3. Modelización (Práctica)

Taller: calidad del aire

Minería de datos II

Curso 2019/2020

En esta práctica vamos a utilizar los datos de **calidad del aire** que hemos trabajado en clase para aplicar los modelos de minería de datos que conocemos.

Rercuerda:

> **El objetivo es predecir el nivel de `pm25` para el día siguiente**. Esto permitirá al ayuntamiento activar los protocolos anticontaminación con la suficiente antelación.

La entrega consistirá en un script R donde se lleven a cabo todos los pasos y un archivo .RDS con los RMSE resultantes de los distintos modelos. Se indicará claramente en los comentarios con qué pregunta se corresponde cada sección del código. Se valorará positivamente que el código esté lo más limpio y ordenado posible y el uso de comentarios relevantes.

Paso 1 (2 puntos)

1. Importa los datos de `train` y `test` que generamos en la preparación de la modelización. Para evitar problemas que hayáis podido tener, en el aula virtual estań subidos los conjuntos de datos `train.RDS` y `test.RDS` que hemos generado en clase.
2. La primera observación de `train` no tendrá sentido ya que **todas las variables `_lag` aparecerán como `NA`**. Elimina esta primera observación de `train`.
3. Elimina la columna fecha de `train` y `test`.

Paso 2 (1.5 puntos)

En el segundo pdf puedes ver que la variable más correlacionada con `pm25` era ella misma en el instante anterior, es decir, `pm25_lag`.

1. Representa en un **diagrama de dispersión** (recuerda que en ggplot tienes que usar `geom_point`) la relación entre ambas variables. Como es habitual, en el eje x representa la variable predictora `pm25_lag` y el eje y, la variable objetivo `pm25`.

En muchos proyectos, antes de abordar la modelización con modelos complejos, se comienza realizando una modelización muy simple que se suele denominar *baseline* y que nos permite compararla con otros modelos más complejos que podamos utilizar así como hacernos la pregunta de si vale la pena ese esfuerzo.

2. Genera un modelo denominado `mod_baseline` que sea una regresión lineal (utiliza la función `lm`) que prediga `pm25` utilizando solamente `pm25_lag`. Genera la predicción de test en un objeto llamado `pred_baseline`, calcula su RMSE y guárdalo en una variable llamada `rmse_baseline` (puedes utilizar `source` para importar la función `rmse` que hemos creado en clase).

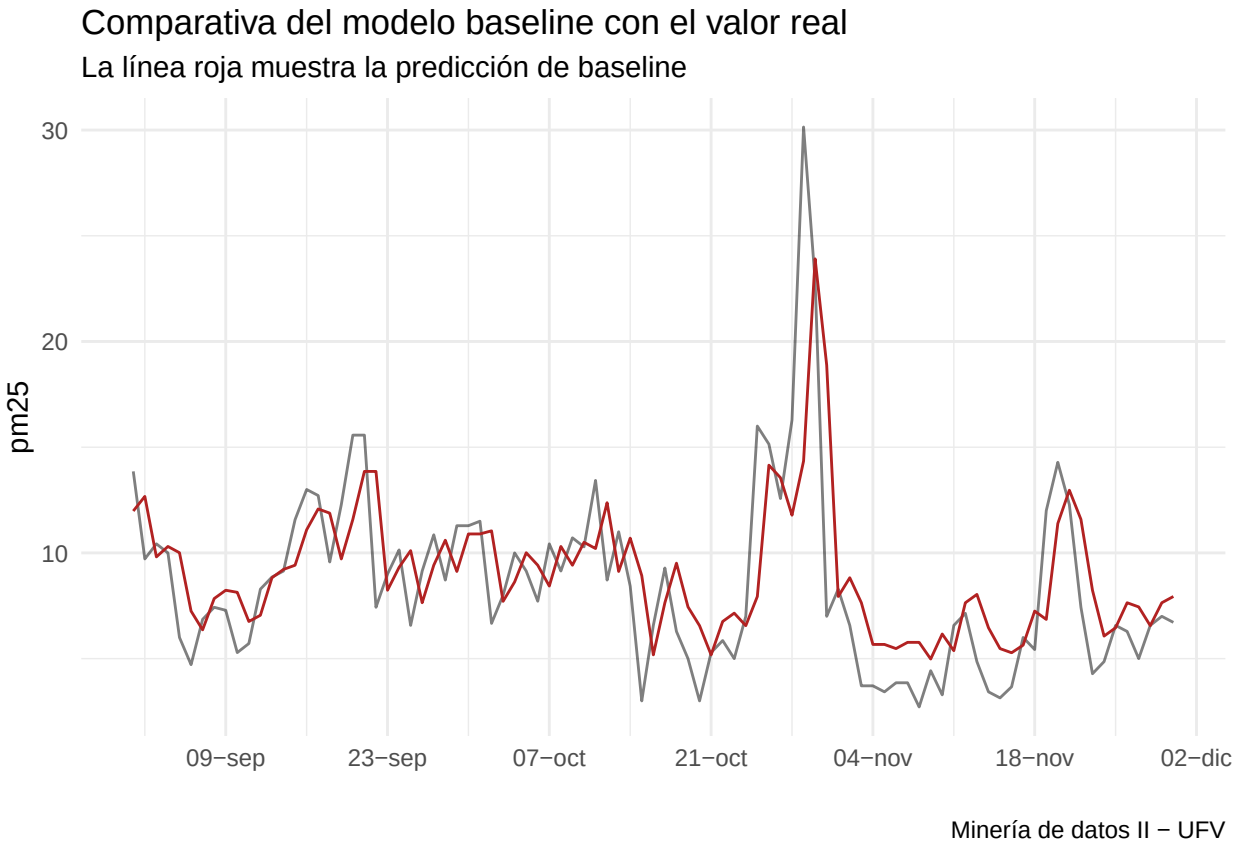
Paso 3 (3 puntos)

1. Aplica un modelo de `bagging` sobre `train`. Prueba valores 50, 100, 150, 200 para el número de iteraciones. Calcula el rmse en test para cada modelo de bagging y quédate con el mejor modelo. Llama a ese modelo `mod_bagging`, a su predicción `pred_bagging` y a su RMSE `rmse_bagging`.
2. Aplica un modelo de random forest denominado `mod_rf`. Genera valores de `ntree` de 100 a 500 de 100 en 100. Genera valores de `mtry` de 2 al número de columnas de train de 3 en 3. Quédate con el mejor modelo y, de forma similar al apartado anterior, utiliza la nomenclatura `mod_rf`, `pred_rf` y `rmse_rf`.
3. Aplica un modelo de boosting denominado `mod_boost`. Utilizando los parámetros `eta`, `max_depth`, `subsample`, `colsample_bytree`, genera un mínimo de 50 combinaciones (en total) y quédate con la mejor. De nuevo, utiliza la nomenclatura `pred_boost` y `rmse_boost`. Utiliza un `early stopping` de 50 y `nrounds` de 1000

Paso 4 (1.5 puntos)

Compara los *RMSE* de los 4 modelos que hemos generado (*baseline*, bagging, random forest y boosting). ¿Cuál predice mejor? *(Contestar a la pregunta en un comentario)*

En este caso, al tratarse de unos datos temporales, podemos representar fácilmente la información. Genera un gráfico *similar* al siguiente en el que se compare el valor real en test con el predicho por **cada modelo**, es decir, debes hacer **un gráfico para baseline, bagging, random forest y boosting**.



Nota: en el gráfico anterior se ha modificado la apariencia básica de ggplot2. Se dará puntuación extra si se hace el esfuerzo de hacer un gráfico lo más parecido posible.

Paso 5 (2 puntos)

Vamos a enriquecer el conjunto de datos para ver si podemos mejorar la predicción.

1. Descarga del aula virtual el conjunto de datos `dias_laborables.RDS`. Importa este conjunto de datos (recuerda que tiene un formato `.RDS`). Este `data.frame` contiene una variable `fecha` y otra variable `laborable` que toma valor 1 si es un día laborable o 0 si es un festivo, sábado o domingo.
2. Añade esta información a `train` y `test` utilizando la función `left_join()` de forma adecuada.
3. Reentrena los modelos `baseline`, `bagging`, `random forest` y `boosting` con esta nueva información (para el modelo baseline utiliza `pm25 ~ pm25_lag + laborable`). Llama a cada modelo igual que hicimos antes pero terminado en 2 (por ejemplo, `mod_boost2`, `pred_boost2`, `rmse_boost2`).+
4. Calcula y representa gráficamente la importancia de las variables del modelo de boosting. ¿Es relevante que un día sea laborable para el modelo?
5. ¿Se mejoran los valores de $RMSE$ conseguidos anteriormente? Para contestar a la pregunta genera un data frame con una columna `modelo` y otra `rmse` donde se recojan los valores de forma similar a la tabla siguiente (los valores de la tabla del ejemplo no tienen por qué ser reales). Guarda este data frame como un archivo `.RDS` utilizando la función `writeRDS()`.

```
## # A tibble: 8 x 2
##   modelo          rmse
##   <chr>          <dbl>
## 1 baseline        3.21
## 2 baseline2       3.20
## 3 bagging         2.92
## 4 bagging2        2.88
## 5 random forest   2.94
## 6 random forest2  2.91
## 7 boosting        2.91
## 8 boosting2       2.89
```