Análisis y modelado de datos

Pablo Domínguez

2022-06-23

Planteamiento del problema a abordar

Nos encontramos con un conjunto de datos obtenidos a partir de mediciones meteorológicas realizadas por el gobierno de Australia[1]. Estos datos, recogidos en distintas localidades, se han capturado realizando mediciones diarias de temperatura, lluvia, evaporación, sol, viento, humedad etc.

En la referencia mencionada advierten que el control de calidad aplicado a la captura de estos datos ha sido limitado, por lo que es posible que existan imprecisiones debidas a datos faltantes, valores acumulados tras varios datos faltantes o errores de varios tipos. Es por este motivo que empezaremos nuestro estudio realizando una revisión de la calidad y estructura del dato. Tras este proceso, construiremos una serie de variables que transformarán el problema y la estructura de datos para que puedan aplicarse los modelos de clasificación supervisada planteados.

Partiendo de la base de datos procesada, la segmentaremos para aplicar varios modelados diferentes por zonas (siguiendo cierto criterio). Finalmente, compararemos los modelos, los ensamblaremos y presentaremos unos resultados de la precisión del modelo final.

Con esta aplicación práctica de los modelos teóricos abordados en el capítulo anterior buscamos reflejar la capacidad de herramientas matemáticas abstractas a la hora de resolver situaciones que pueden tener un gran beneficio en varios ámbitos, tales como sociales, económicos o medioambientales.

Origen de los datos y variable objetivo

El buró de metereología australiano coordina una serie de estaciones metereológicas locales repartidas a lo largo del territorio. De esta manera, recopila y reporta datos sobre mediciones meteorológicas. En nuestro caso, tenemos información de **(numero)** ciudades repartidos a lo largo de **cantidad** años.

```
# Buscar manera alternativa (representable) de evaluar los tipos de dato.
str(db)
```

[1] Notes about Daily Weather Observations - Australian Goverment (2004)

Table 1: Muestra de los datos[note]

Date	Location	MinTemp	MaxTemp	Rainfall	—	Temp9am	Temp3pm	RainToday	RainTomorrow
2008-12-01	Albury	13.4	22.9	0.6	—	16.9	21.8	No	No
2008-12-02	Albury	7.4	25.1	0.0	—	17.2	24.3	No	No
2008-12-03	Albury	12.9	25.7	0.0	—	21.0	23.2	No	No
2008-12-04	Albury	9.2	28.0	0.0	—	18.1	26.5	No	No
2008-12-05	Albury	17.5	32.3	1.0	—	17.8	29.7	No	No
2008-12-06	Albury	14.6	29.7	0.2	—	20.6	28.9	No	No

```
## 'data.frame':    145460 obs. of  23 variables:
##  $ Date         : Date, format: "2008-12-01" "2008-12-02" ...
##  $ Location     : Factor w/ 49 levels "Adelaide","Albany",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ MinTemp      : num  13.4 7.4 12.9 9.2 17.5 14.6 14.3 7.7 9.7 13.1 ...
##  $ MaxTemp      : num  22.9 25.1 25.7 28 32.3 29.7 25 26.7 31.9 30.1 ...
##  $ Rainfall     : num  0.6 0 0 0 1 0.2 0 0 0 1.4 ...
##  $ Evaporation  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ Sunshine     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ WindGustDir  : Factor w/ 16 levels "E","ENE","ESE",..: 14 15 16 5 14 15 14 14 7 14 ...
##  $ WindGustSpeed: int  44 44 46 24 41 56 50 35 80 28 ...
##  $ WindDir9am   : Factor w/ 16 levels "E","ENE","ESE",..: 14 7 14 10 2 14 13 11 10 9 ...
##  $ WindDir3pm   : Factor w/ 16 levels "E","ENE","ESE",..: 15 16 16 1 8 14 14 14 8 11 ...
##  $ WindSpeed9am : int  20 4 19 11 7 19 20 6 7 15 ...
##  $ WindSpeed3pm : int  24 22 26 9 20 24 24 17 28 11 ...
##  $ Humidity9am  : int  71 44 38 45 82 55 49 48 42 58 ...
##  $ Humidity3pm  : int  22 25 30 16 33 23 19 19 9 27 ...
##  $ Pressure9am  : num  1008 1011 1008 1018 1011 ...
##  $ Pressure3pm  : num  1007 1008 1009 1013 1006 ...
##  $ Cloud9am     : int  8 NA NA NA 7 NA 1 NA NA NA ...
##  $ Cloud3pm     : int  NA NA 2 NA 8 NA NA NA NA NA ...
##  $ Temp9am      : num  16.9 17.2 21 18.1 17.8 20.6 18.1 16.3 18.3 20.1 ...
##  $ Temp3pm      : num  21.8 24.3 23.2 26.5 29.7 28.9 24.6 25.5 30.2 28.2 ...
##  $ RainToday    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 2 ...
##  $ RainTomorrow : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 2 1 ...
```

Información de los datos

*renombrar columnas Hablamos sobre la brecha de fechas, sobre los datos faltantes y el "salto" cuando ya no hay datos faltantes: 2013-03-01. Filtramos por el último valor a partir del cual no hay datos faltantes, esto es, 2013-12-31

```
# Rango de a 2013-12-31 y menor a 2017-06-24
db <- db %>% filter(.,Date>as.Date("2013-12-31") & Date<=as.Date("2017-06-24"))

# Comprobamos datos limpios
dates_df <- date_range()
dates_df %>% View()
```

Hablar de cómo hemos asignado las zonas climáticas. Puto google maps y el mapa ese del aus gov

```
# Creamos vectores de zonas climáticas
zona1 <- c("Exmouth", "Dampier", "PortHedland", "Broome", "Derby", "Wyndham", "TimberCreek", "Katherine
zona2 <- c("Mackay", "Rockhampton", "Maryborough", "Brisbane", "CoffsHarbour","GoldCoast")
zona3 <- c("Goondiwindi", "Taroom","Charleville", "Longreach","Thargomindah","Birdsville","MountIsa","Al
zona4 <- c("Woomera","Yalgoo", "Wiluna", "KalgoorlieBoulder", "Norseman", "Merredin","Newdegate","Warbu
zona5 <- c("Geraldton","Perth","Witchcliffe","Bunbury","MargaretRiver","Esperance","Eucla","Ceduna","Po
zona6 <- c("Albany","Burra","Kingscote","KingstonSE","MountGambier","Horsham","Watsonia","Melbourne","La
zona7 <- c("Ballarat","Canberra","Bathurst","Devonport","Strahan","Launceston","Swansea","Hobart","Soutl
zona8 <- c("MountGinini")

zonas <- c(zona1,zona2,zona3,zona4,zona5,zona6,zona7,zona8)
zona_climatica <- c()
```

```r
# Asignamos cada ciudad a una zona climática
for(city in levels(db$Location)){
  if(city %in% zona1){
    zona_climatica <- c(zona_climatica,1)
  }
  else if(city %in% zona2){
    zona_climatica <- c(zona_climatica,2)
  }
  else if(city %in% zona3){
    zona_climatica <- c(zona_climatica,3)
  }
  else if(city %in% zona4){
    zona_climatica <- c(zona_climatica,4)
  }
  else if(city %in% zona5){
    zona_climatica <- c(zona_climatica,5)
  }
  else if(city %in% zona6){
    zona_climatica <- c(zona_climatica,6)
  }
  else if(city %in% zona7){
    zona_climatica <- c(zona_climatica,7)
  }
  else if(city %in% zona8){
    zona_climatica <- c(zona_climatica,8)
  }
  else { # Los lugares sin zona climática le asignamos el 0, para luego eliminar estas observaciones
    zona_climatica <- c(zona_climatica,0)
  }
}

# Creamos dataframe de zonas climáticas
zonas_climaticas <- data.frame(levels(db$Location),zona_climatica) %>% rename(Location=levels.db.Locati
zonas_climaticas %>% View()
```

Hemos eliminado localizaciones las cuales no ha sido posible determinar la zona climática a la que pertenecen. En concreto, se han eliminado *NorfolkIsland*, *Portland*, *Richmond*, *Walpole* y *Williamtown*. Mostramos observaciones por zona climática.

```r
# Añadimos las zonas climáticas

db <- left_join(zonas_climaticas,db, by="Location",strings)
db$Location <- db$Location %>% as.factor()

# Eliminamos la zona climática 0
db <- db %>% filter(., zona_climatica != 0)
db <- droplevels(db)
#db$zona_climatica <- db$zona_climatica %>% as.factor()

# Contamos valores totales por zona climática:
db %>% count(zona_climatica) %>% View()

# Contamos cantidad de ciudades por zona climática
```

```
db %>% count(zona_climatica,Location) %>%count(zona_climatica,name="n_locations") %>%  View()

# Comprobamos datos continuos
db %>% count(zona_climatica,Location) %>% View() # esto confirma que tenemos 1272 datos para cada ciuda
dates_df <- date_range()
dates_df %>% View()
```

Comprobar tipos

División y limpiado de los datos

```
db_zone1 <- db %>% filter(., zona_climatica == 1) %>% select(., -zona_climatica)
db_zone2 <- db %>% filter(., zona_climatica == 2) %>% select(., -zona_climatica)
db_zone3 <- db %>% filter(., zona_climatica == 3) %>% select(., -zona_climatica)
db_zone4 <- db %>% filter(., zona_climatica == 4) %>% select(., -zona_climatica)
db_zone5 <- db %>% filter(., zona_climatica == 5) %>% select(., -zona_climatica)
db_zone6 <- db %>% filter(., zona_climatica == 6) %>% select(., -zona_climatica)
db_zone7 <- db %>% filter(., zona_climatica == 7) %>% select(., -zona_climatica)
db_zone8 <- db %>% filter(., zona_climatica == 8) %>% select(., -zona_climatica)
# Creamos diccionario con los df para poder iterar
zonas <- list("zona1"=db_zone1,
              "zona2"=db_zone2,
              "zona3"=db_zone3,
              "zona4"=db_zone4,
              "zona5"=db_zone5,
              "zona6"=db_zone6,
              "zona7"=db_zone7,
              "zona8"=db_zone8)
print(nrow(db) == (nrow(db_zone1) + nrow(db_zone2) + nrow(db_zone3) + nrow(db_zone4) + nrow(db_zone5) +
```

```
## [1] TRUE
```

Tratamiento de los atributos: missing, outliers, normalización, balanceo de datos

```
# Crear función de comprobar missings
# counting missing values
get_missings <- function(){
  df_missings <- NULL;
  total_rows <- c()
  for(i in 1:length(zonas)){
    row <- zonas[[i]] %>% select(everything()) %>% summarise_all(funs(sum(is.na(.))))*100 / nrow(zonas[
    total_rows <- c(total_rows,nrow(zonas[[i]]))
    df_missings <- df_missings %>% rbind(.,row)
  }
  df_missings["Total obs"] <- total_rows
  rownames(df_missings) <- c("zona1",
                             "zona2",
                             "zona3",
                             "zona4",
```

```
                              "zona5",
                              "zona6",
                              "zona7",
                              "zona8")
  return(df_missings)

}
df_missings <- get_missings()
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
df_missings %>% View()

# Vamos a eliminar todas las variables que tienen el 100% de NA en la zona8, esto es,
# Evaporation, Sunshine, Pressure9am, Pressure3pm, Cloud9am, Cloud3pm, ya que hay indicativos de que es
```

```
for(i in 1:length(zonas)){
  zonas[[i]] <- zonas[[i]] %>% select(., -any_of(c("Evaporation", "Sunshine", "Pressure9am", "Pressure3
}
```

Metrics imputations is a way to fill NaN values with some special metrics that depend on your data: mean or median for example.

Mean value is the sum of a value in a series divided by a number of all values of series. It is one of the most used types of metrics in statistics. But why do we impute the NaN values with mean value? Mean has a very interesting property, it doesn't change if you add some more mean values to your series.

```
calc_mode <- function(x){

  # List the distinct / unique values
  distinct_values <- unique(x)

  # Count the occurrence of each distinct value
  distinct_tabulate <- tabulate(match(x, distinct_values))
  top <- which.max(distinct_tabulate)
  # Return the value with the highest occurrence
  mode <- distinct_values[top]
  if(is.na(mode)){
    top <- distinct_tabulate[distinct_tabulate!=distinct_tabulate[top]] %>% which.max()
    mode <- distinct_values[top]
  }
```

```
  return(mode)
}
```

```
# mutate missing values

columnas_enteras <- zonas[[1]][, unlist(lapply(zonas[[1]], is.integer), use.names = FALSE) ] %>% colname
columnas_numericas <- zonas[[1]][, unlist(lapply(zonas[[1]], is.numeric), use.names = FALSE) ] %>% colna
columnas_categoricas <- zonas[[1]][, unlist(lapply(zonas[[1]], is.factor), use.names = FALSE) ] %>% coln

# reemplazamos variables continuas por la media
for(i in 1:length(zonas)){
  zonas[[i]] <- zonas[[i]] %>% mutate_at(columnas_numericas, ~replace_na(.,mean(., na.rm = TRUE)))
}

# reemplazamos variables enteras por la media truncada
for(i in 1:length(zonas)){
  zonas[[i]] <- zonas[[i]] %>% mutate_at(columnas_enteras, ~replace_na(.,floor(mean(., na.rm = TRUE))))
}

# reemplazamos variables categóricas por la moda
for(i in 1:length(zonas)){
  zonas[[i]] <- zonas[[i]] %>% mutate_at(columnas_categoricas, ~replace_na(.,calc_mode(.)))
}

get_missings
```

```
## function(){
##   df_missings <- NULL;
##   total_rows <- c()
##   for(i in 1:length(zonas)){
##     row <- zonas[[i]] %>% select(everything()) %>% summarise_all(funs(sum(is.na(.))))*100 / nrow(zona
##     total_rows <- c(total_rows,nrow(zonas[[i]]))
##     df_missings <- df_missings %>% rbind(.,row)
##   }
##   df_missings["Total obs"] <- total_rows
##   rownames(df_missings) <- c("zona1",
##                              "zona2",
##                              "zona3",
##                              "zona4",
##                              "zona5",
##                              "zona6",
##                              "zona7",
##                              "zona8")
##   return(df_missings)
##
## }
## <bytecode: 0x562ba193a760>
```

```
df_missings %>% View()
```

Eliminación de variables y representación de los datos

Feature selection

As presented in[2]

Representación

Distribuciones (histogramas)

Comprobar la proporción de dias que llueve vs dias que no por zona (rain_ratio)

Creación de nuevas variables ventana

Modelado

Train-test split

Australian Goverment. 2004. "Notes about Daily Weather Observations." Bureau of Meteorology. http://www.bom.gov.au/climate/dwo/IDCJDW0000.pdf.

Guyon, I., and A. Elisseeff. 2003. "An Introduction to Variable and Feature Selection." *Journal of Machine Learning Research* 3 (March).

[2] An Introduction to Variable and Feature Selection - Guyon and Elisseeff (2003)