Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Grado en Ingeniería Informática

Proyecto Final de Tecnologías de la Información para las Organizaciones

---

Informe

Machine Learning (R, Caret, Algoritmos y Métodos de Entrenamiento)

Eduardo Ernesto Brito Sánchez
Sergio Delgado López
Pablo González González
Alba Cruz Torres
Jefry Izquierdo Hernández
Daniel Abreu García

San Cristóbal de La Laguna, 05 de Enero de 2020

C/ Padre Herrera s/n
38207 La Laguna
Santa Cruz de Tenerife. España

T: 900 43 25 26

ull.es

Índice General

Capítulo 1. Introducción

1.1 ¿Qué es un modelo?

Una definición concisa del concepto de modelo en el mundo del Machine Learning es una cuestión particularmente compleja. Dado que su uso no se limita solo a expertos de machine learning sino a profesionales de otras disciplinas que necesitan aplicar técnicas de ML. Se puede apreciar en la literatura científica la definición de modelo como:

- Modelos estadísticos relacionados intrínsecamente con distribuciones de probabilidad.
- Modelos matemáticos, utilizados como descripciones de un sistema que utiliza conceptos matemáticos y del lenguaje.
- Modelos de datos utilizados en machine learning, conformado por las columnas incluidas, las fuentes de datos y todos los metadatos.
- En algunas situaciones todas estas definiciones se expanden para incluir estructuras de machine learning por encima de las ecuaciones y los metadatos, tales como la especificaciones para las redes neuronales.

1.2 ¿Qué es machine learning?

Cuando se habla de Machine Learning o Aprendizaje automático [1], se refiere al conjunto de algoritmos que permite identificar patrones presentes en los datos y crear con ellos modelos que les representen. Después de que se creen dichos modelos, se utilizan para predecir información sobre hechos o eventos que aún no han sido observados. Una particularidad que se debe tener en cuenta es que los sistemas de machine learning sólo pueden reconocer patrones presentes en los datos utilizados para entrenarlos, por tanto, no pueden reconocer patrones nuevos. Su utilidad escala mientras más estable (y por tanto predecible) sea el ámbito que analicen.

Capítulo 2. Características del DataSet

2.1 Selección del DataSet

Para tratar con datos más actuales, se escogió el DataSet de 2016, que recoge las mediciones en la estación meteorológica de Castillo del Romeral, del cual se proporcionarán más detalles a continuación.

2.2 Análisis exploratorio del DataSet

2.2.1 Tipos de variables

Las variables observables en el DataSet son las siguientes:

- Hora: hora de la lectura en formato 24h
- SO2: La lectura de dióxido de azufre en la fecha y hora marcada.
- NO:Lectura del óxido de nitrógeno en la fecha y hora marcadas.
- NO2: Lectura del dióxido de nitrógeno en la fecha y hora marcadas.
- PM10: Lectura de las partículas sólidas o líquidas de polvo en la fecha y hora marcadas.
- PM2,5: Lectura de las partículas en suspensión de menos de 2.5 micras en la fecha y hora marcadas.
- O3: Lectura del ozono en la fecha y hora marcadas.
- CO: Lectura del monóxido de carbono en la fecha y horas marcadas.

Los valores de los contaminantes son expresados por µg/m3.

Por defecto, viene con la configuración de España, es decir, utilizando comas como separador de parte entera y parte decimal. Modificamos el DataSet para que use puntos, lo que además facilitará el uso de las funciones de R y Caret.

La primera problemática surge a la hora de importar el DataSet al Notebook de Google Collaborative. Dado que el símbolo que se usa cuando no hay una lectura de un contaminante es “N”, la función **read_csv()** de R interpreta que el tipo de variable de las lecturas es de tipo char. Por tanto, para poder realizar operaciones numéricas, se deberá modificar dicho símbolo por un “NA” del lenguaje R. Tras estas modificaciones, se pueden observar los datos cargados de manera correcta.

2.3 La estación meteorológica: selección y geolocalización

La estación meteorológica escogida es la de Castillo del Romeral, en la zona turística sur de Gran Canaria (Islas Canarias), por la variedad de indicadores disponibles y por la posibilidad de realizar un análisis comparativo en zonas costeras.

Las coordenadas de Castillo del Romeral son:

DD: **28.800580, -15.461093**

Figura 1: Localización de la estación metereológica.

Capítulo 3. Cronograma de ejecución de proyecto

3.1 Hitos

1. Definición del proyecto a realizar. Preparado para el 19/11/2019.
2. Resultado del análisis exploratorio. Preparado para el 24/11/2019.
3. Métricas de éxito. Preparadas para el 27/11/2019.
4. Documento de las estrategias de evaluación. Preparado para el 01/12/2019.
5. Dataframe con los datos procesados (imputación de valores, varianza próxima a cero, etc.). Preparado para el 09/12/2019.
6. Modelo inicial de predicción. Preparado para el 17/12/2019.
7. Versión final del modelo. Preparado para el 30/12/2019.

3.2 Actividades

1. Definir el problema asignado para el proyecto.
2. Explorar y entender los datos que se van a emplear para crear el modelo (análisis exploratorio).
3. *Métrica de éxito*: definir una forma apropiada de cuantificar cómo de buenos son los resultados obtenidos.
4. Preparar la estrategia para evaluar el modelo: separar las observaciones en un conjunto de entrenamiento, un conjunto de validación y un conjunto de test.
5. *Preprocesar los datos*: aplicar las transformaciones necesarias para que los datos puedan ser interpretados por el algoritmo de *machine learning* seleccionado.
6. Crear un primer modelo capaz de superar unos resultados mínimos en las evaluaciones definidas previamente.
7. Gradualmente, mejorar el modelo optimizando sus hiperparámetros (parámetros que definen la configuración del modelo).
8. Evaluar la capacidad del modelo final con el conjunto de test para tener una estimación de la capacidad que tiene el modelo cuando predice nuevas observaciones.

3.3 Cronograma

El cronograma del proyecto se encuentra en una hoja de cálculo [2], y en ella aparece una tabla compuesta por una lista de actividades que se

han de realizar en el proyectos, con sus fechas de inicio y fin. La duración de cada actividad aparece representada también de forma gráfica.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	**Lista de actividades**	Inicio	Fin	11/11/2019	12/11/2019	13/11/2019	14/11/2019	15/11/2019	16/11/2019	17/11/2019	18/11/2019	19/11/2019	20/11/2019	21/11/2019	22/11/2019	23/11/2019	24/11/2019	25/11/2019
2	Preparación del proyecto	11/11/2019	16/11/2019															
3	Definición del problema	17/11/2019	19/11/2019															
4	Análisis exploratorio	20/11/2019	24/11/2019															
5	Definición de la métrica de éxito	25/11/2019	27/11/2019															
6	Preparación de las estrategías de evaluación	28/11/2019	01/12/2019															
7	Preprocesamiento de datos	02/12/2019	05/12/2019															
8	Selección de predictores	06/12/2019	09/12/2019															
9	Aproximación del primer modelo	10/12/2019	17/12/2019															
10	Evolución del modelo	18/12/2019	30/12/2019															
11	Evaluación del modelo final	24/12/2019	30/12/2019															

Figura 2: Cronograma del proyecto.

3.4 Recursos del proyecto.

1. Recursos humanos: trabajo de las personas pertenecientes al grupo.
2. Google Collaboratory: cuaderno colaborativo para desarrollo del análisis en R.
3. Google Drive: organización y almacenamiento de los archivos relacionados con el proyecto.
4. Google Docs: elaboración de documentos para las entregas.
5. R-Studio: IDE para trabajar con el lenguaje R en local.

Capítulo 4. Análisis exploratorio.

4.1. Variables disponibles en el conjunto de datos.

- Fecha: fecha de la lectura en formato dd/mm/aa
- Hora: hora de la lectura en formato 24h
- SO2: La lectura de dióxido de azufre en la fecha y hora marcada.
- NO:Lectura del óxido de nitrógeno en la fecha y hora marcadas.
- NO2: Lectura del dióxido de nitrógeno en la fecha y hora marcadas.
- PM10: Lectura de las partículas sólidas o líquidas de polvo en la fecha y hora marcadas.

- PM2,5: Lectura de las partículas en suspensión de menos de 2.5 micras en la fecha y hora marcadas.
- O3: Lectura del ozono en la fecha y hora marcadas.
- CO: Monóxido de carbono.

Capítulo 5. Selección de una clase.

A la hora de seleccionar cuál sería la clase que se va a analizar, se ha hecho hincapié en cuáles de los predictores disponibles en el dataset es más nocivo y perjudicial para el ser humano. Es un hecho que la calidad del aire se ve contaminada por muchos gases nada adecuados para la salud. Si bien es cierto que las partículas de polvo como el **PM10** (presente en estado sólido o líquido) o el **PM2,5** (se encuentra en suspensión) son bastante contraproducentes a la hora de respirarlos, uno de los gases dañinos que llama la atención es el ***Dióxido de Nitrógeno (NO2)*** [3], el cual es un compuesto químico gaseoso de color marrón amarillento formado por la combinación de un átomo de nitrógeno y dos de oxígeno y se relaciona con diversas enfermedades de las vía respiratorias como disminución de la capacidad pulmonar, bronquitis agudas, asma y se considera el culpable de los procesos alérgicos, sobre todo en niños. Se ha relacionado las exposiciones crónicas a bajo nivel con el enfisema pulmonar. Otros efectos menores son la irritación ocular y de las mucosas.

Debido a este problema, se ha decidido establecer el NO2 como clase para analizar y ver cómo evolucionarán dichos datos a lo largo de los próximos años, esperando tener unos resultados que favorezcan su disminución y aumentar así la calidad del aire.

Al basar las predicciones del modelo de Machine Learning en este predictor, se estará tratando un caso de **regresión,** pues todos los valores que presenta el dataset seleccionado son continuos.

Capítulo 6. División de los datos en entrenamiento y test.

Evaluar la capacidad predictiva de un modelo consiste en comprobar cómo de próximas son sus predicciones a los verdaderos valores de la variable respuesta. Para poder cuantificar de forma correcta este error, se necesita disponer de un conjunto de observaciones, de las que se conozca la variable respuesta, pero que el modelo no haya “visto”, es decir, que no

hayan participado en su ajuste. Para conseguir esto se ha decidido trabajar con una proporción de 80-20, donde se usará para la primera parte de entrenamiento el 80% de los datos y el 20 restante para comprobar la respuesta.

```
train <- createDataPartition(y = na.omit(datos_castillo$NO2), p = 0.8,
   list = FALSE, times = 1)
datos_train <- datos_castillo[train, ]
datos_test  <- datos_castillo[-train, ]
```

Figura 3: Código de división de datos.

Capítulo 7. Preprocesado de variables.

7.1. Imputación de variables.

El conjunto de datos a trabajar inicialmente tiene el aspecto que se muestra en la figura, por ello hay que proceder con la imputación de variables con valores ausentes, y también tener en cuenta establecer que los valores que están en el conjunto a un formato aceptado por R, y también que convenga a la convivencia entre R y Caret, para facilitar operaciones futuras con los datos.

```
Observations: 8,784
Variables: 8
$ Hora  <chr> "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", …
$ SO2   <chr> "3", "3", "4", "5", "3", "3", "3", "3", "3", "4", "3", "3", "3"…
$ NO    <chr> "13", "6", "5", "4", "5", "4", "6", "8", "10", "15", "7", "5", …
$ PM1.0 <chr> "67", "71", "15", "47", "66", "43", "18", "16", "52", "42", "38…
$ PM2.5 <chr> "23", "21", "19", "22", "13", "13", "8", "13", "18", "13", "13"…
$ O3    <chr> "33", "35", "36", "31", "44", "46", "14", "25", "46", "46", "61…
$ CO    <chr> "0.1", "0.1", "0.1", "0.1", "0.1", "0.1", "0.1", "0.1", "0.1", …
$ NO2   <chr> "34", "32", "27", "35", "22", "19", "52", "41", "23", "26", "13…
```

Figura 4. Muestra del dataset inicial.

Dentro del dataset se cambian los valores “N” (que son aquellos en los que la estación no tiene observaciones) por un valor “NA” que es la forma que tiene R de designar a los valores nulos.

```
# Establecer todos los datos a numerico
datos_castillo$Hora <- as.numeric(datos_castillo$Hora)
datos_castillo$SO2 <- as.numeric(datos_castillo$SO2)
datos_castillo$NO <- as.numeric(datos_castillo$NO)
datos_castillo$PM1.0 <- as.numeric(datos_castillo$PM1.0)
datos_castillo$PM2.5 <- as.numeric(datos_castillo$PM2.5)
datos_castillo$O3 <- as.numeric(datos_castillo$O3)
datos_castillo$CO <- as.numeric(datos_castillo$CO)
datos_castillo$NO2 <- as.numeric(datos_castillo$NO2)
```

Figura 5. Código de cambio de N por NA.

Como ya se ha destacado, también se busca la comodidad para trabajar en operaciones futuras, esto se refiere a cambiar las variables de tipo "char" (es decir, todas las que hacen referencia a sustancias en el aire) a un tipo "numeric".

```
# Establecer NA de R
datos_castillo$NO[datos_castillo$NO == "N"] <- NA
datos_castillo$NO2[datos_castillo$NO2 == "N"] <- NA
datos_castillo$PM1.0[datos_castillo$PM1.0 == "N"] <- NA
datos_castillo$PM2.5[datos_castillo$PM2.5 == "N"] <- NA
datos_castillo$O3[datos_castillo$O3 == "N"] <- NA
datos_castillo$CO[datos_castillo$CO == "N"] <- NA
datos_castillo$NO2[datos_castillo$NO2 == "N"] <- NA
```

Figura 6. Código de cambio a numeric.

7.2. Eliminación de NA en los predictores.

Para poder ejecutar un modelo predictivo es necesario que en los valores tomados para el predictor elegido tengan el formato correcto para todas las observaciones, es decir, que no tengan valores ausentes, para ello se ha ejecutado el siguiente código:

```
####################################
#Para eliminar las filas con valor nulo en la clase
datos_castillo <- datos_castillo[!is.na(datos_castillo$NO2),]
datos_castillo <- datos_castillo[!is.na(datos_castillo$NO),]
datos_castillo <- datos_castillo[!is.na(datos_castillo$PM1.0),]
datos_castillo <- datos_castillo[!is.na(datos_castillo$PM2.5),]
datos_castillo <- datos_castillo[!is.na(datos_castillo$O3),]
datos_castillo <- datos_castillo[!is.na(datos_castillo$CO),]
datos_castillo <- datos_castillo[!is.na(datos_castillo$SO2),]
```

Figura 7. Código de eliminación de valores nulos.

De esta manera, los valores nulos del dataset pasan a ser cero.

Capítulo 8. Definición de una métrica de éxito.

Dentro de R y Caret se usa RMSE como métrica para evaluar un modelo de Machine Learning, aunque hay otras muchas alternativas, se va a usar este tipo de métrica para definir el éxito del modelo.

RMSE (*Root Mean Squared Error*) [4] se trata de la desviación media de las predicciones de los datos observados. Es decir, que el RMSE es siempre no negativo, y un valor de 0 indicaría un ajuste perfecto a los datos. En general, una RECM más baja es mejor que una más alta. Sin embargo, las comparaciones entre diferentes tipos de datos no serían válidas porque la medida depende de la escala de los números utilizados. Es muy útil tener una idea de cómo de bien está funcionando el algoritmo usado en función de los resultados para así proceder a mejorar el modelo en busca de mejores resultados.

Capítulo 9. Métodos de entrenamiento

Los métodos de entrenamiento o *resampling*, son estrategias que permiten estimar la capacidad predictiva de los modelos, haciendo uso sólo de los datos de entrenamiento.

La idea básica es la de ajustar el modelo empleando un subconjunto de las observaciones de la parte de entrenamiento y evaluar el modelo con las observaciones restantes. Este proceso se repite muchas veces y los nuevos resultados generados se agregan para poder promediarlos. Lo que

diferencia a cada uno de estos métodos de los otros suele ser la manera en que generan los subconjuntos de entrenamiento.

En este caso, se ha decidido utilizar tres de ellos: *Repeated k-Fold-Cross-Validation, Bootstrapping* y *Out of bag*.

9.1. Repeated k-Fold-Cross-Validation (repeated CV)

RepeatedCV [5] consiste en repartir las observaciones de entrenamientos en *k* conjuntos (*fold*) del mismo tamaño. Luego, se ajusta el modelo con todas las observaciones excepto las del primer conjunto, para poder así evaluar el modelo prediciendo las observaciones del conjunto que se ha excluido. Esto dará lugar a la primera métrica. Este paso se repite k veces, excluyendo cada vez un conjunto distinto, generando al final *k* valores de la métrica, dando lugar así a la estimación final de validación.

Por último, todo este proceso se vuelve a repetir por completo *n* veces. De manera que al final se tenga una mejor estimación final de validación, pues se han podido generar mejores resultados gracias a que se ha podido compensar las posibles desviaciones que pudieran surgir.

9.2. Bootstrapping

Boot [6] consiste en obtener una muestra (*muestra bootstrap*) a partir de la muestra original, siguiendo el método de muestreo aleatorio con reposición (*resampling with replacement*). Esto significa que, después de haber extraído una observación, ésta se vuelve a poner a disposición para las siguientes extracciones. Como resultado, algunas de las observaciones aparecerán cantidad de veces en la muestra *bootstrap*, mientras que otras no aparecerán ninguna vez. Estas últimas son denominadas out-of-bag (*OOB*), con las que se evaluará el modelo ajustado con la *muestra bootstrap*.

El *bootstrapping* sigue los pasos que se nombran a continuación:

1. Obtener una nueva muestra *(muestra bootstrap*) del mismo tamaño que la muestra original (*muestreo aleatorio con reposición*).
2. Ajustar el modelo empleando la nueva *muestra bootstrap*.

3. Se calcula el error del modelo a partir de las observaciones que no se han incluido en la *muestra bootstrap*.
4. Repetición del proceso *n* veces y calcular la media de los errores.
5. Ajustar modelo final utilizando las observaciones de entrenamiento originales.

9.3. Out of bag (*OOB*)

OOB [7] es un método que mide el error de predicción de los random forests, los árboles de decisión y otros modelos de Machine Learning que utilizan el *Bootstrapping o Bagging*.

Es decir, *OOB* es el error de predicción medio en cada muestra de entrenamiento, calculado utilizando los árboles que no tenían dicha muestra de entrenamiento en su muestra original.

Se puede estimar la mejora del rendimiento de la predicción mediante la evaluación de las predicciones sobre aquellas observaciones que no se utilizaron en la construcción de la *muestra bootstrap* siguiente.

Capítulo 10. Creación de modelos predictivos

Después de haber procesado todos los datos, se debe realizar un modelo predictivo basado en un algoritmo de Machine Learning que nos permita conocer qué patrones siguen los datos y así predecir aquellos datos que el modelo no conoce.

Una vez que se ha dividido el dataset en la parte de entrenamiento y en la parte de test, se debe proceder a la fase de entrenamiento. Para ello, se hará uso de la función ***train()***, con la que se podrá especificar qué algoritmo y métrica se van a utilizar y la forma en que se lleva a cabo dicho entrenamiento.

10.1. Consideraciones previas para la implementación de los modelos.

Para proceder a la creación de un modelo predictivo con un algoritmo, debe definirse, antes de entrenar el modelo, el método de

entrenamiento que se va a utilizar. Esta definición se realizará utilizando la función ***trainControl()***, que viene dentro del paquete *Caret*.

En primer lugar, se hará uso del método de entrenamiento *RepeatedCV*, explicado en el apartado anterior. Los argumentos que se le pasan a la función ***trainControl()*** en este caso son:

- El método que se va a utilizar.
- El número de conjuntos (*fold*).
- El número de repeticiones *n*.

```
tc <- trainControl(method="repeatedcv",
                   number=8,
                   repeats=4,
                   verboseIter = FALSE,
                   allowParallel = TRUE)
```

Figura 8. Código de trainControl para RepeatedCV.

En este caso se ha detallado el método de *RepeatedCV*, pero a lo largo de todo el proyecto se trabajará con otros métodos, nombrados previamente, en los que simplemente habría que modificar el nombre del método en la llamada a la función ***trainControl()***. Y en el caso pertinente, modificar los parámetros propios de cada método.

Una vez definido el método de entrenamiento a utilizar, se puede empezar a entrenar el modelo. Para ello, es necesario indicar los argumentos que necesitará la función ***train()***:

- La clase a predecir.
- Los datos de entrenamiento.
- El método a utilizar.
- La función ***trainControl()*** creada anteriormente.
- importance=true
- La métrica a usar será RMSE.

```
modelo <- train(NO2~.,
                data=datos_train ,
                method='rf',
                trControl=tc,
                tuneLength=6,
                importance=TRUE,
                metric='RMSE')
modelo
```

Figura 9. Código de entrenamiento.

Esta función ***train()*** se usa en todos los modelos de igual manera, sencillamente se cambia el algoritmo a usar para cada modelo (RandomForest, SVM-Lineal, Regresión, etc.).

Acto seguido, tras la creación de un modelo entrenado sobre los datos preparados para dicho entrenamiento, se procede a la evaluación del modelo respecto a los datos de testeo. Ésto también es compartido por todos los modelos sin tener en cuenta los métodos, por lo que es el mismo código.

```
set.seed(123)
modelo_pred <- predict(modelo,
                       newdata = datos_test )
postResample(modelo_pred,
             datos_test$NO2)
```

Figura 10. Código de evaluación.

Con esto se consigue ver cómo de bien puede el modelo creado predecir datos partiendo de unos datos de entrenamiento.

Cabe destacar también que podemos conocer los resultados de rendimiento promedio de los mejores parámetros ajustados, haciendo uso de la función ***getTrainPerf()***.

10.2. Algoritmo ***Random Forest***.

Si bien es cierto que existen muchos algoritmos de Machine Learning que pueden dar lugar a un buen modelo, en este caso se ha decidido utilizar, en primer lugar, el algoritmo denominado Random Forest.

Random Forest [8] es un algoritmo predictivo que usa la técnica de Bagging para combinar diferentes árboles, donde cada uno es construido con observaciones y variables aleatorias. Este algoritmo sigue los siguientes pasos:

1. Selecciona individuos al azar para crear diferentes set de datos.
2. Crea un árbol de decisión con cada set de datos, obteniendo diferentes árboles.
3. Al crear los árboles, se eligen variables al azar en cada nodo del árbol, haciendo crecer el árbol en profundidad.
4. Predice los nuevos datos usando el "voto mayoritario", donde saldrá el resultado de la media de la predicción de todos los árboles.

10.2.1. Resultado para **RepeatedCV**.

Después de haber ejecutado la función ***train()*** se deben evaluar los resultados. Para ello, aparte de hacer hincapié en la información del modelo que se ha generado, se realizará una gráfica que explique más detalladamente la evolución del entrenamiento.

```
Random Forest

6811 samples
   7 predictor

No pre-processing
Resampling: Cross-Validated (8 fold, repeated 4 times)
Summary of sample sizes: 5960, 5960, 5961, 5959, 5959, 5958, ...
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
  2     4.515748  0.6734158  3.114337
  3     4.511609  0.6681854  3.089219
  4     4.543505  0.6623907  3.103661
  5     4.563794  0.6590809  3.113343
  6     4.583344  0.6560488  3.126219
  7     4.597303  0.6538754  3.134312

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 3.
```

Figura 11. Resultado del entrenamiento para el modelo de RandomForest con RepeatedCV.

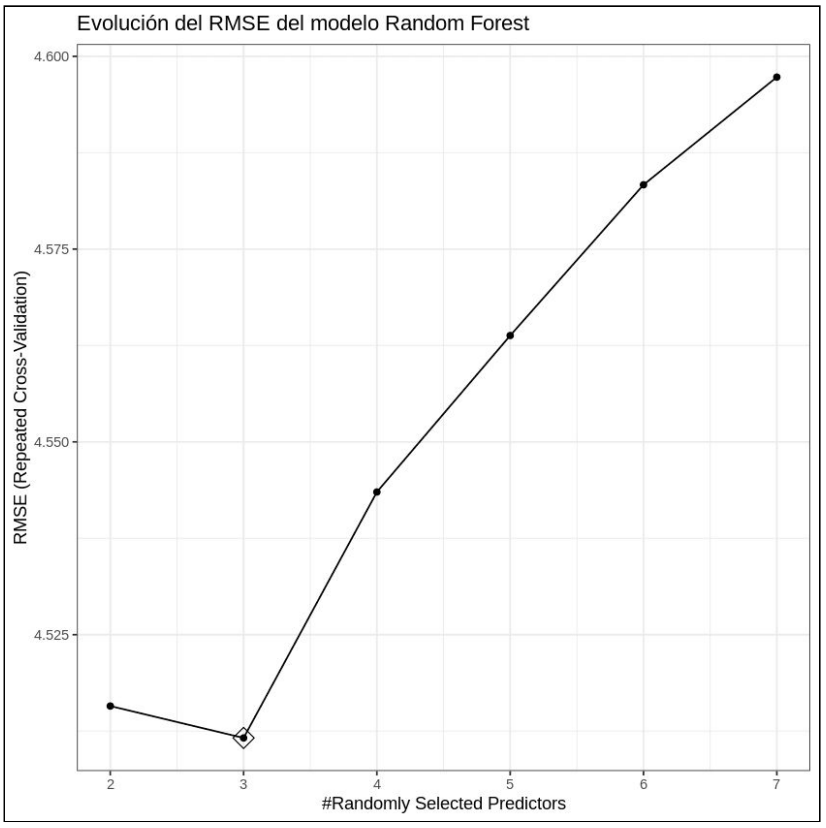

Figura 12. Resultado de RandomForest evaluado con RMSE.

Tanto en el resultado del modelo como en la gráfica, se puede ver que con el parámetro de ajuste o *mtry = 3,* se ha conseguido el valor del RMSE más pequeño, dando lugar así al modelo más óptimo.

Por otro lado, aquí se puede observar cuáles han sido los valores más pequeños de cada una de las métricas que se muestran.

```
getTrainPerf(modelo)
```

A data.frame: 1 × 4

TrainRMSE	TrainRsquared	TrainMAE	method
<dbl>	<dbl>	<dbl>	<chr>
4.511609	0.6681854	3.089219	rf

Figura 13. getTrainPerf para el modelo sobre RandomForest.

Por último, después de que el modelo haya sido ajustado, se procederá a evaluar el modelo, creando un objeto de predicciones realizadas con el objeto devuelto por ***train()*** y los datos del test.

Esta evaluación se llevará a cabo utilizando la función ***postResample()*** que se encargará de calcular el RMSE de las predicciones generadas y del predictor que se ha escogido como clase. Según las predicciones, se conseguirá obtener un valor de RMSE aún más pequeño que el generado en la fase de entrenamiento.

Figura 14. Evaluación del modelo (con REPEATED-CV) con la predicción sobre los datos de test.

10.2.2. Resultado para **oob**.

A la hora de utilizar este método de entrenamiento, se han observado cambios en los resultados con respecto a los generados con el método de entrenamiento repeatedCV.

En primer lugar, el valor más pequeño de RMSE obtenido se ha conseguido con un mtry = 2.

```
Random Forest

6811 samples
   7 predictor

No pre-processing
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared
  2     4.500362  0.6685671
  3     4.502866  0.6681980
  4     4.527538  0.6645522
  5     4.556288  0.6602783
  6     4.576470  0.6572621
  7     4.584086  0.6561204

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 2.
```

Figura 15. Resultado del entrenamiento para el modelo de RandomForest con oob.

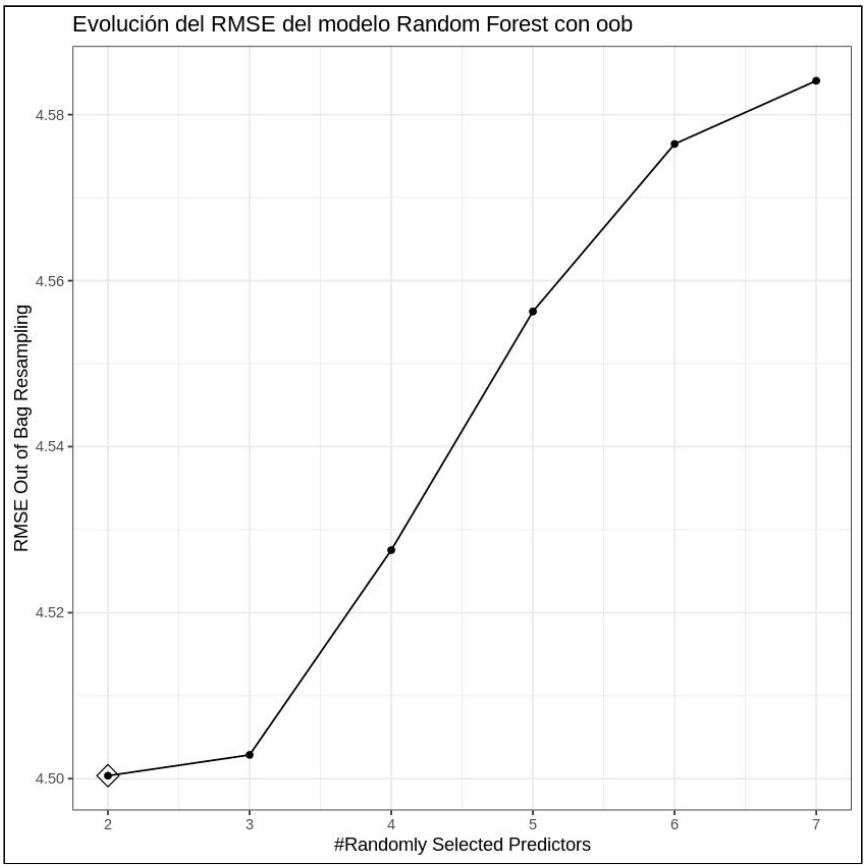

Figura 16. Resultado de RandomForest evaluado con RMSE y método oob.

Al calcular el valor del rendimiento promedio de los predictores, se puede ver que, haciendo uso del método de entrenamiento *OOB*, se ha conseguido un valor de RMSE más pequeño que el conseguido en el apartado anterior, lo cual es muy bueno, ya que se desea conseguir el valor de RMSE lo más próximo a 0 posible.

```
getTrainPerf(modelo)
```

A data.frame: 1 × 4

TrainRMSE	TrainRsquared	TrainMAE	method
<dbl>	<dbl>	<dbl>	<chr>
4.540839	0.6564286	3.146157	rf

Figura 17. getTrainPerf para el modelo sobre RandomForest.

Por otro lado, a la hora de evaluar el modelo haciendo uso de las predicciones generadas, se ha contemplado un decremento del valor del RMSE, que no llega a ser menor que el conseguido en el método *repeatedCV*.

```
RMSE
        4.36003292844077
Rsquared
        0.679588042850279
MAE
        3.11169102801696
```

Figura 18. Evaluación del modelo (con oob) con la predicción sobre los datos de test.

10.2.3. Resultado para **boot.**

Una vez realizado el ajuste del modelo, se procederá a evaluar los resultados. En este caso, el valor más pequeño del RMSE ha sido generado con un mtry = 2. Sin embargo, el valor generado con el método de *Bootstrapping* es el más grande comparado con los otros dos métodos de entrenamiento utilizados

```
Random Forest

6811 samples
   7 predictor

No pre-processing
Resampling: Bootstrapped (6 reps)
Summary of sample sizes: 6811, 6811, 6811, 6811, 6811, 6811, ...
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
  2     4.540839  0.6564286  3.146157
  4     4.604423  0.6419075  3.161031
  7     4.708091  0.6263386  3.223310

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 2.
```

Figura 20. Resultado del entrenamiento para el modelo de RandomForest con bootstrapping.

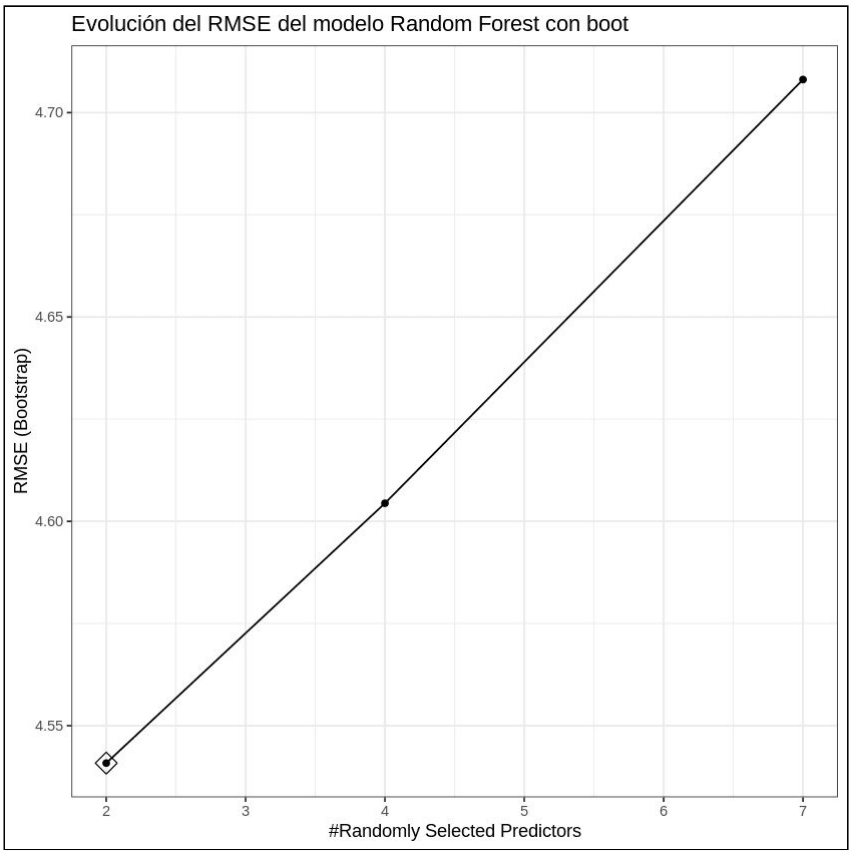

Figura 21. Resultado de RandomForest evaluado con RMSE y método bootstrapping.

En la siguiente figura se podrá observar los mejores resultados obtenidos en el modelo.

```
getTrainPerf(modelo)
```

A data.frame: 1 × 4

TrainRMSE	TrainRsquared	TrainMAE	method
<dbl>	<dbl>	<dbl>	<chr>
4.540839	0.6564286	3.146157	rf

Figura 22. getTrainPerf para el modelo sobre RandomForest.

Después de haber realizado las predicciones pertinentes, se puede comprobar una disminución del valor del RMSE, el cual queda igual que el valor generado con el método *oob* y que sigue siendo mayor que el obtenido con el método *repeatedCV*.

Figura 23. Evaluación del modelo (con bootstrapping) con la predicción sobre los datos de test

10.3. Algoritmo ***SVM-Lineal***.

Las máquinas de vector soporte o SVM [9] son un conjunto de algoritmos de aprendizaje supervisado. Estas se fundamentan en el *Maximal Margin Classifier,* el cual es un clasificador capaz de dar una distancia asociada al límite de decisión de cada ejemplo, que se basa a su vez en la idea del hiperplano, que no es más que un subespacio plano.

Para comprender a fondo los principios de las SVM's es necesario tener conocimientos de álgebra lineal. La SVM busca un hiperplano que separe de forma óptima a los puntos de una clase de la otra, los cuales han podido ser previamente proyectados a un espacio de dimensionalidad superior.

Originariamente, este método fue desarrollado como método de clasificación. No obstante, se realizó una nueva versión para regresión, que consiste en realizar un mapeo de los datos de entrenamiento a un espacio de mayor dimensión a través de un mapeo no lineal, donde se puede llevar a cabo una regresión lineal.

10.3.1. Resultado para **boot.**

Para evaluar los resultados, se muestra el objeto que devuelve la función ***train()***. Al ser lineal, el parámetro de ajuste se pone por defecto

igual a 1, dando como resultado el valor más pequeño de RMSE que se puede generar.

```
Support Vector Machines with Linear Kernel

6811 samples
   7 predictor

No pre-processing
Resampling: Bootstrapped (5 reps)
Summary of sample sizes: 6811, 6811, 6811, 6811, 6811
Resampling results:

  RMSE      Rsquared   MAE
  5.994952  0.4350647  3.920991

Tuning parameter 'C' was held constant at a value of 1
```

Figura 24. Resultado del entrenamiento para el modelo de SVM con bootstrapping.

A data.frame: 1 × 4

TrainRMSE	TrainRsquared	TrainMAE	method
<dbl>	<dbl>	<dbl>	<chr>
5.994952	0.4350647	3.920991	svmLinear

Figura 25. getTrainPerf para el modelo sobre RandomForest.

Por otro lado, si se evalúan las predicciones, en este caso se obtiene un valor de RMSE mayor que el obtenido en la fase de entrenamiento. Esto quiere decir que, con este modelo, no se conseguirá predecir resultados buenos y fiables.

```
RMSE
      6.02646785918962
Rsquared
      0.391255932949122
MAE
      3.98574978506953
```

Figura 26. Evaluación del modelo (con boot) con la predicción sobre los datos de test.

10.3.2. Resultado para **RepeatedCV**.

```
Support Vector Machines with Linear Kernel

6811 samples
   7 predictor

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 4 times)
Summary of sample sizes: 5450, 5448, 5447, 5449, 5450, 5449, ...
Resampling results:

  RMSE      Rsquared   MAE
  6.104706  0.4316359  3.949576

Tuning parameter 'C' was held constant at a value of 1
```

Figura 27. Resultado del entrenamiento para el modelo de SVM_Lineal con RepeatedCV.

A data.frame: 1 × 4

TrainRMSE	TrainRsquared	TrainMAE	method
<dbl>	<dbl>	<dbl>	<chr>
6.104706	0.4316359	3.949576	svmLinear

Figura 28. Evaluación del modelo (con boot) con la predicción sobre los datos de test.

Para comprobar que se ha obtenido un mejor resultado con el algoritmo SVM se ha decidido usar también el método de RepeatedCV, donde se puede observar que estableciendo una C constante, para el entrenamiento se tiene un valor mayor que al realizar la evaluación predictiva del modelo con los datos de testeo. Con esto se puede decir que el modelo es capaz de realizar de una manera adecuada y más o menos acertada, una predicción de datos.

Figura 29. Evaluación del modelo (con boot) con la predicción sobre los datos de test.

10.4. Algoritmo ***LM***.

La Regresión Lineal [10] consiste en generar un modelo de regresión para explicar la relación lineal existente entre dos variables: una de ellas dependiente y la otra, independiente (predictora). Para hallar dicha relación, se necesita la ordenada en el origen, la pendiente y el error aleatorio que representa la diferencia entre el valor ajustado por la recta y el valor real.

Normalmente, tanto la ordenada en el origen como la pendiente son valores desconocidos y, por tanto, se obtienen sus estimaciones a partir de una muestra. Dichas estimaciones toman aquellos valores que minimizan la suma de cuadrados residuales, dando lugar así a la recta que pasa más cerca de todos los puntos. Con dicha recta se puede explicar la relación entre las dos variables, generando de esta manera un modelo de regresión lineal.

10.4.1. Resultado para **boot**.

Usando el método de entrenamiento de *bootstrapping*, pero en este caso sobre el algoritmo de LM (Linear Model). En las figuras que se muestran a continuación, se puede observar que para el entrenamiento del modelo se ha obtenido un valor se métrica mayor que el de la evaluación con los datos de testeo, esto significa que el modelo tal y como se ha definido no es capaz de asegurar del todo una correcta predicción, pero al no ser un diferencia muy alta, se concluye que se comporta dentro de lo esperado.

```
Linear Regression

6811 samples
   7 predictor

No pre-processing
Resampling: Bootstrapped (6 reps)
Summary of sample sizes: 6811, 6811, 6811, 6811, 6811, 6811, ...
Resampling results:

  RMSE      Rsquared   MAE
  5.735445  0.4445653  4.11345

Tuning parameter 'intercept' was held constant at a value of TRUE
```

Figura 30. Resultado del entrenamiento para el modelo de LM con bootstrapping.

A data.frame: 1 × 4

TrainRMSE	TrainRsquared	TrainMAE	method
<dbl>	<dbl>	<dbl>	<chr>
5.735445	0.4445653	4.11345	lm

Figura 31. Evaluación del modelo (con boot) con la predicción sobre los datos de test.

```
RMSE
        5.7987013305661
Rsquared
        0.423344064500396
MAE
        4.23322211965282
```

Figura 32. Evaluación del modelo (con boot) con la predicción sobre los datos de test.

10.4.2. Resultado para **RepeatedCV**.

```
Linear Regression

6811 samples
   7 predictor

No pre-processing
Resampling: Cross-Validated (8 fold, repeated 4 times)
Summary of sample sizes: 5960, 5960, 5961, 5959, 5959, 5958, ...
Resampling results:

  RMSE      Rsquared  MAE
  5.806321  0.448464  4.131756

Tuning parameter 'intercept' was held constant at a value of TRUE
```

Figura 33. Resultado del entrenamiento para el modelo de LM con RepeatCV.

A data.frame: 1 × 4

TrainRMSE	TrainRsquared	TrainMAE	method
<dbl>	<dbl>	<dbl>	<chr>
5.806321	0.448464	4.131756	lm

Figura 34. Evaluación del modelo (con boot) con la predicción sobre los datos de test.

Para este caso se ha decidido usar el método de RepeatedCV, que al igual que el apartado anterior, se realiza estableciendo el parámetro “intercept” a TRUE, que indica el valor medio esperado para Y cuando la X toma el valor 0. Sabiendo que la regresión lineal representa a los valores de Y mediante una combinación lineal de los valores de X, se procede a interpretar los resultados.

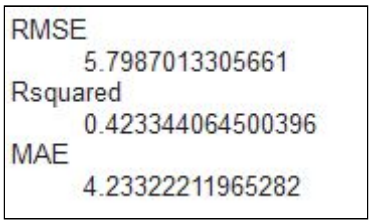

Figura 35. Evaluación del modelo (con boot) con la predicción sobre los datos de test.

Una vez tratados todos estos modelos, lo más importante es tratar de elegir el que mejor se adapta a las necesidades del proyecto de Machine Learning y el que ofrece el mejor resultado.

Capítulo 11. Epílogo.

Para finalizar, es el momento de sacar conclusiones teniendo en cuenta todos los modelos realizados y los resultados obtenidos en cada uno, con el fin de determinar el mejor de los modelos.

11.1. Resultados para los entrenamientos.

En primer lugar se ha de observar los resultados obtenidos en el entrenamiento de los modelos, observando la métrica de éxito elegida para el proyecto, que es el RMSE que cuanto más cercano se encuentre a 0 significa una mejor predicción. Teniendo en cuenta los algoritmos usados y a su vez los métodos de entrenamiento usados para cada uno, para ello se van a agrupar en función del algoritmo:

- RandomForest:

  Para el RandomForest se han usado 3 métodos distintos de entrenamiento, RepeatedCV, OOB y Boot, y el que ha ofrecido el mejor resultado ha sido el modelo en el que se ha usado OOB, proporcionando un RMSE igual a **4.50036**.

- SVM-Linear:

  En el caso de SVM solo se han usado 2 maneras distintas den entrenamiento, repitiendo los anteriores, el RepeatedCV y Boot. Se

ha obtenido el mejor resultado usando el parámetro C = 1 (es decir, constante), para el caso de Bootstrapping con un RMSE de **5.95547**.

- LM:

  Al igual que para el algoritmo de SVM, en LM se han utilizado Boot y RepeatedCV. En este caso, por tratarse de un modelo lineal se ha establecido que el valor de "intercept" sea TRUE (es decir, los valores medios de Y serán aquellos que tomen cuando X=0). Con ello, el mejor RMSE obtenido es **5.73541** con Bootstrapping.

Algoritmos	**Métodos de entrenamiento**	**Parámetros**	**Resultados: RMSE (entrenamiento)**
RandomForest	Repeated-CV	mytr = 3	4.51161
	OOB	mytr = 2	4.50036
	Boot	mytr = 2	4.95547
SVM-Lineal	Boot	c = 1 (cte)	5.95547
	Repeated-CV	c = 1 (cte)	6.1047
LM	Boot	intercept = TRUE	5.73541
	Repeated-CV	intercept = TRUE	5.80632

Tabla 1. Resultados de los algoritmos para el entrenamiento

Con los resultados observados, para la parte en entrenamiento de los modelos, se concluye que la combinación entre algoritmo y método de entrenamiento que proporciona la menor raíz del error cuadrático medio (RMSE) es: **RandomForest con OOB**.

11.2. Resultados de las evaluaciones en cuanto a predicción.

Al igual que para el entrenamiento de los modelos, ahora toca elegir el que proporciona el mejor resultado a la hora de predecir un subconjunto de datos partiendo de un subconjunto complementario de los mismos datos (tal y como se explicó con anterioridad). En este caso también se ha elegido como métrica el RMSE, que se buscará el que sea más cercano a 0. Se procede igual que anteriormente.

- RandomForest:

  En este caso y a diferencia del entrenamiento, se proporciona el mejor resultado por parte del método RepeatedCV con un RMSE de **4.347312**.

- SVM-Linear:

  Indiferentemente del método usado, se obtiene el mismo resultado, que es un RMSE de **6.02647**.

- LM:

  Para el modelo lineal, pasa como en el SVM, que ambos métodos de entrenamiento ofrecen el mismo RMSE de **5.7987**.

Algoritmos	Métodos de entrenamiento	Resultados: RMSE (predicción)
RandomForest	Repeated-CV	4.347312
	OOB	4.36003
	Boot	4.36003
SVM-Lineal	Boot	6.02647
	Repeated-CV	6.02647
LM	Boot	5.7987
	Repeated-CV	5.7987

Tabla 2. Resultados de los algoritmos para las predicciones.

Para los resultados obtenidos, se ha dado que en las predicciones, también ha sido el algoritmo de RandomForest el que ha ofrecido el mejor valor, pero en esta ocasión usando RepeatedCV. Por lo que se concluye que para la evaluación de las predicciones de modelos, el mejor ha sido: **RandomForest con RepeatedCV**.

El mejor rendimiento es el que da el RandomForest, pero estos resultados podrían mejorarse y tomar valores óptimos cambiando los parámetros de ejecución de los modelos. Por ejemplo, se podría cambiar la proporción de datos para entrenar y testear, o también cambiar el número de repeticiones usadas por los métodos de entrenamiento, incluso cambiar los propios métodos (usar *timeslice*, *LOOCV*, u otras derivaciones de *bootstrapping*).

11.3. Conclusión del proyecto.

La realización de este proyecto ha sido la primera experiencia de trabajo en Machine Learning, de desarrollo con el lenguaje R y uso del entorno colaborativo Google Colab de todos los integrantes del grupo, de esta forma ha resultado complejo avanzar en las primeras iteraciones del proyecto y seguir la planificación inicial del cronograma. Pero conforme se

iba profundizando y buscando información respecto a estos conceptos, se ha logrado obtener unas metas satisfactorias para el grupo, abarcando todos los requisitos del proyecto, desde el inicio en la elección de los datos y la estación metereológica, hasta el final con los resultados de los modelos, pasando por los análisis y procesado de datos.

Se ha adquirido conocimiento acerca de los distintos algoritmos de Machine Learning y los métodos de entrenamiento, siendo fructífero lo desarrollado en el entorno colaborativo, pudiendo forjar una idea superficial sobre lo que es trabajar con Machine Learning, porque este área de conocimiento abarca mucho más de lo acontecido en este proyecto.

Para finalizar, señalar que todo el código de la implementación del proyecto está a disposición de todos en Google Colab [11] y todos los documentos referentes a este proyecto se encuentra tanto en un repositorio de GitHub [12] como en una carpeta compartida de Google Drive.

Cualquier aportación y recomendación será bien recibida, mediante los correos electrónicos de los integrantes del grupo.

Capítulo 12. Referencias.

1. Introducción a Machine Learning: Enlace.

2. Cronograma del proyecto: Enlace.

3. Información sobre el Dióxido de Nitrógeno: Enlace.

4. Raíz del Error Cuadrático Medio: Enlace.

5. Método de entrenamiento RepeatedCV: Enlace.

6. Método de entrenamiento Boot: Enlace.

7. Método de entrenamiento OOB: Enlace.

8. Algoritmo de Random Forest: Enlace.

9. Algoritmo de SVM: Enlace.

10. Algoritmo de Regresión Lineal: Enlace.

11. Enlace al código dentro de la plataforma Google Colab: Enlace.

12. Enlace al repositorio en GitHub con el código del proyecto: Enlace.

Capítulo 12. Anexo.

Correos institucionales de los integrantes del grupo:

- Eduardo Ernesto Brito Sánchez (alu0100783315@ull.edu.es)
- Sergio Delgado López (alu0100893601@ull.edu.es)
- Pablo González González (alu0100887037@ull.edu.es)
- Alba Cruz Torres (alu0100889635@ull.edu.es)
- Jefry Izquierdo Hernández (alu0100996944@ull.edu.es)
- Daniel Abreu García (alu0100898385@ull.edu.es)