciente a la Universidad Industrial de Santander.

La metodología ágil permite tener una visión clara del avance de la solución, realizando una lista de acciones a realizar, las cuales se dividen en fases o etapas. Es importante aclarar que cada fase debe ser evaluada, y de ser necesario, corregida en la siguiente acción estipulada en el cronograma del plan.

5.1 PREPARACIÓN DE HERRAMIENTAS Y REVISIÓN DE MARCO DE REFERENCIA

En esta etapa se estudian conceptos fundamentales e intrínsecos referentes al tema de investigación, así como también se practica el ejercicio de buscar las herramientas más apropiadas para facilitar y agilizar el desarrollo del proyecto en cuestión.

- Aprendizaje acerca de Python y *Machine Learning*.
- Aprendizaje acerca del preprocesamiento de datos.

5.2 CARACTERIZACIÓN, CAPTURA Y LIMPIEZA DE DATOS

En esta etapa se trabaja con los datos que proporciona el observatorio Pierre Auger. En un principio los datos son limpiados y adecuados de manera que en las fases posteriores su manipulación sea más fácil.

- Especificación datasets.
- Captura desde fuentes de datos de Pierre Auger.
- Limpieza mediante filtros de los datos de Pierre Auger.

5.3 DEFINICIÓN DE TAREAS PREDICTIVAS

En este punto se definen las actividades predictivas en base a los datos que se procesan en la etapa anterior. En este punto también se define el preprocesado y la respectiva integración de los datos. Por último, en esta fase también se definen los algoritmos de aprendizaje automático que se utilizaran para abordar de forma apropiada el problema en cuestión.

- Definir algoritmos a utilizar.
- Definir el preprocesamiento de datos.

5.4 CREACIÓN Y ENTRENAMIENTO DE LOS MODELOS

En esta etapa del proyecto se implementan los algoritmos definidos anteriormente, e inmediatamente se entrenan los datos en cada algoritmo implementado.

- Implementación de los algoritmos.
- Entrenamiento de los datos.

5.5 VALIDACIÓN EXPERIMENTAL

En esta etapa se implementa el preprocesamiento de datos definido anteriormente, para poder corroborar si los resultados ofrecidos por el modelo concuerdan con los esperados, y así, finalmente lograr determinar si dicho resultado es apto para hacer predicciones con nuevos datos de entrada, en diferentes regiones del observatorio.

- Implementación del preprocesamiento de datos.
- Evaluar confiabilidad de los resultados.

5.6 TRAIN/TEST CRUZADOS

En la última etapa del proyecto, se implementan funciones las cuales posibilitan la experimentación, a partir de combinaciones de datos provenientes de 4 estaciones y otras que permiten dividir los datos como lo describe la estrategia de validación cruzada que se adoptó.

- Variación de las variables X y Y tanto para los datos de entrenamiento como para los datos de prueba.
- Reutilización y evaluación de datos en cualquier zona del observatorio.

6. DESARROLLO DEL PROYECTO

En este capítulo se describe por fases, el procedimiento llevado a cabo para desarrollar el framework de trabajo, el preprocesamiento y el análisis general de los datos para elegir una manera de abordar el problema, la implementación de funciones de graficación y demás componentes necesarios para emprender la búsqueda de los mejores modelos predictivos.

Después de tener una idea general sobre cuál era el objetivo, de cuál era el lenguaje de programación que se iba a utilizar, del entorno versátil en el cual se iba a facilitar la visualización de gráficas y dataframes, y de haber aprendido el fundamento teórico del flujo de trabajo de un proyecto de análisis de datos y predicción con *Machine Learning*, el primer paso fue el de la obtención de datos que se describe a continuación.

6.1 CONVERSIÓN Y PREPROCESAMIENTO DE DATOS

6.1.1 Obtención y Conversión de Datos Como cualquier proyecto de *Machine Learning* con el enfoque supervisado, se solicitó al observatorio la pareja de datos correspondientes a los datos de entrada (*input data*) y los etiquetados (*supervisory signal*). Los primeros fueron todos los registros capturados por los distintos sensores de los 1660 detectores en ficheros DAT, los cuales estaban formateados de modo que cada registro estuviera separado por espacios. Para la obtención de dichos ficheros fue necesario llevar a cabo la conversión de ficheros root mediante un script en el lenguaje de programación C, que fue redactado por personal del observatorio.

Teniendo en cuenta que el proceso de conversión anteriormente descrito es tardío y exhaustivo en cuanto a consumo de recursos, la tarea se realizó en un servidor del grupo de investigación planetario Halley quienes hacen parte del proceso colaborativo con Pierre Auger. El procedimiento tardó más de 1 mes para arrojar como resultado 6 meses de registros de los distintos detectores disponibles.

Para los datos etiquetados, se obtuvieron del sitio web AUGER - UIS en el cual se publica de forma periódica los registros obtenidos por las distintas estaciones meteorológicas.

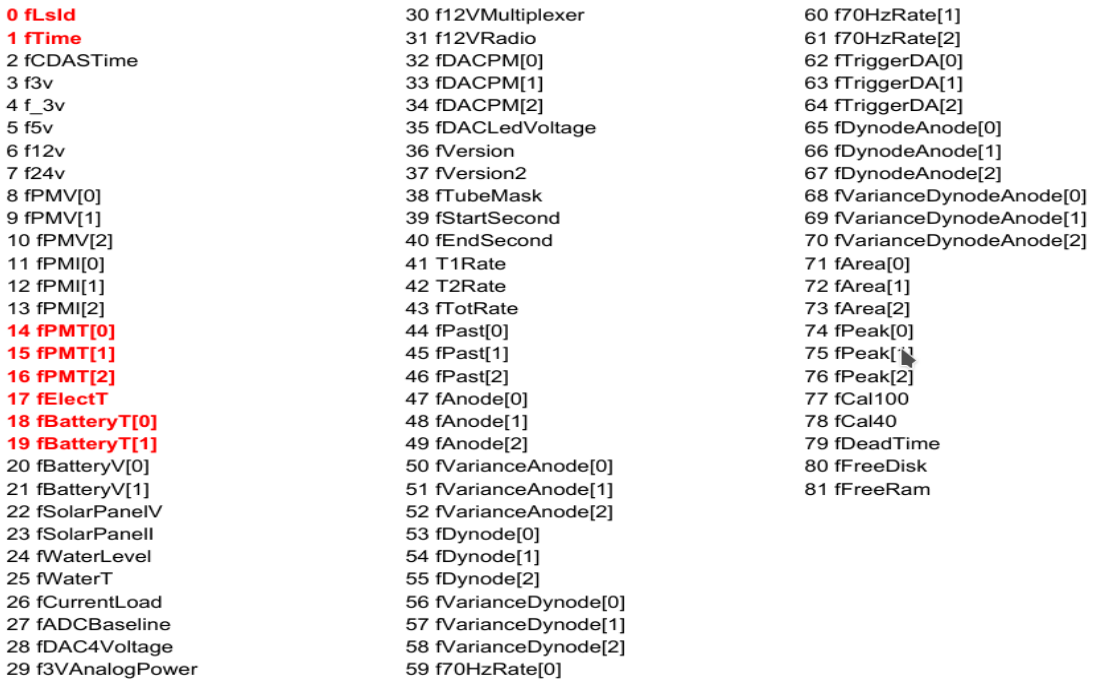
Una vez obtenidos los datos, de haber revisado la estructura de un fichero de los datos de los detectores y de una de las estaciones meteorológicas, el siguiente paso fue el de extraer la información que se consideraría necesaria por las dos siguientes razones: Primeramente, se descartarían señales las cuales introducen un sesgo en las predicciones y segundo, se reduciría el consumo de recursos debido a que la cantidad de datos a analizar es directamente proporcional al consumo de memoria RAM, un recurso limitado en el hardware que se dispuso.

6.1.2 Estructura de los Datos Antes de hablar acerca de la estrategia que se empleó para seleccionar los datos utilizados en el proyecto, es válido hacer una revisión general de los ficheros que fueron proporcionados.

Dichos ficheros estaban en formato de caracteres ASCII, separados por espacios y sin cabeceras. Una vez fueron convertidos a *CSV* (una manera más flexible de representación), un dataframe de detectores en condiciones ideales tendría que disponer de unas 1660 veces (la cantidad de detectores) 216 registros debido a que cada registro de cada detector, en teoría, debía registrar información a una frecuencia de 1 registro por cada 400 segundos.

En cuanto al número de columnas, es decir, la información proporcionada por los sensores de los detectores, se dispuso de 81 columnas de datos que se pueden ver a continuación en la figura 18.

Figura 18. **Columna de atributos de cada detector.**



Con respecto a los datos de las estaciones meteorológicas, estos también eran ASCII y también estaban separados por espacios, pero a diferencia de los detectores que estaban divididos por días, estos eran un solo fichero por cada estación.

Los registros de temperatura de cada estación estaban comprendidos desde su fecha de operación por vez primera en el observatorio, el cual, en la mayoría de los casos, fue desde el año 2002, hasta la fecha actual. Para el caso del fichero de Coihueco, el intervalo comprendido era más pequeño: desde el año 2006 hasta el año 2016.

Por otra parte, a diferencia de la frecuencia con la que se registraban los datos de los detectores, los de temperatura en las estaciones se guardaban a una frecuencia de 1 registro por cada 300 segundos.

Es importante mencionar que los datos de las columnas referentes al tiempo en ambos conjuntos de datos fueron representados en el formato estándar *Unix Timestamp*. También es válido hacer hincapié en que los ficheros en ambos casos no eran ideales debido a cortes de energía en los detectores o en las estaciones meteorológicas, por el mal funcionamiento de alguno de ellos o por cualquier otra razón.

6.1.3 Selección y Tratamiento de los Datos

6.1.3.1 Datos de los Detectores Cherenkov Gracias a la versatilidad de Pandas, una librería de Python para el tratamiento de datos en sus distintas estructuras, la tarea de seleccionar los datos después de haber sido copiados a la memoria RAM fue relativamente simple. Sólo fue cuestión de filtrar algunas columnas que, para nuestro juicio, guardaban relación con la señal etiquetada, es decir, la señal a predecir. En la figura 18 expuesta anteriormente, se denotan de color rojo los atributos elegidos.

Una vez elegidas las columnas con los datos que se requerían, se subdividieron los dataframes correspondientes de varios días, usando como criterio de selección el identificador de cada registro. Posteriormente, se agruparon estos datos en listas, de manera que cada elemento de la lista correspondía al conjunto de datos correspondientes a un detector y a un día en concreto.

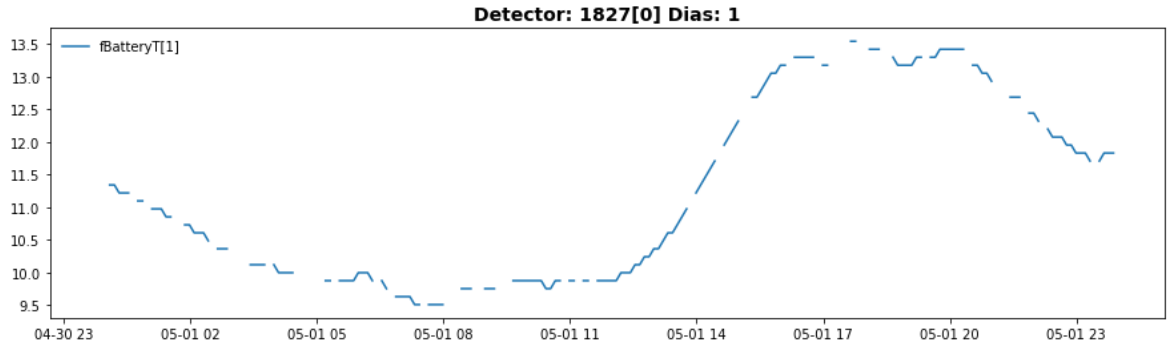
Por otra parte, en virtud de que los datos no eran ideales fue necesario emplear una función que completara los espacios nulos. Al abordar el problema como un modelo de series temporales, como primera opción se experimentó con las funciones

‘*resample’* (remuestreo) con el objetivo de estandarizar los registros a un mismo intervalo de tiempo y ‘*ffill*’, la cual reemplaza cualquier valor nulo por el registro inmediatamente anterior a la fila en la cual se situaba dicho valor nulo.

El problema con esta estrategia es que, al graficar los datos, estos no reflejaban las tendencias naturales de las curvas de las variables de los datos. Para solucionar dicho percance se optó por usar las funciones ‘*resample’* y posteriormente la de interpolación a sabiendas de que mejoraba los resultados. Esto, porque trazaban curvas más razonables, y temporalmente no comprometerían la integridad del problema de series temporales, gracias a que los datos nulos no serían numerosos.

A continuación, se expone como ejemplo el caso de una variable del detector identificado con el *ID* 1827, el cual se encuentra cerca a la estación Coihueco. Esta variable presenta valores nulos.

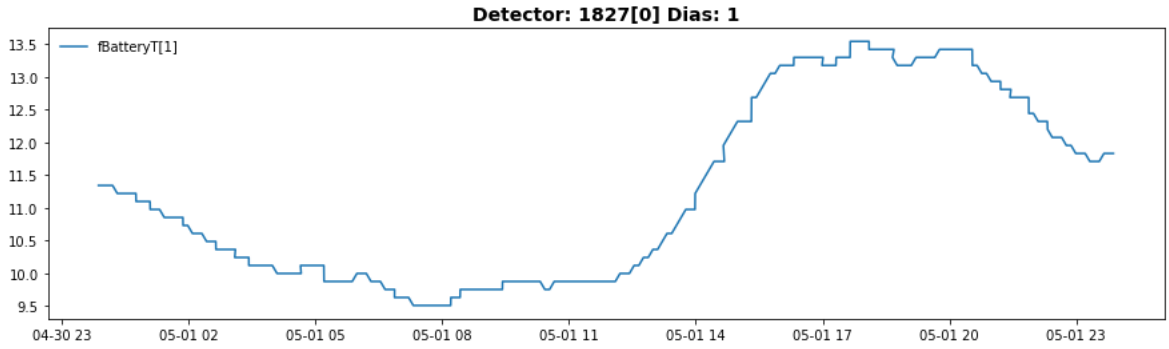
Figura 19. **Detector *ID1827*, variable *fBatteryT***



Fuente: AN IOT ENVIRONMENTAL DATA COLLECTION System for Fungal Detection in Crop Fields – IEEE Conference Publication [en línea]. <http://ieeexplore.ieee.org/abstract/document/7946787/?reload=true> [citado en 25 de enero del 2018]

Ahora bien, si se completan los datos usando la función de llenado hacia adelante ‘*ffill’*, los resultados mejoran, pero no lo suficiente para que dicho llenado sea considerado el más óptimo.

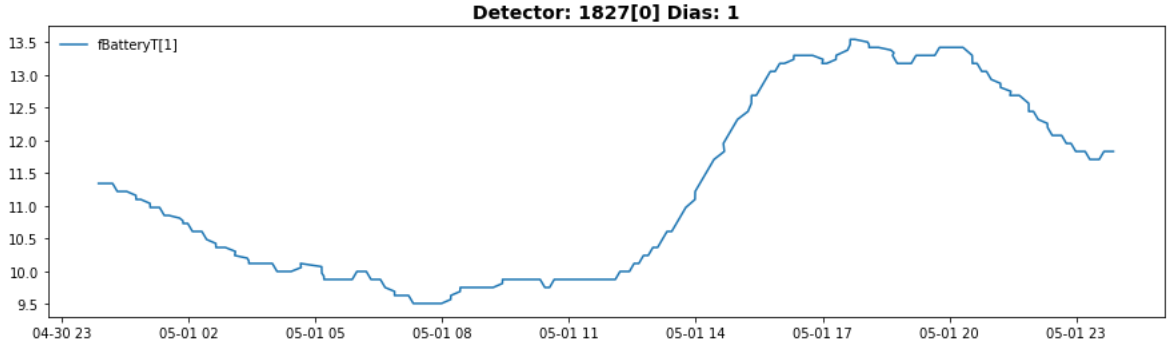
Figura 20. **Función '*ffill*' aplicada al Detector *ID1827*, variable *fBatteryT***



Fuente: AN IOT ENVIRONMENTAL DATA COLLECTION System for Fungal Detection in Crop Fields – IEEE Conference Publication [en línea]. <http://ieeexplore.ieee.org/abstract/document/7946787/?reload=true> [citado en 25 de enero del 2018]

Finalmente, si los valores nulos son completados interpolando, la gráfica se vuelve más razonable, siendo una solución mucho más óptima.

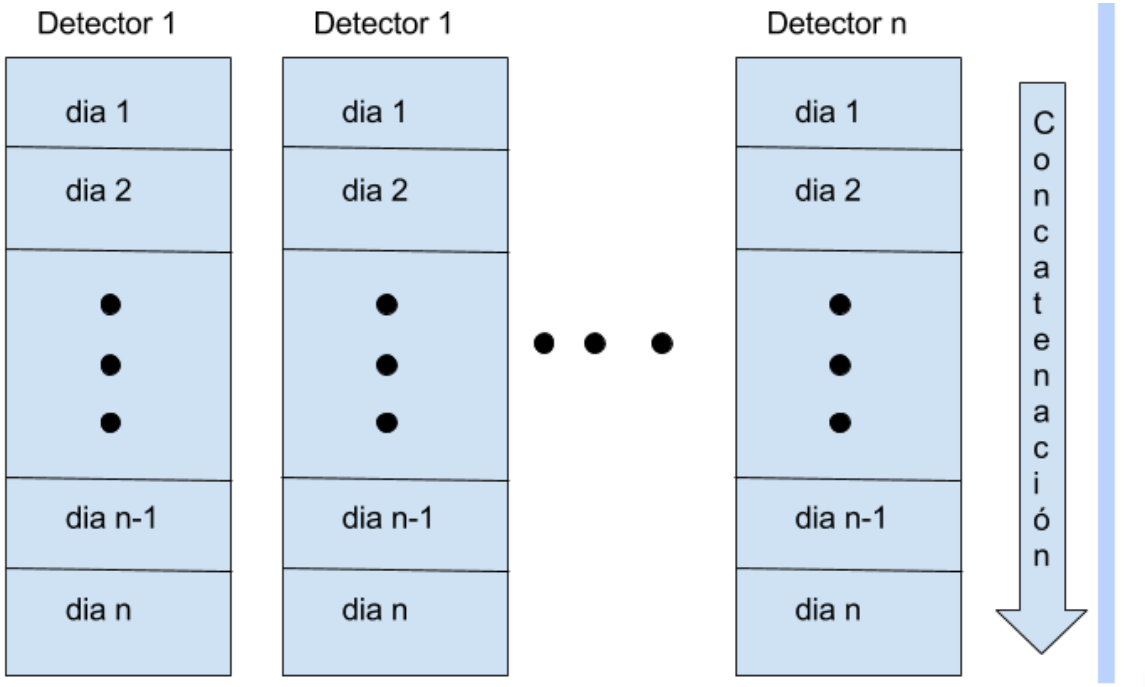
Figura 21. **Interpolación aplicada al Detector *ID1827*, variable *fBatteryT***



Fuente: AN IOT ENVIRONMENTAL DATA COLLECTION System for Fungal Detection in Crop Fields – IEEE Conference Publication [en línea]. <http://ieeexplore.ieee.org/abstract/document/7946787/?reload=true> [citado en 25 de enero del 2018]

En vista de que se tenían agrupados los registros de varios días con datos ya corregidos en listas, se procedió a la tarea de concatenación vertical (un dataframe encima de otro).

Figura 22. **Concatenación vertical de los dataframes**



Antes de la concatenación y del *resampling*, para los datos de los detectores cercanos a la estación Coihueco, fue necesario la realización de un paso extra. Éste consistió en duplicar los registros cuyos índices eran iguales. Esto en virtud de que al realizar la operación de *resampling*, Pandas encontraba los datos ambiguos y por tanto no podía llevar a cabo dicha operación.

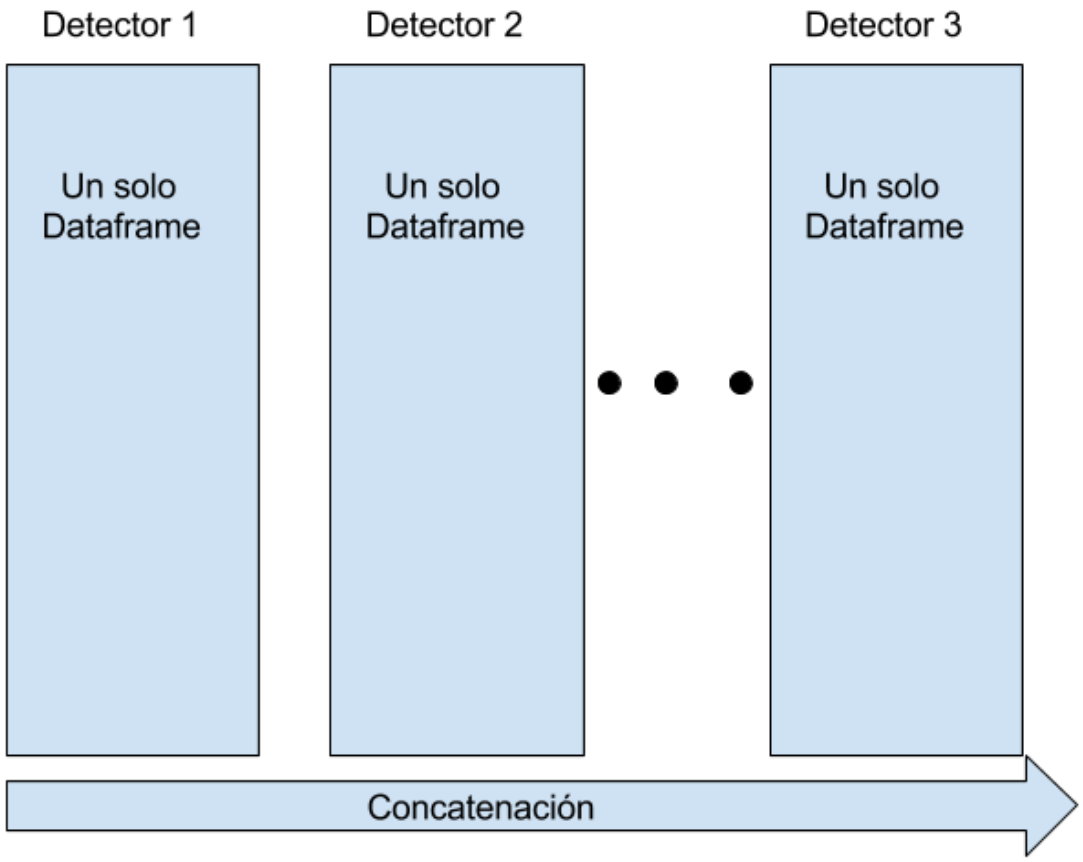
Para este procedimiento, se utilizó la función '*duplicated'*, que fue responsable de mostrar en diccionarios de booleanos, valores verdaderos en la primera incidencia de un registro con índice repetido. La operación restante, fue la de negar el ejercicio anteriormente descrito, debido a que lo que se necesitaba era extraer los registros no repetidos.

Ulteriormente, se implementó una función que contara el número de registros por cada detector, para garantizar que el preprocesado se había hecho exitosamente.

Para esto, se empleó una variable contadora y una estructura de control de repetición.

Finalmente, al disponer del arreglo de dataframes correspondientes a los detectores cercanos a una estación determinada, con índices concordantes, sin datos nulos y con la misma cantidad de columnas, solo faltó concatenar de manera horizontal con la misma función de concatenación vertical anteriormente utilizada, simplemente modificando el parámetro del eje.

Figura 23. **Concatenación horizontal de los dataframes**



El conjunto de operaciones anteriormente descritas, se hicieron con la finalidad de que todos los datos referentes a los detectores cercanos a una determinada

estación se consolidaran en un solo dataframe, y de esta manera dar vía libre al siguiente paso, el cual consistiría en tratar los datos de las estaciones meteorológicas para que más adelante estos dataframes con el atributo de la temperatura, pudieran ser concatenados al dataframe '*detectores'* que resultó de la concatenación horizontal anteriormente expuesta en la figura 19.

6.1.3.2 Datos de las Estaciones Meteorológicas Afortunadamente, los datos de las cuatro estaciones meteorológicas utilizadas estaban en mejores condiciones que la de los detectores Cherenkov. Exceptuando algunos pocos registros, que fueron corregidos de forma manual para evitar la inserción de sesgo, de modo que el único paso de preprocesado necesario fue el de *resample*.

6.1.3.3 Downsampling Como anteriormente se mencionó, la frecuencia de registros en los datos de las estaciones era mucho mayor que la de los detectores, por ello se llevó a cabo la tarea de hacer una disminución de muestreo (*downsampling*) en el conjunto de datos pertenecientes a las estaciones, con el fin de que los índices de ambos conjuntos de datos coincidieran y subsiguientemente estos se pudieran unificar en un solo dataframe.

Este ejercicio requirió de la función Pandas ya citada, '*resample*', con el parámetro de llenado hacia adelante '*ffil'* en su constructor. A continuación, se expone un ejemplo con registros separados por intervalos de 300 segundos.

Figura 24. **Registros de temperatura, dividida en intervalos de 300 segundos**

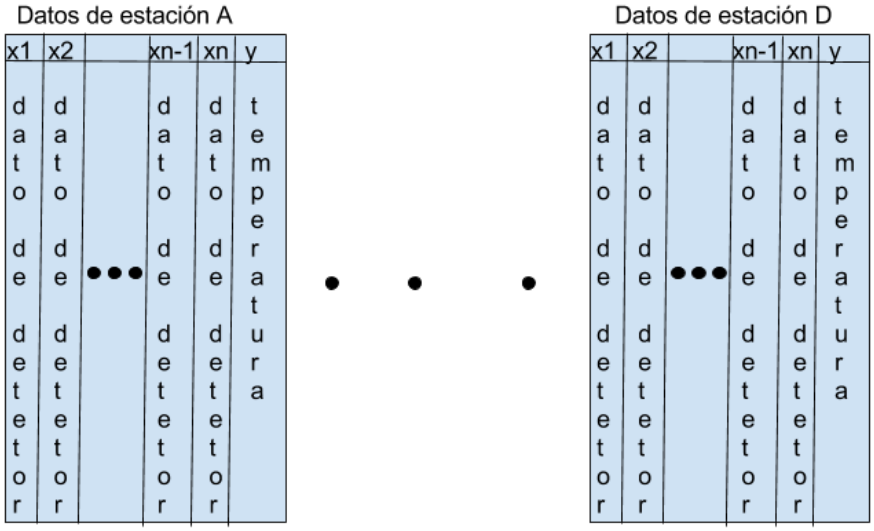
	Temperatura
2001-01-01 00:00:00	5
2001-01-01 00:05:00	20
2001-01-01 00:10:00	60
2001-01-01 00:15:00	150
2001-01-01 00:20:00	70
2001-01-01 00:25:00	10
2001-01-01 00:30:00	70
2001-01-01 00:35:00	50

Figura 25. **Registros de temperatura, después de aplicar *resample***

	Temperatura
2001-01-01 00:00:00	5
2001-01-01 00:06:40	20
2001-01-01 00:13:20	60
2001-01-01 00:20:00	70
2001-01-01 00:26:40	10
2001-01-01 00:33:20	70

Una vez con los datos de las estaciones ajustados a la misma frecuencia, se procedió a cortar el intervalo de datos de interés, es decir, la franja de tiempo que correspondería a la de los datos de los detectores. Para esto se requirió del uso de las funciones '*Loc'* y '*lloc'* de Pandas, las cuales son útiles para seleccionar, manipular o mostrar índices. Finalmente, con ambos conjuntos de datos estandarizados, corregidos y completados, el acto a seguir fue el de concatenar los datos de forma horizontal.

Figura 26. **Dataframe resultante al concatenar horizontalmente los registros de los detectores y de temperatura por cada estación**



6.2 ENTRENAMIENTO Y PREDICCIÓN

En el siguiente estado del proceso, se implementaron una serie de funciones con el fin de preparar los datos con los que se alimentarían los objetos estimadores, se aprestarían los datos según la estrategia de validación cruzada y se representarían los datos ya sea en tablas o en gráficas. Posteriormente, se optimizaron dichas funciones con el propósito de mejorar su flexibilidad a la hora de mover hiperparámetros de los objetos estimadores.

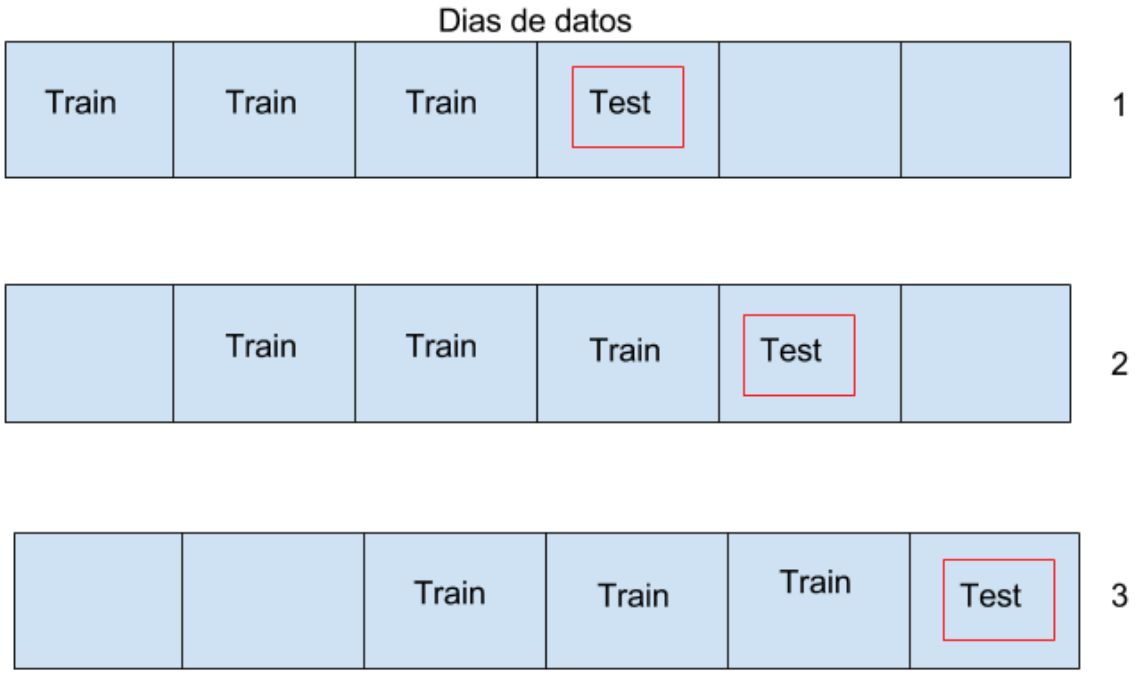
6.2.1 Estrategia de Validación Cruzada Como los datos de entrenamiento y la variable a predecir estaban representadas en el tiempo, la estrategia de validación cruzada no se podía trabajar como alguna otra tarea de *Machine Learning*, en las cuales se utiliza un porcentaje de ejemplos (*registros*) seleccionados mediante un proceso *poisson* (distribución probabilística discreta) para datos de entrenamiento y el resto de ejemplos como datos de prueba (*testing*). Este enfoque no es compatible con series temporales a causa de que no se puede plantear un modelo de predicción

del presente, usando señales provenientes del futuro. De manera más específica, no es lógico plantear un modelo para predecir la temperatura a las 12:00 pm del día de hoy en un detector cercano a la estación Los Leones, usando datos del día de mañana de los detectores cercanos a dicha estación.

Como solución, la táctica que se adoptó fue la de usar una cantidad determinada de días pasados (por ejemplo, los primeros 3 días de datos) como datos de entrenamiento y el día inmediatamente después al último día de entrenamiento (por ejemplo, el día 4), como datos de prueba. Este procedimiento se repetiría desplazando los datos de entrenamiento y los de prueba a razón de un día, hasta que el último día de prueba correspondiera al último día de los datos a analizar.

Por ejemplo, si se analizan 6 días de datos y el tamaño de los datos de entrenamiento fueran 3 días, la estrategia de validación cruzada sería como la que a continuación se muestra en la figura 27.

Figura 27. **Validación cruzada para 6 días de datos, 3 días de entrenamiento y 1 día de prueba**



Una vez definido esto, se prosiguió a entrenar y a graficar valores reales contra los de predicción usando como estimador *Linear Regression.* Se tomó esta inicialmente porque al prescindir de hiperparámetros para ajustar, se hacía mucho más fácil de implementar. Lo siguiente fue la selección de la métrica del cálculo de error de las predicciones.

6.2.2 Cálculo de Error Pese a que existen varias maneras de calcular el error entre dos conjuntos de datos, el principio que siguen cada uno de ellos es similar: se intenta encontrar la diferencia entre los elementos de cada conjunto de datos. Algunas formas de calcular el error implican elevar las diferencias a ciertas potencias, otras calculan las raíces, y algunas otras hacen alguno de los dos pasos anteriormente mencionados para finalmente promediar y obtener como resultado un escalar.

Después de sopesar las distintas métricas, se optó por el error absoluto medio (*MAE*) y está justificado por dos razones. La primera razón, porque al no elevar la diferencia a una potencia, el error calculado sería menos proclive a ser afectado por datos aislados (*outliers*). Y la segunda razón, porque al no involucrar raíces en sus operaciones, las unidades no se ven comprometidas de modo que los resultados son mucho más fáciles de interpretar.

$$\mathrm{MedAE}(y, \hat{y}) = \mathrm{median}(\mid y_1 - \hat{y}_1 \mid, \ldots, \mid y_n - \hat{y}_n \mid).$$

Con el framework optimizado, hacer el procedimiento anteriormente descrito para otros estimadores como *Random Forest* o *Support Vector Regression*, fue solamente necesario cambiar la línea de código en la que se instanciaba el objeto estimador[15].

6.2.3 Visualización Por la cantidad ingentes de valores que surgían de las predicciones y de los cálculos de errores, tener una idea de que tan bien era un modelo no era fácil de saber. Por ello, se recurrió a la librería de graficación Matplotlib, la cual es una herramienta de graficación bidimensional, diseñada para dibujar figuras en una variedad de formatos. Entre ellos graficación de curvas y diagrama de barras, los formatos que se usaron a lo largo del proyecto[16].

En principio, se utilizó para visualizar las curvas de distintas variables que disponía un detector con el fin de corroborar que el completado de datos se había hecho de forma exitosa. Luego se utilizó para graficar los datos reales de temperatura contra los datos predichos para tener una idea de que tan alejadas eran las peores y mejores predicciones, puesto que el valor del error no es lo suficientemente diciente

[15] HINDAWI Advances in Meteorology, Hindawi, 18 de junio del 2014 [en línea]. <https://www.hindawi.com/journals/amete/2014/245104/> [citado en 25 de enero del 2018]
[16] INTRODUCTION. MATPLOTLIB: Python Plotting – Matplotlib 2.1.1 Documentation [en línea]. <https://matplotlib.org> [citado en 18 de enero del 2018]

para saberlo. Más adelante, se utilizaron para representar en gráficos de barras los errores de distintas combinaciones de datos cuando se reutilizaban modelos. Y finalmente, la visualización también fue crucial para hacer correlaciones manuales de datos aislados (*outliers*).

6.2.4 Experimentación y Reutilización de Modelos

6.2.4.1 Combinaciones En esta nueva etapa, se empezó a experimentar con distintos arreglos de datos con la finalidad de evaluar cuales eran los que reflejaban mejores resultados, y que factores harían esto posible. Como era de suponer, los primeros experimentos que se hicieron, tomaban solamente datos de una estación como datos de entrenamiento, y los mismos como datos de prueba (caso A). Posteriormente, se experimentó usando datos de entrenamiento con datos de una estación y datos de prueba de otra estación (caso B). Finalmente, se experimentó utilizando datos de entrenamiento provenientes de dos estaciones diferentes, y de dos estaciones más para los datos de prueba (caso C). A continuación, se ilustran los tres ejemplos para cada una de las estrategias anteriormente descritas.

Figura 28. **Combinación con datos de 1 estación: caso A**

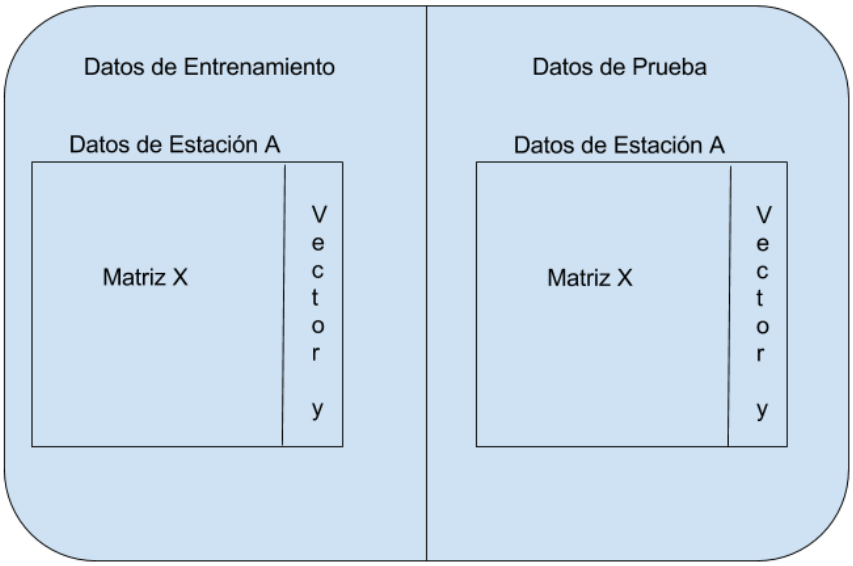


Figura 29. **Combinación con datos de 2 estaciones: caso B**

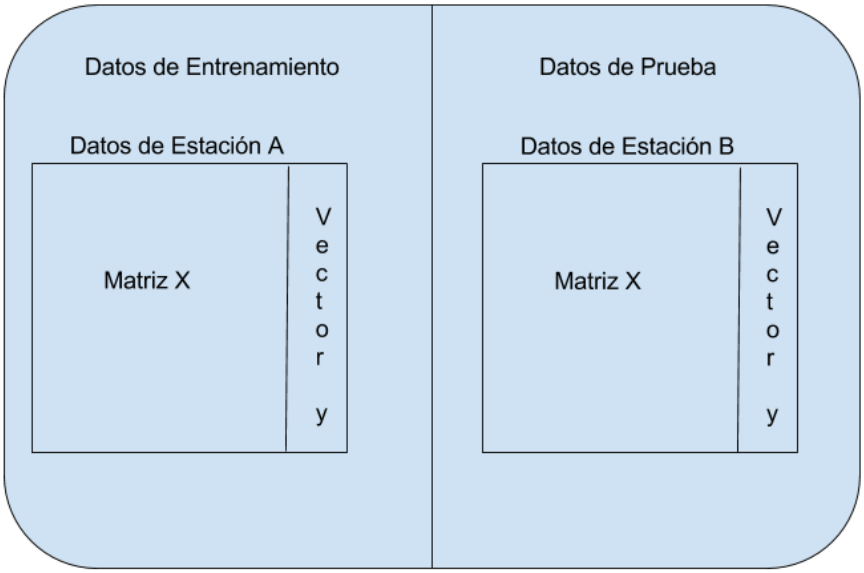
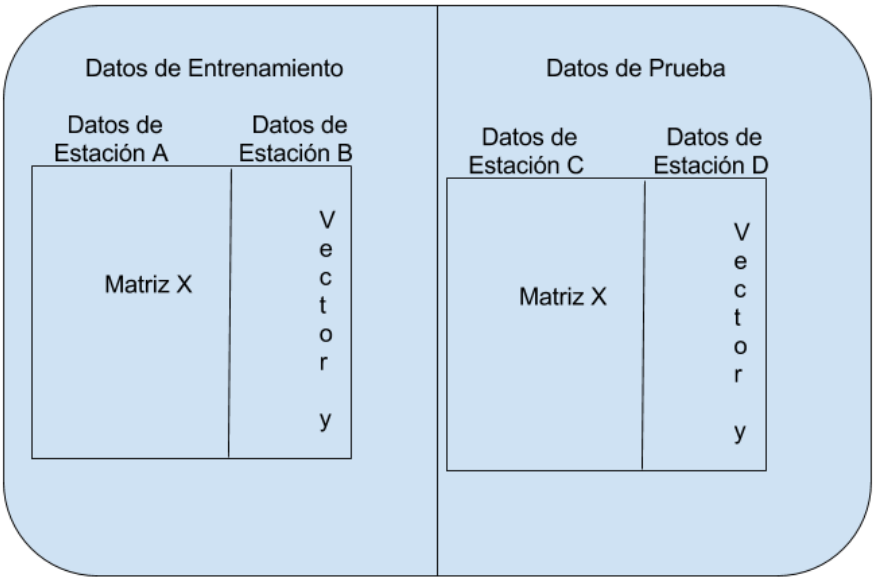


Figura 30. **Combinación con datos de 4 estaciones: caso C**



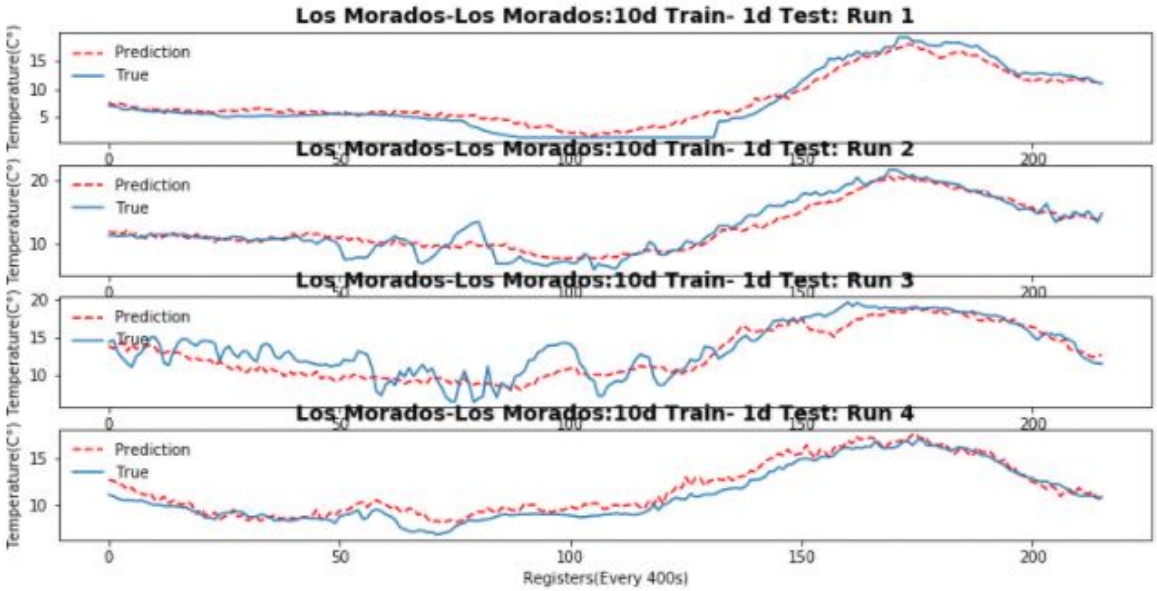
Con fines explicativos, a continuación, se mostrará un ejemplo por cada caso. El experimento para cada uno consiste en calcular el rendimiento de las predicciones, usando *Linear Regression*, de modelos cuando se analizan 15 días de datos de 19

detectores cercanos a cada estación y se toman 10 días de entrenamiento para la configuración de validación cruzada, pero variando los datos de entrenamiento (matriz X, vector y) y de prueba (matriz X, vector y) según sea el caso.

Tabla 1. Caso A, predicción en la estación Los Morados usando *Linear Regression*

Con **Linear Regression**: Los Morados para entrenamiento- Los Morados para probar	
Error en grados	Desviación estándar
1.06	0.24

Figura 31. Caso A, predicción en la estación Los Morados usando *Linear Regression*



Como se puede apreciar en la figura 31 y en la tabla 1, la media de los errores de las predicciones y las desviaciones son pequeñas. Esto ocurre, como consecuencia de que como la temperatura en un día y en un punto geográfico determinado es

similar al de días posteriores, el modelo que se ajusta a un día será parecido al de los días siguientes.

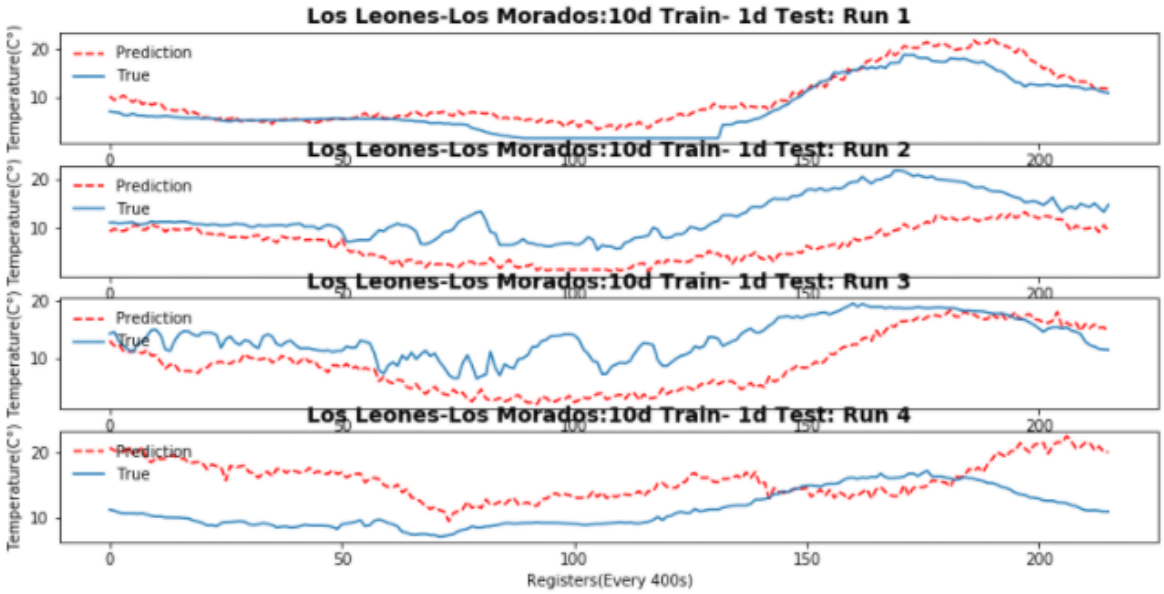
Una analogía de esto, es cuando un profesor prepara a sus estudiantes con material de estudio sobre los temas que aparecerán en un parcial. Una vez los estudiantes enfrentan la prueba, es de esperarse que les irá bien.

El siguiente ejemplo, analiza el segundo caso, en donde se entrena el modelo a partir de datos de una estación y se reutilizan para probarlos en datos de otras estaciones. En este caso se entrenan con la estación de Los Leones, y se prueban en la estación de Los Morados. A diferencia del ejemplo anterior, aquí se utilizó *Random Forest* como estimador.

Tabla 2. Caso B, predicción con las estaciones Los Leones y Los Morados usando *Random Forest*

Con **Random Forest Regression**: Los Leones para entrenamiento- Los Morados para probar	
Error en grados	Desviación estándar
4.51	1.3

Figura 32. **Caso B, predicción con las estaciones Los Leones y Los Morados usando *Random Forest***



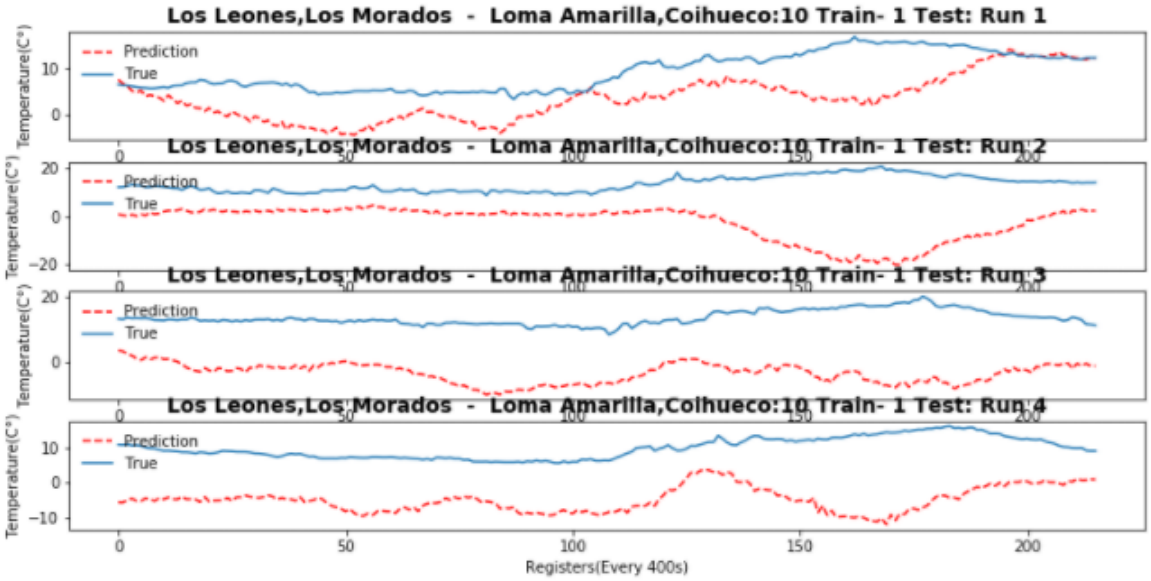
En esta ocasión, los errores aumentaron en comparación al caso anterior, pero no de forma tan pronunciada como se esperaría. Una posible razón a este fenómeno, es que el nuevo estimador hace mejores predicciones que el que se utilizó en el ejemplo anterior, porque los datos de prueba son particularmente diferentes a los de entrenamiento.

A continuación, se mostrará el caso en el que se tomó la matriz X de entrenamiento con los datos de Los Leones, el vector y de entrenamiento con datos de Los Morados, y para prueba, la matriz X con datos de Loma Amarilla y el vector y con datos de Coihueco para hacer predicción usando *Support Vector Regression*.

Tabla 3. **Caso C, predicción con todas las estaciones usando *Support Vector Regression***

Con **Support Vector Regression**: Los Leones y Los Morados para entrenamiento- Loma Amarilla y Coihueco para probar	
Error en grados	Desviación estándar
13.39	4.53

Figura 33. **Caso C, predicción con todas las estaciones *Support Vector Regression***



Entrenando y probando con datos provenientes de diferentes posiciones geográficas, y utilizando un estimador con todos sus parámetros por defecto, era de esperar la poca similitud de las curvas de las predicciones con las curvas de los datos reales. Lo bueno de este último enfoque, es que permite la exploración de la flexibilidad de los modelos al momento de ser reutilizados. Esto, entre otras razones, permite tener una mirada del panorama general, lo cual facilita el ejercicio del mejor modelo.

Para finalizar el apartado, es importante recordar que el desarrollo del framework no fue creado solamente de forma incremental, sino que fue adquiriendo mejoras en la medida que se requería. Entre las funciones adicionales agregadas más importantes, se encuentran las que eran responsables de graficar los errores de las predicciones y los distintos atributos de los datos de los detectores, las encargadas de mejorar la versatilidad de las funciones para seleccionar el número de detectores o atributos de detectores a analizar, las responsables de contar registros por días en cada detector y finalmente las encargadas de mejorar la forma de representar los errores de varios experimentos en un solo dataframe.

7. RESULTADOS OBTENIDOS

Teniendo una idea clara de cómo es el flujo de trabajo, y de las herramientas que se desarrollaron para dar solución al problema de *Machine Learning* que se abordó, en esta sección se expondrán los mejores experimentos que se llevaron a cabo con sus respectivos resultados, y que lograron determinar los indicios que permitirían seleccionar los mejores modelos predictivos.

7.1 COMBINACIONES CON DATOS DE 2 ESTACIONES

Como primera medida, se mostrarán gráficas en las cuales se vea manifiesto el rendimiento de las predicciones mediante el error (barras de colores) junto con la desviación estándar de cada error (líneas verticales negras), a medida que se utilizaban distintas configuraciones (número de días de entrenamiento) de validación cruzada y para cada uno de los tres distintos estimadores que se utilizaron.

Es válido tener en cuenta que para este ejercicio y el siguiente, se utilizaron 15 días de datos. Las gráficas ilustran el error

Figura 34. Comparativa de errores para *Linear Regression* de diferentes configuraciones de validación cruzada cuando se usan datos de Los Leones como datos de entrenamiento

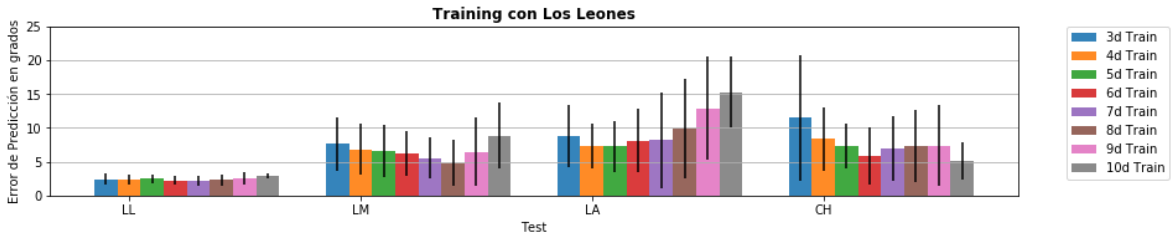


Figura 35. Comparativa de errores para *Linear Regression* de diferentes configuraciones de validación cruzada cuando se usan datos de Los Morados como datos de entrenamiento

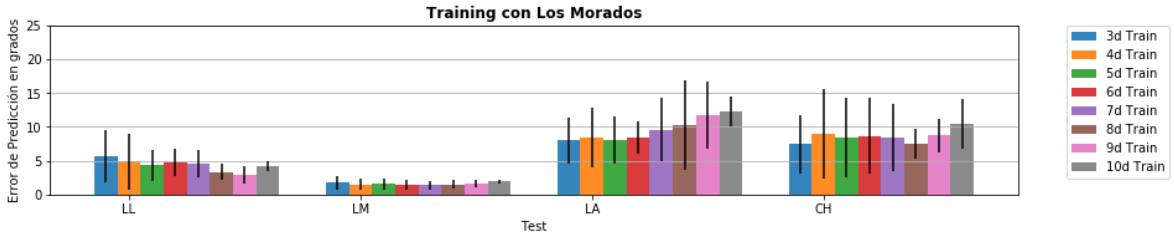


Figura 36. Comparativa de errores para *Linear Regression* de diferentes configuraciones de validación cruzada cuando se usan datos de Loma Amarilla como datos de entrenamiento

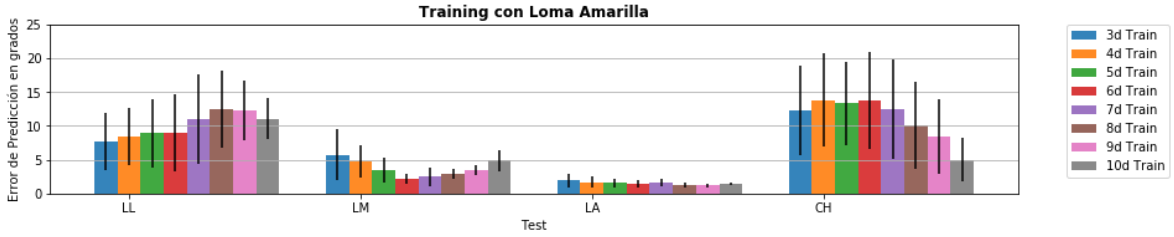
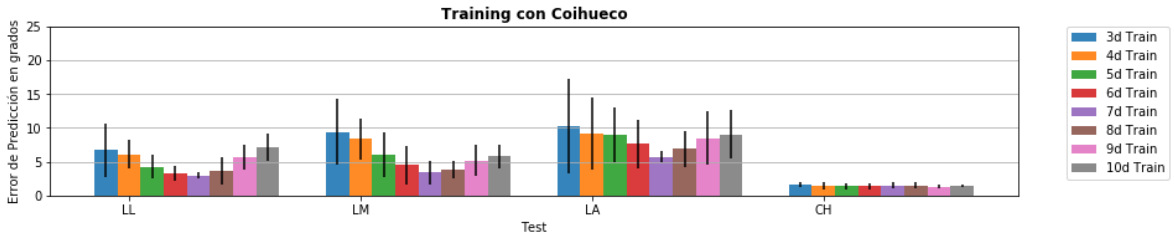


Figura 37. Comparativa de errores para *Linear Regression* de diferentes configuraciones de validación cruzada cuando se usan datos de Coihueco como datos de entrenamiento



Lo primero que se puede interpretar a partir de las predicciones basadas en *Linear Regression*, es que los mejores resultados siempre se dan en los casos en los cuales no se reutilizaron los modelos. Esto se puede constatar viendo los resultados de los errores en los siguientes casos: LL-LL, LM-LM, LA-LA, CH-CH.

También se puede tener en cuenta que, en muchos casos, las predicciones son mejores cuando se utiliza un mayor número de días para entrenar. Esto quiere decir que, si el modelo se entrena usando 3 días, lo más probable es que los resultados sean mucho menos desfavorables que cuando se entrena usando 4 días.

Figura 38. Comparativa de errores para *Random Forest* de diferentes configuraciones de validación cruzada cuando se usan datos de Los Leones como datos de entrenamiento

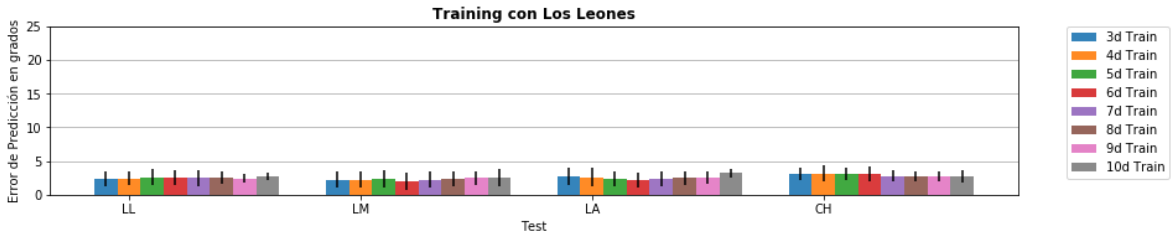


Figura 39. Comparativa de errores para *Random Forest* de diferentes configuraciones de validación cruzada cuando se usan datos de Los Morados como datos de entrenamiento

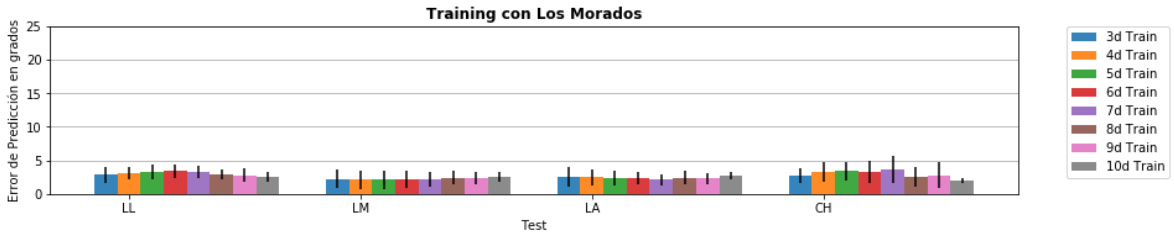


Figura 40. Comparativa de errores para *Random Forest* de diferentes configuraciones de validación cruzada cuando se usan datos de Loma Amarilla como datos de entrenamiento

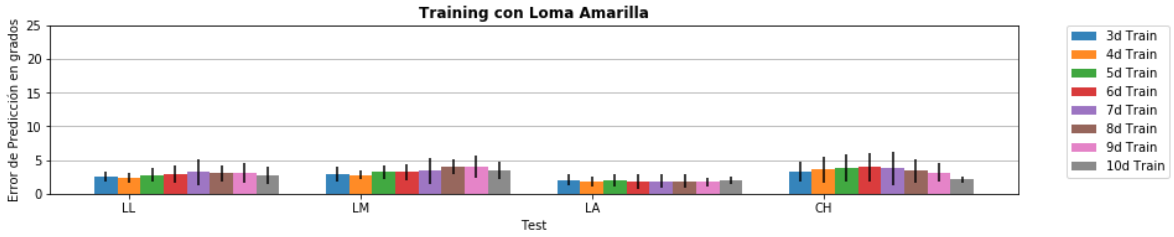
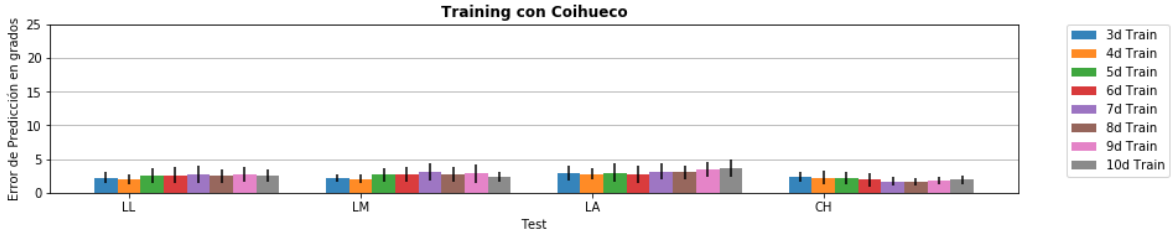


Figura 41. Comparativa de errores para *Random Forest* de diferentes configuraciones de validación cruzada cuando se usan datos de Coihueco como datos de entrenamiento



Es importante saber que se establecieron 10 árboles de decisión como parámetro de *Random Forest.*

En comparación con los resultados de *Linear Regression*, los resultados de R*andom Forest* son muchos mejores. Pasaron de ser errores que fluctuaban entre 2 a 15 grados, a ser errores comprendidos entre 1.6 y 4 grados.

A partir de la interpretación gráfica, se pueden extraer dos resultados importantes. El primero, es que con el nuevo enfoque el rendimiento de las predicciones no guarda relación con el número de días de entrenamiento. En la mayoría de los casos, pese a que se usan más días de datos para entrenar, los errores aumentan o disminuyen de forma arbitraria.

El segundo, es que el rendimiento de las predicciones en cuyos casos no se reutiliza el modelo, no son significativamente mejores que el resto de predicciones.

Figura 42. Comparativa de errores para *Support Vector Regression* de diferentes configuraciones de validación cruzada cuando se usan datos de Los Leones como datos de entrenamiento

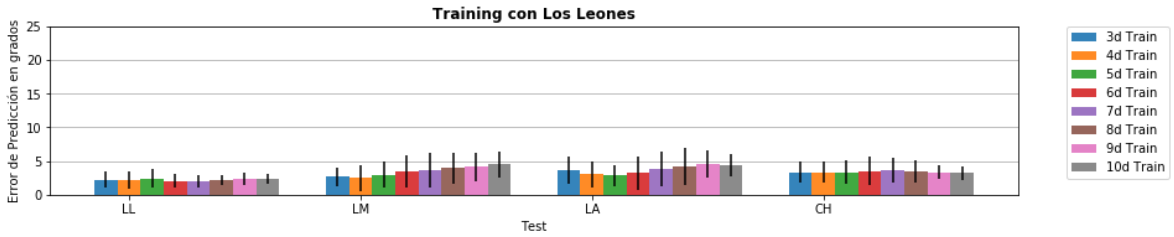


Figura 43. Comparativa de errores para *Support Vector Regression* de diferentes configuraciones de validación cruzada cuando se usan datos de Los Morados como datos de entrenamiento

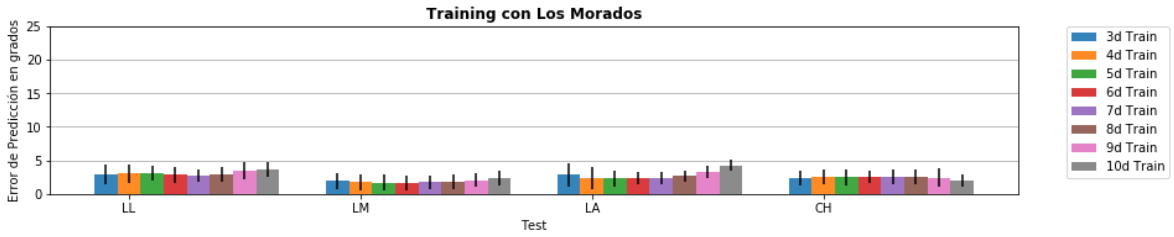


Figura 44. Comparativa de errores para *Support Vector Regression* de diferentes configuraciones de validación cruzada cuando se usan datos de Loma Amarilla como datos de entrenamiento

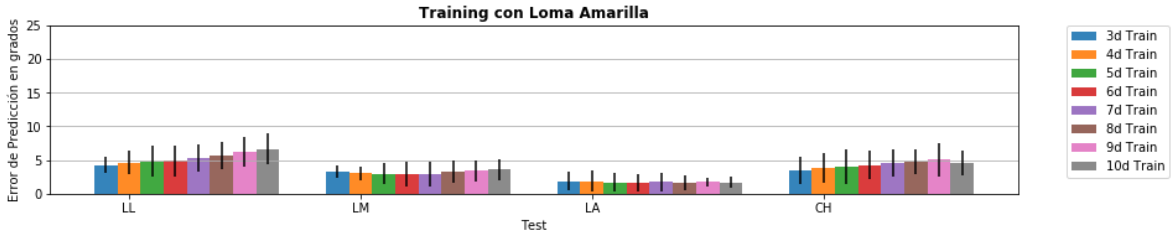
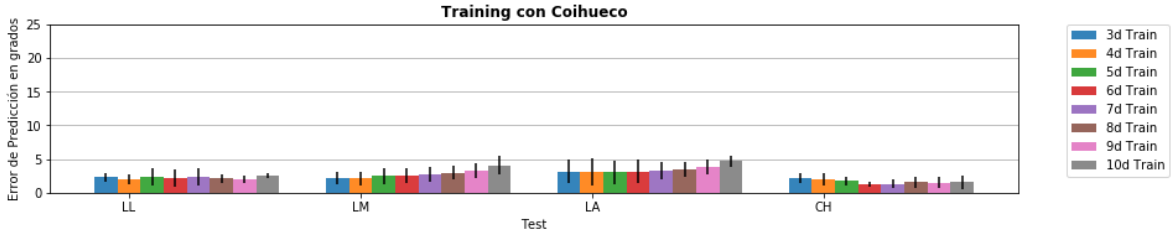


Figura 45. Comparativa de errores para *Support Vector Regression* de diferentes configuraciones de validación cruzada cuando se usan datos de Coihueco como datos de entrenamiento



Es importante destacar que para la utilización del estimador *Support Vector Regression*, se establecieron los siguientes parámetros: *kernel Radial-bias(rbf), gamma=10**-3, C=15 (parámetro de penalización), epsilon=13**-3, degree=3.*

De las gráficas de los errores para las predicciones usando *Support Vector Regression*, se puede afirmar que los resultados son mucho mejores que los de *Linear Regression*, aunque no son tan buenos como los de *Random Forest* con 10 árboles de decisión. Este último resultado, surge del hecho de que existen casos en los que la reutilización de modelos entrenados con *Support Vector Regression* superan los 5 grados de error, situación que nunca se presentó para el caso anteriormente descrito.

7.2 COMBINACIONES CON DATOS DE 4 ESTACIONES

Con la finalidad de tener mayor información que permita tener certeza de los mejores modelos a seleccionar, a continuación, se mostrarán los resultados para el caso C expuesto en el inciso 6.2.5.1, donde los datos de cada experimento provienen de 4 estaciones diferentes (o iguales).

En aras de mejorar la inteligibilidad, se representaron los errores de las distintas combinaciones en un mapa de calor, en donde los colores más oscuros representan valores mayores de error, que los que son representados por colores más claros. Con el mismo objetivo, la media de los errores y las desviaciones se representan en gráficas diferentes.

Figura 46. Media de los errores (izquierda) y desviación estándar (derecha) para *Linear Regression* cuando se usan 5 detectores

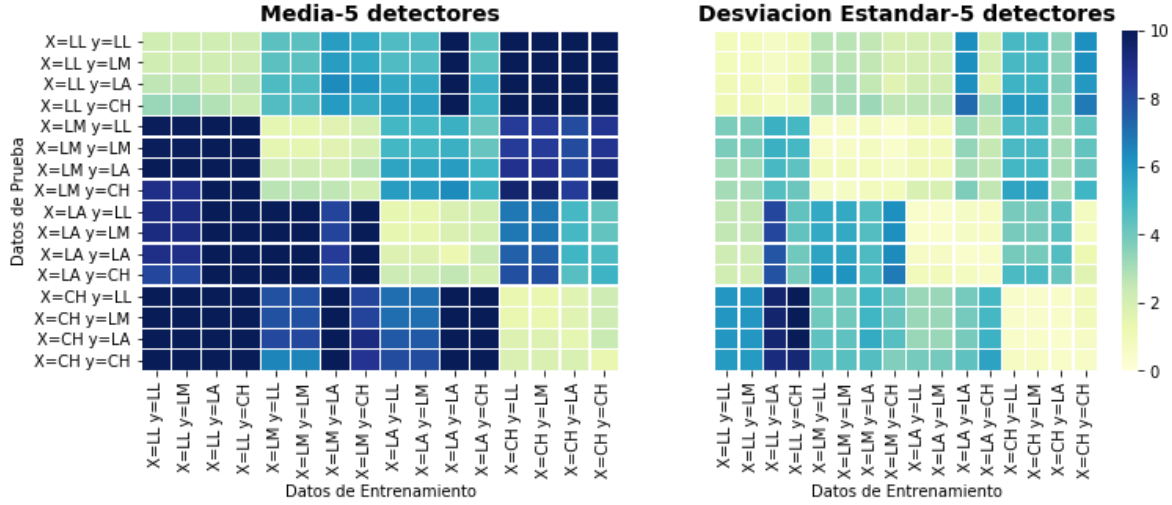


Figura 47. Media de los errores (izquierda) y desviación estándar (derecha) para *Linear Regression* cuando se usan 10 detectores

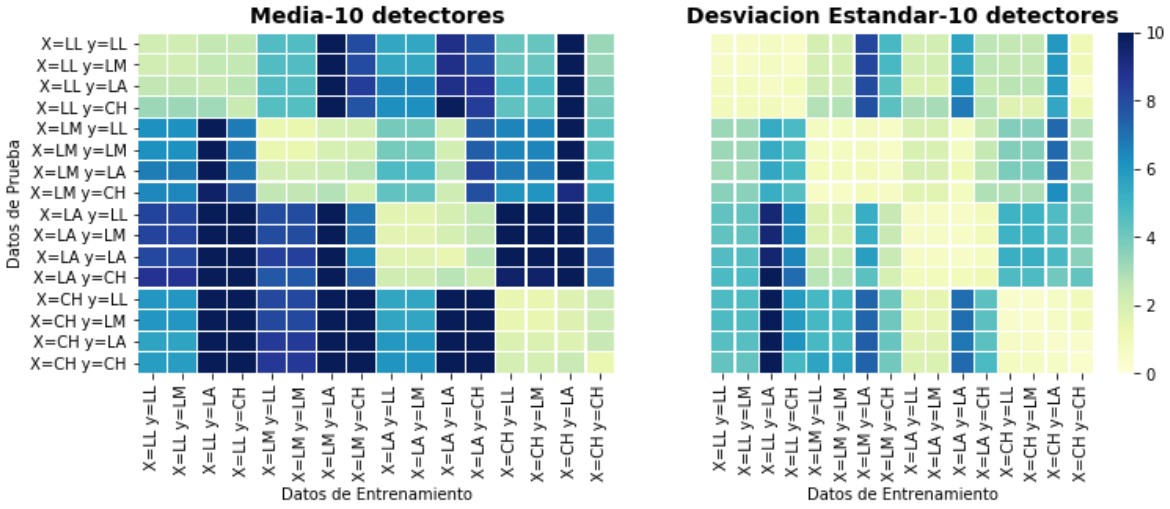


Figura 48. Media de los errores (izquierda) y desviación estándar (derecha) para *Linear Regression* cuando se usan 15 detectores

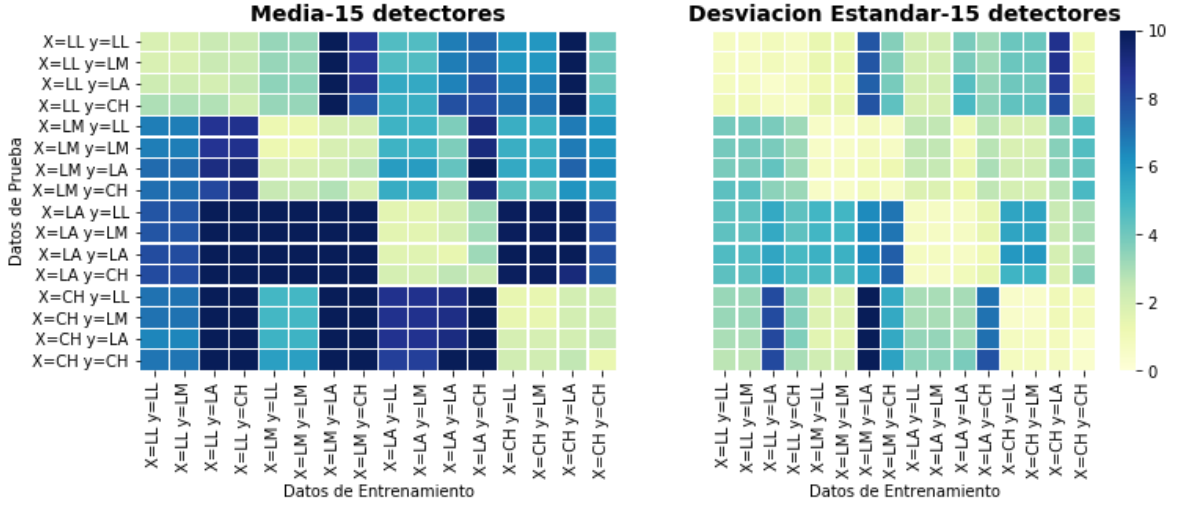


Figura 49. Media de los errores (izquierda) y desviación estándar (derecha) para *Linear Regression* cuando se usan 19 detectores

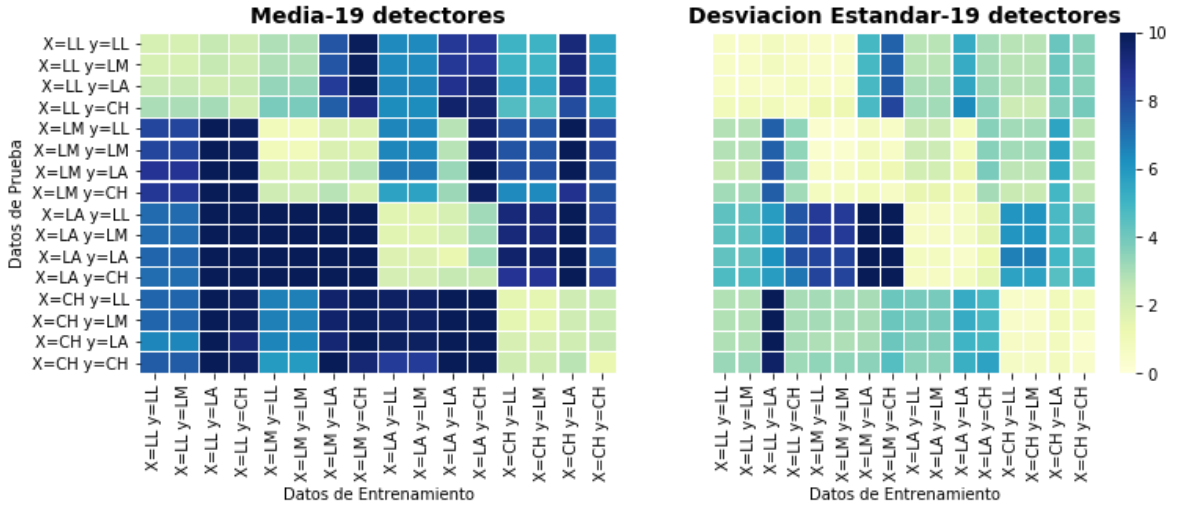


Figura 50. **Media de los errores (izquierda) y desviación estándar (derecha) para *Random Forest* cuando se usan 5 detectores**

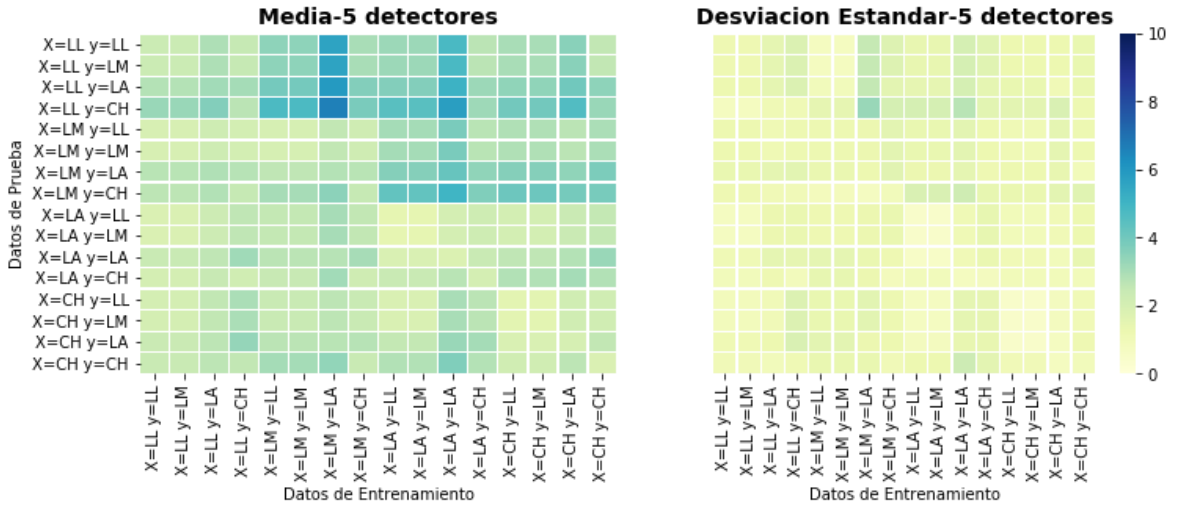


Figura 51. **Media de los errores (izquierda) y desviación estándar (derecha) para *Random Forest* cuando se usan 10 detectores**

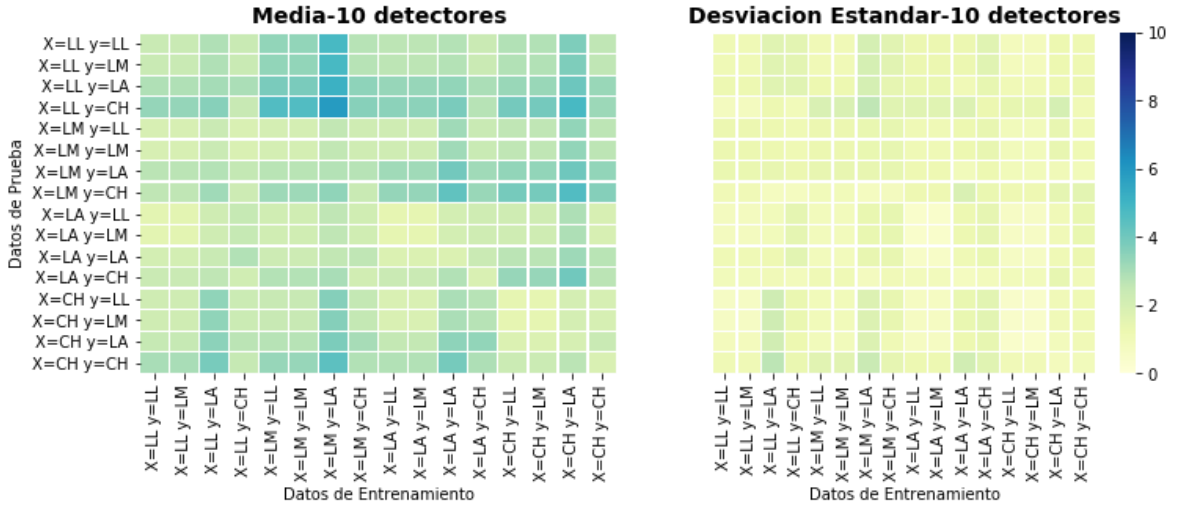


Figura 52. Media de los errores (izquierda) y desviación estándar (derecha) para *Random Forest* cuando se usan 15 detectores

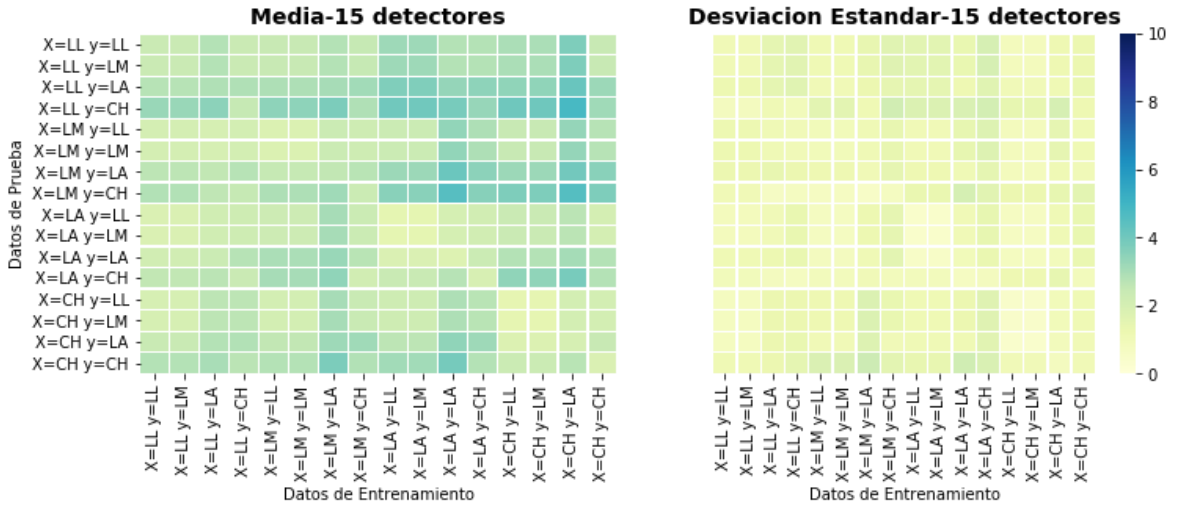


Figura 53. Media de los errores (izquierda) y desviación estándar (derecha) para *Random Forest* cuando se usan 19 detectores

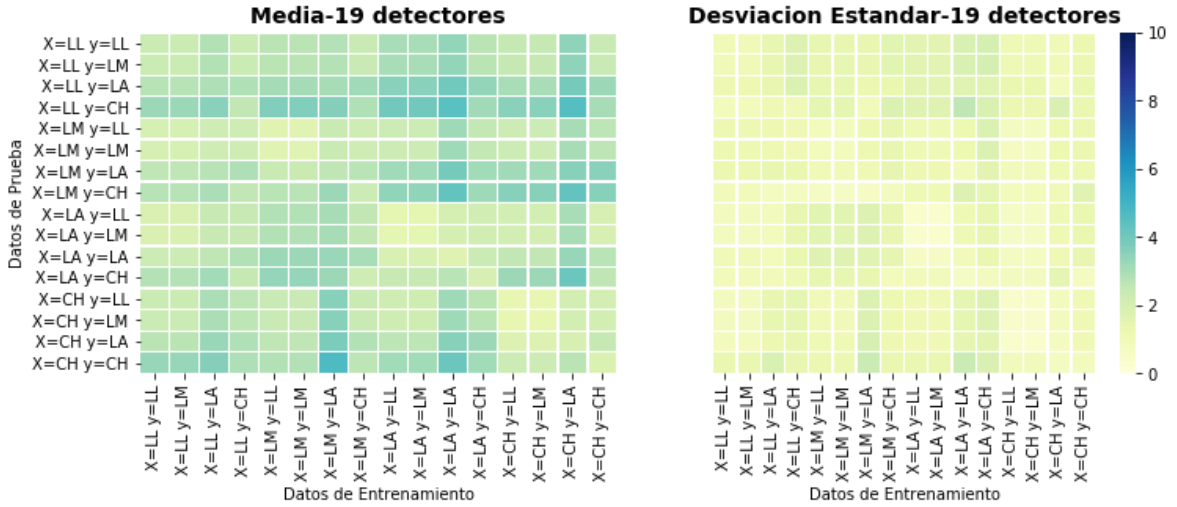


Figura 54. Media de los errores (izquierda) y desviación estándar (derecha) para *Support vector Regression* cuando se usan 5 detectores

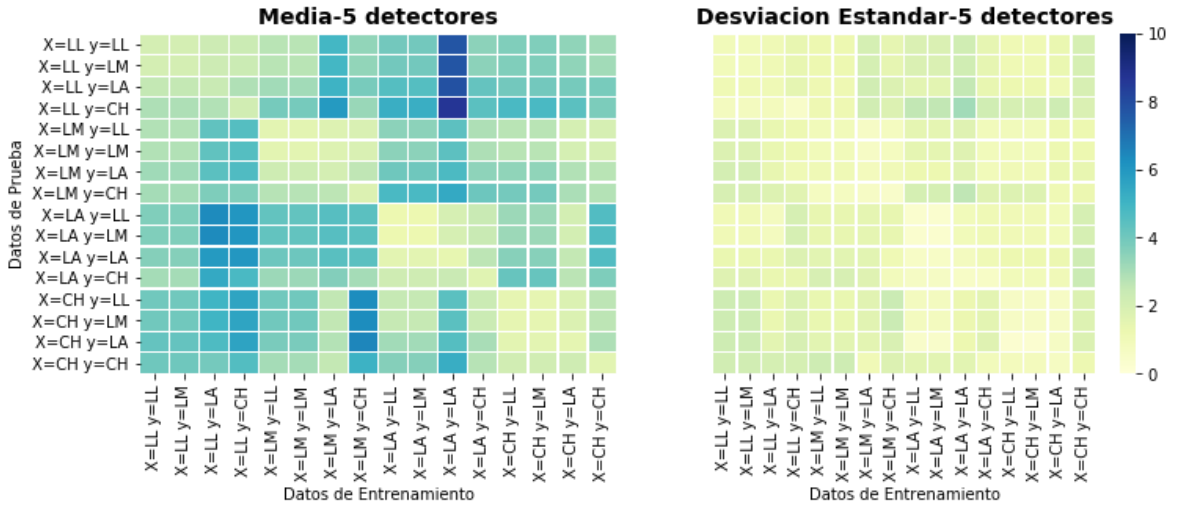


Figura 55. Media de los errores (izquierda) y desviación estándar (derecha) para *Support vector Regression* cuando se usan 10 detectores

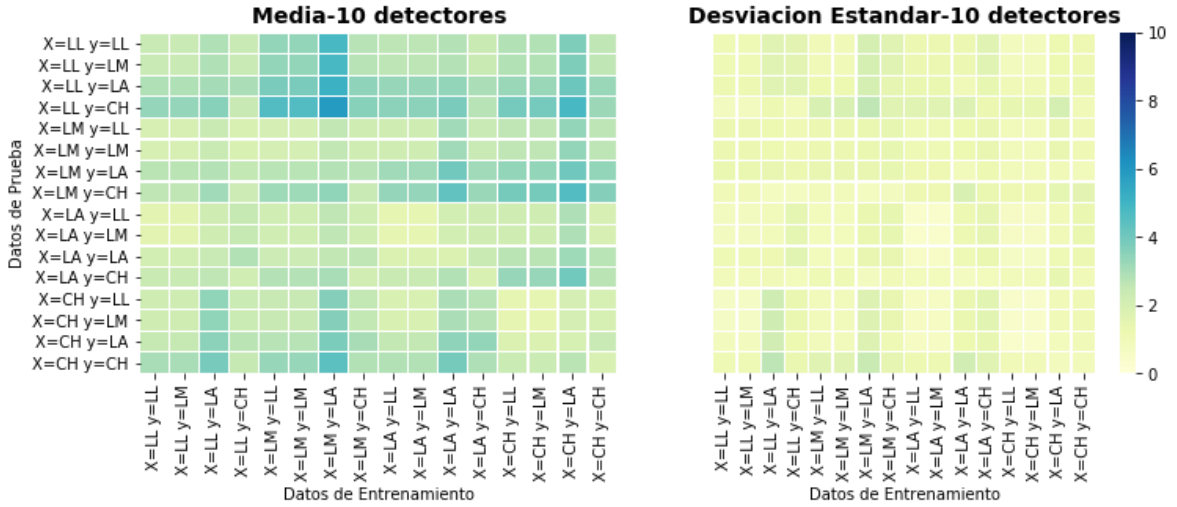


Figura 56. **Media de los errores (izquierda) y desviación estándar (derecha) para *Support vector Regression* cuando se usan 15 detectores**

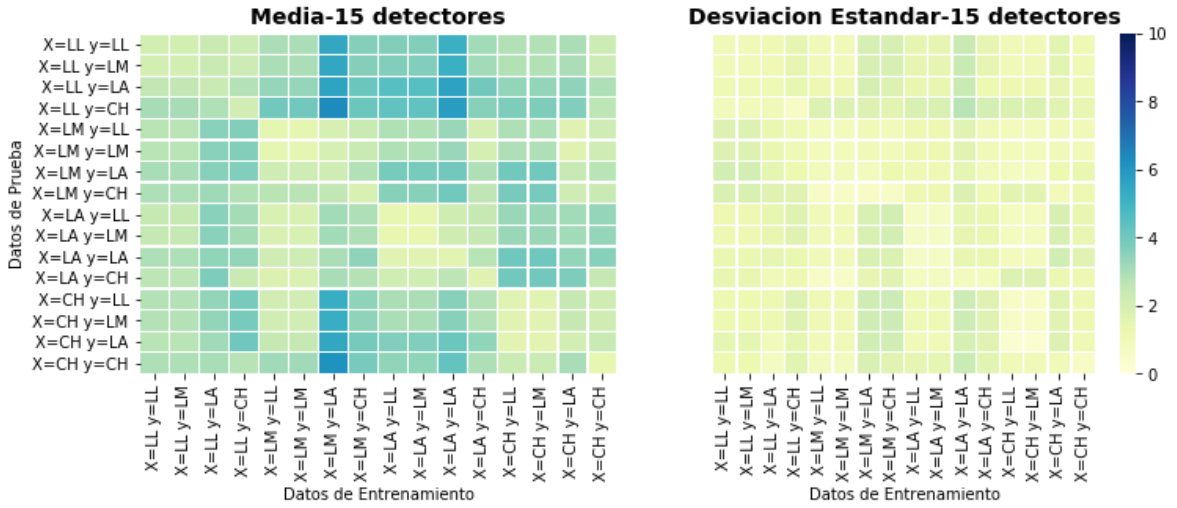
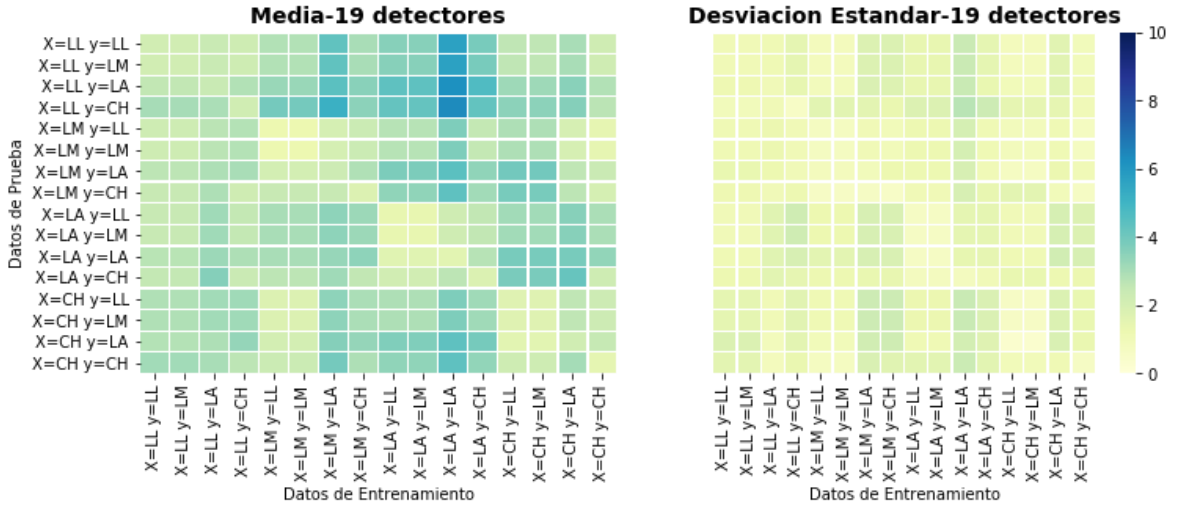


Figura 57. **Media de los errores (izquierda) y desviación estándar (derecha) para *Support vector Regression* cuando se usan 19 detectores**



7.3 ANÁLISIS

Dando una mirada retrospectiva al rendimiento de todas las combinaciones usadas en los distintos experimentos para cada estimador y usando para este nuevo caso, el cambio de la variable que ajusta el número de detectores, se pueden sintetizar los resultados generales que se expondrán a continuación:

- Al igual que en el caso en donde se utilizaban datos provenientes de dos estaciones por experimento, en este nuevo enfoque, los mejores resultados con *Linear Regression* son aquellos en los cuales no se reutilizaron los distintos modelos. Esto se puede corroborar observando la diagonal de cada uno de los 4 mapas de calor para este estimador, las cuales tienen un color más claro que el resto de casillas. Esto permite concluir, que los modelos basados en *Linear Regression* no son óptimos para predecir la temperatura en cualquier punto del arreglo de detectores situados en el observatorio.
- Para los experimentos con *Random Forest* se puede afirmar que de la misma forma que cuando solo se combinaban datos de 2 estaciones, los resultados fueron considerablemente mejores que los experimentos con el primer estimador. Otro rasgo importante de estos experimentos, es que los resultados mejoraron después de que se utilizaron 15 días de datos de entrenamiento, cosa que no sucedió con los de *Linear Regression*.
- Los experimentos con *Support Vector Regression* arrojaron buenos resultados cuando los modelos fueron entrenados con datos provenientes de 19 detectores. Resultados que son comparables con los estimados usando *Random Forest* cuando se utilizaron datos de 15 o 19 detectores.

7.4 SELECCIÓN DEL MEJOR MODELO

A partir del anterior análisis se determinó, que los mejores modelos para predecir la temperatura en cualquier punto del arreglo de 1660 detectores ubicados en el observatorio Pierre Auger, usando solamente 6 señales (*fPMT[0], fPMT[1], fPMT[2], fElectT, fBatteryT[0], fBatteryT[1]*) de cada uno de los 19 detectores más cercanos a cada una de las 4 estaciones meteorológicas, son los siguientes:

- El modelo que usa como estimador *Random Forest* con 10 árboles de decisión y que se entrena con 6 días de datos.
- El modelo que usa como estimador Support Vector Regression con los parámetros siguientes: kernel Gaussiano(rbf), gamma=10**-3, C=15 (parámetro de penalización), epsilon=13**-3, degree=3. Este modelo se entrena con 6 días de datos.

8. CONCLUSIONES

Las conclusiones que se pudieron extraer a partir del trabajo realizado son las siguientes:

- Llevar a cabo la exploración de modelos predictivos con *Machine Learning* no es un ejercicio en el que siempre se debe seguir invariablemente una serie de pasos. La forma de abordar cada caso depende de la naturaleza de cada problema.

- A pesar de que la etapa de experimentación fue compleja, la fase que requirió más tiempo fue la de preprocesamiento, debido a la incompletitud, corrupción o repetición de datos de monitoreo de algunos detectores.

- Una buena práctica para proyectos de análisis de datos, es desarrollar un framework de trabajo en aras de facilitar el ejercicio de experimentación.

- Al trabajar con cantidades ingentes de datos, es aconsejable usar de forma apropiada las herramientas de graficación.

- Buscar un modelo reutilizable es una buena práctica, porque al ser aplicable a una mayor cantidad de datos de entrada, es más sencilla y funcional su implementación.

- Pese a que experimentar con estimadores como *Random Forest* o *Support Vector Regression* requiere más ajuste de parámetros en comparación con el estimador de *Linear Regression*, esta personalización se puede utilizar a favor para encontrar mejores modelos.

- La importancia de solucionar un problema real de analítica de datos no solamente radica en que se gane experticia para abordar más proyectos de la misma naturaleza, que se aprenda a usar el conjunto de herramientas y tecnologías necesarias para llevar a cabo dicha tarea, que se ponga en práctica principios de ingeniería y de investigación para diseñar un framework que facilite el trabajo, ni que se gane experimentación para posteriormente extraer conclusiones a partir de resultados. Sino que permite poner en práctica el ejercicio de pensar. Una habilidad fundamental para cualquier ingeniero.

BIBLIOGRAFÍA

AN IOT ENVIRONMENTAL DATA COLLECTION System for Fungal Detection in Crop Fields – IEEE Conference Publication [en línea]. <http://ieeexplore.ieee.org/abstract/document/7946787/?reload=true> [citado en 25 de enero del 2018]

ANTHONY, Femi. Mastering Pandas: Master the Features and Capabilities of Pandas, a Data Analysis Toolkit for Python. Pack Publishing Ltd., 2015

CNEA Laboratorio Detección De Partículas y Radiación [en línea]. <http://fisica.cab.cnea.gov.ar/particulas/html/labdpr/auger.php> [citado en 3 de enero del 2018]

CROSS-VALIDATION: Evaluating Estimator Perfomance. 3.1. Cross-Validation: Evaluating Estimator Perfomance – Scikit-Learn 0.19.1 Documentation. [en línea]. <http://scikit-learn.org/stable/modules/cross_validation.html> [citado en 19 de enero del 2018]

CSAIL Random Search for Hyper-Parameter Optimization [en línea]. <http://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf> [citado en 6 de enero del 2018]

CSV - CSV FILE READING AND WRITING. 13.1. Csv – CSV File Reading and Writing – Python 2.7.14 Documentation [en línea]. <https://docs.python.org/2/library/csv.html> [citado en 16 de enero del 2018]

DAILY MEANS AMBIENT TEMPERATURA PREDICTION Using Artificial Neural Network Method: A Case Study of Turkey. Renewable Energy, Pergamon, 6 de septiembre del 2008 [en línea]. <https://www.sciencedirect.com/science/article/pii/S0960148108002851> [citado en 25 de enero del 2018]

ESCUELA VERANO Análisis de Series de Tiempo [en línea]. <http://www.escuela-verano.otrasenda.org/wp-content/uploads/2015/06/curso_series.pdf> [citado en 15 de enero del 2018]

GERÓN, Aurélien. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly, 2017.

HEYDT, Michael. Mastering Pandas for Finance. Pack Publishing Ltd, 2015.

HINDAWI Advances in Meteorology, Hindawi, 18 de junio del 2014 [en línea]. <https://www.hindawi.com/journals/amete/2014/245104/> [citado en 25 de enero del 2018]

INTRODUCTION. MATPLOTLIB: Python Plotting – Matplotlib 2.1.1 Documentation [en línea]. <https://matplotlib.org> [citado en 18 de enero del 2018]

LAND SURFACE TEMPERATURE AND SCALING Factors for Different Satellites Datasets [en línea]. <https://www.omicsonline.org/open-access/land-surface-temperature-and-scaling-factors-for-different-satellitesdatasets-jgg-1000223.pdf> [citado en 4 de enero del 2018]

McKinney, Wes. Python for Data Analysis: Data Wrangling with Pandas, Numpy, and IPython. O'Reilly Media, Inc., 2017

NELLI, Fabio. Python Data Analytics: Data Analysis and Science Using PANDAs, Matplotlib and the Python Programming Language. Apress, 2015

ROGERS, Simon; GIROLANI, Mark. A First Course in Machine Learning. CRC Press, 2017.

SCIKIT-LEARN Support Vector Machines. 1.4 Support Vector Machines – Scikit-Learn 0.19.1 Documentatio [en línea]. <http://scikit-learn.org/stable/modules/svm.html> [citado en 7 de enero del 2018]

SPRINGER Random Forest [en línea]. <https://link.springer.com/content/pdf/10.1023%2FA%3A1010933404324.pdf> [citado en 7 de enero del 2018]

Swamynathan, Manohar. Mastering Machine Learning with Python in six Steps: a Practical Implementation Guide to Predictive Data Analytics using Python. Apress, 2017.

THE PIERRE AUGER COSMIC RAY OBSERVATORY. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associate Equipment, North-Holland, 8 de julio del 2015 [en línea]. <https://www.sciencedirect.com/science/article/pii/S0168900215008086> [citado en 4 de enero del 2018]

VALBUENA, Álvaro; RAMOS-POLLÁN, Raul; A. NUÑEZ, Luis; ASOREY, Hernán. Exploiting Auger monitoring data for Surface temperature prediction. Escuela de Ingeniería de Sistemas y Escuela de Física, Universidad Industrial de Santander, Bucaramanga, Colombia. Laboratorio Detección de Partículas y Radiación Centro Atómico Bariloche, Comisión Nacional de Energía Atómica, San Carlos de Barichole, Argentina.

VAN, TAN PHAN, et al. Seasonal Prediction of Surface Air Temperature across Vietnam Using the Regional Climate Model Version 4.2 (RegCM4.2).