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 cada zona climática[2]. 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 49 ciudades repartida a lo largo de unos 8 años.

Destacar que hemos importado las variables tipo *string* como *factor*. A continuación, podemos comprobar el tipo de cada variable:

[1]Notes about Daily Weather Observations - Australian Goverment (2004)
[2]Desarolladas en el *National Construction Code* por el *Buró de Metereología del Gobierno de Australia*

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

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

```
## '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 ...
```

De esta manera, según se refleja en Australian Goverment (2004), el conjunto de datos con el que vamos a trabajar cuenta con las siguientes variables:

- *Date*: Fecha en la que se realizó la medición, en formato *AAAA-MM-DD*.
- *Location*: Localización donde se realizó la medición.
- *MinTemp*: Temperatura mínima alcanzada, medida en grados celsius[3].
- *MaxTemp*: Temperatura máxima alcanzada, medida en grados celsius[4].
- *Rainfall*: Cantidad total de precipitación, medida en milímetros[5].
- *Evaporation*: Evaporación en milímetros sobre un *tanque evaporimétrico clase "A"*[6].
- *Sunshine*: Tiempo en horas de alto nivel de luminosidad solar[7].
- *WindGustDir*: Dirección general del viento racheado a lo largo del día[8], tomando una de las 16 direcciones posibles del viento.
- *WindGustSpeed*: Velocidad global del viento racheado a lo largo del día[9], medido en kilómetros por hora.
- *WindDir9am*, *WindDir3pm*: Dirección promedio del viento en los 10 minutos previos a la hora indicada en cada variable.
- *WindSpeed9am*, *WindSpeed3pm*: Velocidad promedio del viento en los 10 minutos previos a la hora indicada en cada variable, medida en kilómetros por hora.

[3]En un rango de 24 horas desde las 9am.
[4]En un rango de 24 horas desde las 9am.
[5]En un rango de 24 horas desde las 9am.
[6]En un rango de 24 horas desde las 9am.
[7]En un rango de 24 horas desde las 12am.
[8]En un rango de 24 horas desde las 12am.
[9]En un rango de 24 horas desde las 12am.

- *Humidity9am*, *Humidity3pm*: Humedad relativa, medida en tanto porciento, a las horas indicadas en cada variable.
- *Pressure9am*, *Pressure3pm*: Presión atmosférica medida a nivel del mar a las horas indicadas en cada variable, usando el hectopascal como unidad de medida.
- *Cloud9am*, *Cloud3pm*: Fracción del cielo cubierta por nubes a las horas indicadas, medida en *octas de cielo*.
- *Temp9am*, *Temp3pm*: Temperatura en grados celsius medida a las horas indicadas.
- *RainToday*, *RainTomorrow*: Variables binarias indicando si ha habido o no lluvia ese día.

De este modo, *RainTomorrow* será nuestra variable objetivo en los clasificadores. Con ello, desarrollaremos estrategias buscando que los modelos puedan decidir lo mejor posible si, dadas las observaciones de un día e información agregada de días anteriores, lloverá o no al día siguiente.

Información de los datos y filtrado de datos

A continuación, buscamos comprobar si la serie temporal está completa y sin repetidos para cada valor de la variable *Location*. En la tabla mostrada a continuación, se muestran las variables conteniendo la siguiente información:

- *min_fec*, *max_fec*: Fecha donde comienzan y terminan los datos en la serie temporal.
- *obs*: Número de observaciones total en cada serie temporal.
- *rep*: Booleano que representa si existen fechas repetidas dentro de la serie temporal, esto es, *FALSE* indica ausencia de fechas repetidas.
- *n_diff_dates*: Cantidad de fechas faltantes dentro de la serie temporal.
- *top_diff_date*: Última fecha faltante en la serie temporal.
- *range_free*: Cantidad de datos sin fechas faltates medidos hacia atrás desde el último registro. Es decir, longitud de la serie temporal continua más reciente.
- *range*: Rango, medido en dias, de la serie temporal. Esto es, número de días entre *max_fec* y *min_fec*.

Como podemos observar en el análisis anterior, vemos que la localización *NorahHead* tiene el último hueco en su serie temporal el *2013-12-31*. Es por este motivo que vamos a filtrar a partir de este valor con el fin de asegurarnos que no existan fechas faltantes dentro de las series temporales. Se puede observar además que el resto de localizaciones tienen como último salto en su serie temporal la fecha *2013-02-28*, y que las localizaciones de *Katherine*, *Nhil* y *Uluru* comienzan su serie temporal el *2013-03-01*. Esto es, que cuando comenzó a haber registros en estastres localizaciones dejó de haber errores de grabado de datos en la serie temporal en el resto de localizaciones (salvo en *NorahHead*). Este suceso dentro del proceso de grabación de datos podría ser indicativo de una reestructuración de los mecanismos de captación de los datos climáticos por parte del buró de meteorología australiano.

A continuación, filtrando sobre la fecha mencionada se puede observar que ya no existen discontinuidades en ninguna serie temporal:

```
# 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()
```

Agrupación por zona climática

Con el objetivo de modelar con mayor precisión los datos, vamos a realizar una segmentación de la información asignando la zona climática[10] a la que pertenece cada localización. Estas zonas están descritas de la siguiente manera:

- *Zona 1*: Veranos húmedos y cálidos, con inviernos cálidos.
- *Zona 2*: Veranos húmedos y templados, con inviernos suaves.
- *Zona 3*: Veranos secos y cálidos, con inviernos cálidos.
- *Zona 4*: Veranos secos y cálidos, con inviernos frescos.
- *Zona 5*: Clima templado.
- *Zona 6*: Clima suave.
- *Zona 7*: Clima fresco.
- *Zona 8*: Clima alpino.

```
# 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()

# 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){
```

[10] Desarolladas en el *National Construction Code* por el *Buró de Metereología del Gobierno de Australia*

```
    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()
```

Una estrategia alternativa que planteamos como propuesta de mejora en este punto consistiría en realizar un clustering no supervisado, de manera que se segmenten los datos en 8 categorías no equitativas. A continuación, compararíamos esta categorización no supervisada contra la categorización por zona climática para comprobar la similitud de ambas asignaciones.

En cualquier caso, en el planteamiento que proponemos hemos eliminado aquellas localizaciones donde 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.

```
# 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()
```

División y limpiado de los datos

Ahora si, estamos en disposición de crear un conjunto de datos para cada zona climática, a los cuales someteremos a una limpieza y procesado de datos. Es importante aplicar este procedimiento por separado para cada zona climática ya que imputaremos datos faltantes entre otras técnicas, para lo cual nos interesa que la información con la que sustituimos estos valores sea coherente con la subdivisión del conjunto de datos original.

```
db_zone1 <- db %>% filter(., zona_climatica == 1) %>% select(., -zona_climatica)
db_zone1 <- droplevels(db_zone1)
db_zone2 <- db %>% filter(., zona_climatica == 2) %>% select(., -zona_climatica)
```

```
db_zone2 <- droplevels(db_zone2)
db_zone3 <- db %>% filter(., zona_climatica == 3) %>% select(., -zona_climatica)
db_zone3 <- droplevels(db_zone3)
db_zone4 <- db %>% filter(., zona_climatica == 4) %>% select(., -zona_climatica)
db_zone4 <- droplevels(db_zone4)
db_zone5 <- db %>% filter(., zona_climatica == 5) %>% select(., -zona_climatica)
db_zone5 <- droplevels(db_zone5)
db_zone6 <- db %>% filter(., zona_climatica == 6) %>% select(., -zona_climatica)
db_zone6 <- droplevels(db_zone6)
db_zone7 <- db %>% filter(., zona_climatica == 7) %>% select(., -zona_climatica)
db_zone7 <- droplevels(db_zone7)
db_zone8 <- db %>% filter(., zona_climatica == 8) %>% select(., -zona_climatica)
db_zone8 <- droplevels(db_zone8)
# 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
```

Tenemos por tanto el conjunto de datos segmentado en 8 zonas disjuntas, que abordaremos y modelaremos en paralelo.

Calidad del dato

A continuación, nos disponemos a tratar los aspectos relativos a la calidad del dato. Esto es, tratamiento de valores faltates o *missings*, valores atípicos o *outliers*, normalización de los datos y balanceo de los datos.

Missings En la tabla que se muestra abajo se refleja el porcentaje en tanto por ciento de valores faltantes en cada variable por zona climática.

```
# 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()
```

Podemos observar que existen múltiples variables que tienen un *100%* de valores faltantes para la zona climática 8. Además, estas mismas variables una gran cantidad de valores faltantes en el resto de zonas climáticas, por lo que prescindiremos de ellas por su baja calidad del dato. Las variables de las que prescindiremos aplicando este criterio son *Evaporation*, *Sunshine*, *Pressure9am*, *Pressure3pm*, *Cloud9am* y *Cloud3pm*.

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

Una vez eliminadas dichas variables, resta por tratar el resto de columnas que presentan valores faltantes. Vemos que, a lo más, hay un *15%* de valores faltantes para la variable *Humidity3pm* en la zona climática 1, mientras que el resto de variables tiene menos de un *10%* de valores faltantes.

Con el objetivo de presentar a los modelos datos limpios con los que puedan trabajar, vamos a imputar de diferentes maneras los datos faltantes según el tipo de dato de cada columna: - En las columnas numéricas (no enteras), sustuiremos en cada variable los valores faltantes por el valor promedio de dicha variable en cada zona. - En las columnas enteras, aplicaremos un tratamiento similar sustituyendo los valores faltantes por la parte entera de dicho valor promedio. - En las columnas categóricas, reemplazaremos los valores faltantes por el valor modal de cada variable en cada zona. Destacar que si el valor modal es *NA*, lo sustituiremos por el siguiente valor categórico más frecuente dentro de la zona climática.

La decisión del uso de estos estadísticos para imputar los valores faltantes de cada variable queda respaldada por el hecho de que el uso de estos estadísticos minimiza el impacto de la imputación sobre los propios estadísticos. Esto es, sustituir valores faltantes por la media no modifica la media de la distribución subyacente.

```r
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)
}
```

```r
# 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(.)))
}

df_missings <- get_missings()
df_missings %>% View()
```

Vemos finalmente que tras el proceso de imputación detallado anteriormente ya no contamos con ningún valor faltante en ninguna variable.

Outliers Mostramos a continuación una serie de diagramas de cajas y violín para cada variable separado por valor de la variable objetivo, para cada zona climática.

```r
get_outliers <- function(){
  plots_list <- list()
  for(i in 1:length(zonas)){
    ps <- list()
    db_temp <- zonas[[i]][,c(columnas_enteras,columnas_numericas)]
```

```
    for(colu in db_temp %>% colnames() %>% setdiff(.,"RainTomorrow")){
      p <- ggplot(zonas[[i]], aes_string(x="RainTomorrow", y=colu, color="RainTomorrow")) + geom_violin
        axis.text.x = element_blank(),
        axis.text.y = element_text(size=6),
        axis.title.x = element_text(size = 8),
        axis.title.y = element_text(size = 8),
        legend.key.size = unit(0.1, 'cm'),
        legend.text = element_text(size=8),
        legend.title = element_blank())
      ps[[colu]] <- p
    }
    new_name <- paste0("zona",as.character(i))
    plots_list[[new_name]] <- ggarrange(plotlist =  ps, nrow = 4,ncol = 3,common.legend = TRUE)
    dev.off()
  }
  return(plots_list)
}

plot_list <- get_outliers()

# For zona in names(plot_list) print nicely test["zona1"]$zona1
```

Debido a la naturaleza caótica del tiempo, no vamos a tratar los outliers como valores atípicos porque consideramos que son de valor para el entrenamiento de los modelos. Nos limitaremos entonces a realizar una normalización de los datos numéricos junto con una selección de variables aplicando por un algoritmo de *ranking* de variables tipo *step*.

Transformación de las variables categóricas Las variables categóricas que disponemos son *Location*, *WindGustDir*, *WindDir9am*, *WindDir3pm*, *RainToday* y *RainTomorrow*. Para *Location*,*RainToday* y *RainTomorrow* vamos a aplicar técnicas de one hot encoding con las que transformaremos estas variables en numéricas. Por otro lado, para las variables categóricas relativas a la dirección del viento, vamos a aplicar una transformación de forma que combinemos esta información con las variables enteras relativas a la intensidad del viento, obteniendo dos variables numéricas que expresan el valor del coseno y del seno del vector viento.

```
# Iterar por zonas
dmy <- dummyVars( ~ +Location+RainToday+RainTomorrow, data = zonas[[1]])
trsf <- data.frame(predict(dmy, newdata = zonas[[1]]))
zonas[[1]] <- zonas[[1]] %>% select(., -any_of(c("Location","RainToday","RainTomorrow"))) %>% cbind(.,t
zonas[[1]] %>% head() %>% View()
```

De esta manera, tras el proceso de *one hot encoding*, explotamos cada variable categórica con n niveles en n variables binarias.

Finalmente, asignamos un valor en radianes a cada dirección del viento, calculamos el coseno y el seno de dicha dirección y los multiplicamos por la variable respectiva de intensidad del viento.

```
radianes <- list("E"=0,"ENE"=pi/8, "NE"=pi/4, "NNE"=3*pi/8,
                 "N" = pi/2,"NNW"=5*pi/8,"NW"=3*pi/4,"WNW"=7*pi/8,
                 "W"=pi,"WSW"=9*pi/8,"SW"=5*pi/4,"SSW"=11*pi/8,
                 "S"=3*pi/2,"SSE"=13*pi/8,"SE"=7*pi/4,"ESE"=15*pi/8)

zonas[[1]] %>% transform(., WindGustDir=radianes[as.character(zonas[[1]]$WindGustDir)]) %>% head() %>% 
```

Normalización de datos. justificar la normalización. Bien para el modelado, bien para el FS

Eliminación de variables y representación de los datos

Feature selection

RESUMEN DE TODO: - Las columnas categóricas son:"Location" "WindGustDir" "WindDir9am" "WindDir3pm" "RainToday" "RainTomorrow". Vamos a sustituir las Wind_cosas por dos variables cada una, que indican el valor del seno (componente norte-sur) y el coseno(componente este-oeste) de la dirección del viento. Además, como tenemos información de la intesidad del viento por hora, multiplicaremos estos valores para agregar la información. Con este criterio, transformamos variablas variables numéricas y categóricas, las cuales con el resto de variables numéricas someteremos a un algoritmo step para seleccionar las más relevantes para el modelado. Adicionalmente, nos restan las variables categóricas "Location" y "RainToday", las cuales someteremos a un onehot encoding para el modelado.

**Con las numéricas normalizamos y hacemos el step* y con las categóricas hacemos PCA

Factorial Analysis of Mixed Data (FAMD)

As presented in[11], hemos seguido los pasos que proponen a la hora de trabajar, es decir:

- Cosas de la lista

```
summary(lm1 <- lm(Fertility ~ ., data = swiss))
slm1 <- step(lm1)
summary(slm1)
slm1$anova
```

Representación

Distribuciones (histogramas)

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

Creación de nuevas variables ventana para trend y seasonality

Crear ventanas para RainToday de estadísticos rolling patrá

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).

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